TUGAS AKHIR

WORDNET BROWSER UNTUK IPHONE / IPAD

Oleh :

Arif Muliadi Chandra

207115640

JURUSAN TEKNIK INFORMATIKA

SEKOLAH TINGGI TEKNIK SURABAYA

SURABAYA

2011

TUGAS AKHIR

WORDNET BROWSER UNTUK IPHONE / IPAD

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar
Sarjana Teknik Informatika
pada
Sekolah Tinggi Teknik Surabaya
S u r a b a y a

Mengetahui/Menyetujui
Dosen Pembimbing

(Ir. Gunawan, M.Kom.)

SURABAYA
SEPTEMBER, 2011

ABSTRAK

Pada tugas akhir ini dibuat sebuah ekstraktor Lexical Database untuk bahasa pemrograman Objective-C dan sebuah browser untuk menampilkan informasi hasil ekstraksi. Lexical database merupakan suatu sumber relasi semantik dalam suatu bahasa. Browser yang dibuat pada tugas akhir dikembangkan untuk iPhone dan iPad.

Ekstraksi Lexical Database terdiri dari dua tahap. Tahap pertama adalah ekstraksi pada kelompok index files. Hasil ekstraksi pada tahap ini berupa synset offset dan relasi yang dimiliki oleh kata. Pada tahap ini synset offset pemilik relasi masih belum diketahui.Tahap kedua adalah ekstraksi pada kelompok data files. Hasil ekstraksi pada tahap ini berupa gloss, sinonim dan relasi yang dimiliki oleh synset. Pada tahap ini synset pemilik dari relasi telah diketahui dan apakah relasi tersebut adalah relasi semantik atau lexical.

Hasil dari ekstraksi akan ditampilkan melalui aplikasi WordNet Browser. Aplikasi tersebut akan menampilkan informasi mengenai kata yang dicari oleh pengguna. Apabila sebuah kata tidak terdapat pada Lexical Database maka aplikasi akan menampilkan kata-kata yang memiliki kemiripan penulisan dengan kata yang diinputkan oleh pengguna.

ABSTRACT

In this final project, a Lexical Database extractor for programming language Objective-C and a browser to show the extraction result are made. Lexical database is a source of semantic relations in a language. The browser made in this final project is developed for iPhone and iPad.

Lexical Database extraction consists of two stages. The first stage is the extraction of the index files group. The extraction results at this stage are synset offsets and relations owned by the words. At this stage the synset offset that owns the relation is still unknown. The second stage is the extraction of the data files group. The extraction results at this stage are gloss, synonym and relations owned by the synsets. At this stage the synset offset that ownss the relation is known and whether the relation is a semantic or lexical relation is also known.

The results of the extraction will be displayed through WordNet Browser application. The application will display information owned by the word that the user is searching. If a word is not found in Lexical Database then the application will display the words that are similarly spelled with the word entered by the user.

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas berkat yang telah diberikannya, sehingga Tugas Akhir ini dapat terselesaikan dengan baik dan tepat pada waktunya. Selama penyusunan hingga selesainya Tugas Akhir ini, penulis telah banyak menerima dorongan, bantuan, dan perhatian dari berbagai pihak. Oleh karena itu pada kesempatan ini, penulis ingin mengucapkan terima kasih kepada:

1. Kedua orang tua, yang telah memberikan dukungan dan motivasi baik berupa material maupun spiritual dalam penyelesaian Tugas Akhir ini.
2. Bapak Ir.Gunawan M.Kom, selaku dosen pembimbing yang banyak memberikan pengarahan, koreksi, serta nasehat kepada penulis selama penyusunan Tugas Akhir ini.
3. Semua dosen pengajar yang telah memberikan bimbingan dan pengajaran selama masa perkuliahan di Sekolah Tinggi Teknik Surabaya.
4. Glenn Kristanto, Yugatha Ivanno, dan Jaya Pranata, teman-teman yang tergabung dalam cocoa head yaitu pengembang aplikasi untuk iPhone dan iPad.
5. Rekan-rekan asisten Laboratorium Komputer, Felix Kurniawan, Andy Saputra, David Alexandre, Joan Santoso, Ronny Honggo, Steve Siebert, Henry Roes Lie, Rendy Setiawan, Edwin Edward S., Aretta, Ezra Christopher, Christian Iwanto, Sherly Tandriono, Ignatia Jovita, Steven Ihan, Sandy Ardianto serta tidak lupa koordinator laboratorium, Bapak Hendrawan Armanto.
6. Yonathan Randyanto, Euridyce, Hermes Vincentius, Melissa Tedjokusumo, Po Dede Christian, Michael Frans dan teman-teman lain di STTS yang senantiasa memberikan dukungan dan bantuan kepada penulis.

Pada kesempatan ini pula, penulis juga ingin mengucapkan terima kasih kepada semua pihak yang belum disebutkan sebelumnya, atas segala bentuk bantuan dan dukungan yang telah dilakukan. Tanpa bantuan dan dukungan dari

pihak-pihak tersebut, penulis tidak akan mampu menyelesaikan Tugas Akhir ini dengan baik.

Ibarat *tiada gading yang tak retak*, begitu pula pengerjaan Tugas Akhir ini tentu tidak luput dari kekurangan-kekurangan yang menyertai. Oleh karena itu penulis sangat mengharapkan kritik dan saran dari pembaca. Akhir kata, semoga Tugas Akhir ini memberikan manfaat bagi para pembaca.

Surabaya, September 2011

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Gambar **Halaman**

DAFTAR TABEL

DAFTAR ALGORITMA

Algoritma **Halaman**

DAFTAR SEGMEN PROGRAM

BAB I
PENDAHULUAN

Dalam bab ini akan dijabarkan mengenai latar belakang pengerjaan Tugas Akhir, tujuan yang diharapkan, ruang lingkup yang menjadi batasan dalam pengerjaan Tugas Akhir ini, metodologi yang digunakan, serta sistematika pembahasan yang akan menjelaskan isi dari tiap bab dalam buku Tugas Akhir ini secara ringkas.

1.1 Latar Belakang

Saat ini mobile device telah mengalami perkembangan yang sangat pesat. Pada awalnya mobile device hanyalah sebuah alat untuk berkomunikasi jarak jauh, tetapi dengan semakin banyaknya pengguna mobile device dan semakin berkembangnya teknologi yang mendukung mobile device maka para produsen pun terus melakukan penelitian untuk meningkatkan kualitas dari mobile device mereka.Seiring dengan perkembangan jaman, mobile device kini telah menjadi sebuah perangkat yang vital bagi kehidupan manusia, bukan hanya sebagai alat berkomunikasi tetapi juga sebagai reminder, calendar, browser, gaming device, translator dan sebagainya.

iPhone merupakan salah satu mobile device yang terkenal dan sukses di kalangan masyarakat. Untuk mendukung perkembangan aplikasi iPhone maka iPhone menyediakan SDK yang dapat digunakan oleh para programmer untuk membuat aplikasi pada iPhone atau iPad serta meletakkannya pada iTunes store supaya para pengguna iPhone atau iPad dapat mengunduhnya. Dengan adanya SDK ini maka munculah berbagai macam aplikasi pada iPhone dengan fungsi yang berbeda-beda. Misalnya permainan Grand Theft Auto, sarana pembelajaran bahasa inggris Miss Spell’s Class, dan sebagainya.

Salah satu fungsi dari iPhone adalah sebagai sarana pembelajaran bahasa. Hal ini ditunjukkan oleh aplikasi-aplikasi serperti Miss Spell’s Class dan Advanced English Dictionary and Thesaurus. Kedua aplikasi tersebut membantu

pengguna untuk lebih mendalami arti serta penulisan kata-kata berbahasa Inggris. Terdapat juga aplikasi kamus untuk bahasa Prancis seperti Larousse. Untuk setiap bahasa yang ingin dipelajari maka pengguna dapat mencari aplikasi kamus untuk bahasa tersebut pada iTunes. Hal ini cukup merepotkan bagi para pengguna yang ingin mempelajari berbagai bahasa, selain itu biaya yang dikeluarkan juga akan semakin besar.

Jumlah aplikasi kamus Bahasa Indonesia dalam iTunes masih tergolong sedikit dan belum berkembang. Aplikasi-aplikasi tersebut hanya menampilkan arti dari kata yang dicari saja dan tidak mengandung relasi-relasi dari kata tersebut seperti sinonim, hipernim, holonim dan sebagainya. Akan sangat baik apabila terdapat sebuah aplikasi kamus Bahasa Indonesia yang mengandung sense, gloss dan relasi-relasi yang dimiliki oleh kata tersebut.Contoh aplikasi yang merupakan kamus Bahasa Indonesia adalah Kamus Lengkap Pro, aplikasi ini menyediakan fasilitas translasi Bahasa Indonesia ke Bahasa Inggris.

1.2 Tujuan

Tugas Akhir ini bertujuan membuat sebuah Wordnet Browser Bahasa Indonesia yang mengandung sense, gloss dan relasi-relasi semantik dari kata yang dicari oleh pengguna. Wordnet Browser ini berjalan pada platform iPhone dan juga iPad.

1.3 Ruang Lingkup

Ruang lingkup yang akan dibahas pada Tugas Akhir ini adalah hal-hal sebagai berikut:

1. Arsitektur Sistem.

   Lexicographer File akan mengalami proses kompilasi oleh Grinder sehingga menjadi file lexical database dengan format yang sama dengan format standard yang digunakan pada Princeton WordNet (PWN). Princeton WordNet adalah sebuah WordNet berbahasa Inggris yang dikembangkan oleh Princeton University. PWN merupakan salah satu

sumber WordNet berbahasa Inggris yang sering dipakai dan strukturnya diikuti oleh aplikasi-aplikasi Wordnet lain.

Lexical Database tersebut terdiri dari 2 jenis file yaitu : index file dan data file. Index file berisi file dengan nama index dan diikuti jenis kata, misalnya index.noun. Index file ini berisi daftar semua kata yang ada diikuti dengan synonymset offset yang akan digunakan untuk melakukan ekstraksi gloss, sense dan relasi semantik yang diperlukan.Data file berisi file dengan Synonym offset yang diikuti dengan synonym offset dari kata lain disertai dengan hubungan antara kedua kata tersebut. Synonym offset ini diakhiri dengan gloss beserta example dari synonym set tersebut.

Proses pencarian kata dimulai dengan mencari class kata tersebut pada index file yang tersedia, pencarian ini dilakukan dengan menggunakan algoritma binary search. Dari hasil pencarian tersebut akan didapatkan Synonym offset dari kata tersebut, synonym offset ini kemudian akan dicari pada data file. Pencarian ini akan menghasilkan gloss, sense, sinonim set dan relasi semantik dari kata tersebut.

2. Input dan output

Pada subbab ini akan dijelaskan mengenai input dan output pada Wordnet Browser Bahasa Indonesia yang akan dibuat pada Tugas Akhir ini beserta dengan contohnya :

A. Input :

Input dari program adalah sebuah kata yang ingin dicari oleh pengguna. Berbagai pilihan juga dapat dipilih oleh pengguna, misalnya apakah gloss dari sense yang ada ingin ditampilkan atau tidak, sense nomor berapakah yang ingin ditampilkan.

B. Output :

Output dari program adalah sense, gloss, sense number, synset dan relasi-relasi semantik dari kata yang dicari. Hasil yang didapatkan akan dikelompokkan berdasarkan sense yang ada, selain itu semua kata yang pernah ada akan dicatat ke dalam history.

Relasi semantik yang didapatkan adalah sebagai berikut :

- Hipernim
- Hiponim
- Holonim
- Meronim

Tidak semua kata memiliki relasi semantik yang telah disebutkan.

C. Contoh input dan output :

Input : binatang

Output :

1. kata benda (makhluk bernyawa yang mampu bergerak terhadap rangsangan tapi tidak berakal budi)

Synset : binatang,fauna,hewan

3. Ketersediaan Resource :

Resource yang digunakan sebagai sumber data pada Tugas Akhir ini adalah Lexical Database Bahasa Indonesia hasil dari Tugas Akhir Jessica Felani Wijoyo. Lexical Database ini sudah diperbaiki secara kolaboratif / manual untuk sejumlah synset dan relasi.

4. Batasan :

Bagian ini akan menjelaskan tentang batasan yang terdapat pada Tugas Akhir ini. Batasan yang terdapat pada Tugas Akhir ini adalah :

A. Hasil ekstraksi synset, gloss, sense dan relasi semantik yang dihasilkan bukanlah konsentrasi dari Tugas Akhir ini. Tugas Akhir ini memanfaatkan hasil dari ekstraksi tersebut untuk pembuatan Wordnet Browser

B. Pemrograman ditujukan secara utama kepada iPhone agar dapat berjalan pada iPad dengan persyaratan SDK yang digunakan versi 4.2 / 4.3 dan Xcode dengan versi 3.25 / 4.0 serta iOS yang digunakan oleh iPhone adalah versi 4.0 ke atas. Apabila

pemrograman ditujukan kepada iPad terdapat kemungkinan tidak akan dapat berjalan di iPhone disebabkan oleh adanya fitur-fitur khusus yang hanya dimiliki oleh iPad.

5. Fitur program :

   Adapun fitur-fitur yang terdapat pada Tugas Akhir ini adalah sebagai berikut :

   A. Show/hide gloss

      Fitur ini memberi pilihan pada pengguna apakah gloss dari sebuah kata akan ditampilkan atau tidak. Secara default maka gloss akan ditampilkan.

   B. Show only class X [,Y]

      Pengguna dapat memilih apakah ingin menampilkan class tertentu saja dari suatu kata. Pengguna dapat memilih lebih dari satu buah class misalnya pengguna ingin menampilkan kata sifat dan kata benda.

   C. Group by class

      Secara otomatis browser akan melakukan grouping dari gloss yang didapatkan berdasarkan class dari kata tersebut. Class yang ditampilkan urut secara alphabetical yaitu, kata benda, kata kerja, kata keterangan, kata sifat.

   D. Auto Complete Text pada saat Searching

      Text yang diinputkan user akan mendapat saran dari browser kata apa yang mungkin ingin diinputkan oleh user. Fitur ini akan berguna bagi para pengguna dalam menginputkan kata yang ingin dicari.

   E. History

      Browser akan menyimpan kata-kata yang pernah dicari oleh pengguna. Apabila pengguna ingin mencari kata yang sudah pernah dicari sebelumnya maka cukup dengan mengakses history saja,

tidak perlu mengetikkan ulang kata yang ingin dicari. Browser akan menyimpan 10 kata yang terakhir kali dicari oleh pengguna.

F. Bookmark

Pengguna dapat melakukan bookmark kepada kata-kata yang ingin disimpan atau sering dilihat. Dengan memanfaatkan fitur ini maka pengguna tidak perlu mengetik berulang kali untuk melihat kata yang sama.

1.4 Metodologi

Tugas Akhir ini merupakan Tugas Akhir yang memanfaatkan Lexical Database Bahasa Indonesia yang telah dihasilkan dan diproses oleh berbagai Tugas Akhir yang lain untuk menciptakan Wordnet Browser pada iPhone / iPad. Dalam pembuatannya Wordnet Browser tersebut akan terus dikembangkan agar dapat mencapai hasil yang lebih baik dari ide dasar yang digunakan.

Proses yang dilakukan selama pembuatan Tugas Akhir ini adalah mempelajari pemrograman dengan bahasa Objective-C dengan memanfaatkan Xcode dan Cocoa Touch. Selain itu, struktur Lexical Database Bahasa Indonesia juga dipelajari agar dapat diekstrak dengan baik dan bisa mendapatkan semua relasi-relasi semantik yang terdapat dalam Lexical Database tersebut. Agar bisa mendapatkan hasil yang maksimal maka akan dilakukan juga berbagai diskusi dengan pembimbing dan mahasiswa lain yang terkait dengan proses pembentukan lexical database Bahasa Indonesia. Untuk menguji fungsionalitas serta kualitas dari program maka akan dilakukan survey untuk mengetahuinya. Hasil survey tersebut akan menjadi dasar untuk pengembangan dan perbaikan dari Wordnet Browser Bahasa Indonesia.

Hasil yang diperoleh dari percobaan ini dapat dianggap sebagai hasil prototype dari Wordnet Browser Bahasa Indonesia, untuk selanjutnya terus dikembangkan dan ditingkatkan kualitasnya. Akurasi hasil dari proses pencarian yang dilakukan sangat tergantung dari sumber Lexical Database yang didapatkan dari Tugas Akhir Jessica Felani Wijoyo. Meskipun demikian, hasil dari Tugas Akhir ini diharapkan dapat menunjang Tugas Akhir lainnya yang memanfaatkan

Lexical Database Bahasa Indonesia serta mendukung perkembangan aplikasi iPhone / iPad berbahasa Indonesia agar menjadi lebih baik dan dapat berkembang lebih jauh.

1.5 Sistematika Pembahasan

Sistematika pembahasan yang digunakan pada buku Tugas Akhir ini terbagi menjadi delapan bab. Isi pembahasan masing-masing bab akan mencakup hal-hal seperti berikut ini:

- Bab I : PENDAHULUAN

  Pada bagian ini dijelaskan mengenai latar belakang, tujuan pembuatan Tugas Akhir, ruang lingkup yang menjelaskan batasan-batasan dalam pembuatan program, metodologi yang digunakan dan sistematika pembahasan tiap bab.

- Bab II : TEORI DASAR

  Pada bagian ini akan dijelaskan mengenai dasar-dasar teori dari program Tugas Akhir ini. Pembahasa meliputi hardware iPhone dan iPad, bahasa pemrograman Objective-C, Xcode, aplikasi yang ada didalam Xcode sebagai IDE yang digunakan dalam pembuatan Tugas Akhir ini, juga sejarah pembangunan WordNet dan struktur yang digunakan. Selain itu juga dijelaskan mengenai algoritma Levenshetein Distance yang digunakan untuk menghitung jarak antar kata.

- Bab III : ANALISA DAN DESAIN SISTEM PADA WORDNET BROWSER

  Pada bagian ini akan dijelaskan mengenai gambaran umu dan arsitektur sistem dari WordNet Browser yang dibuat pada Tugas Akhir ini. Selain itu akan dibahas juga mengenai perbandingan antara WordNet Browser yang dibuat dengan aplikasi sejenis yang lain pada iPhone / iPad serta desktop dan web. Desain interface yang dimiliki oleh WordNet Browser juga akan dijelaskan secara detail.

- Bab IV : EKSTRAKSI INFORMASI LEXICAL DATABASE FILES

  Pada bagian ini akan dijelaskan mengenai proses ekstraksi informasi yang dilakukan pada Lexical Database Files. Pembahasan akan meliputi struktur penyimpanan hasil ekstraksi, proses ekstraksi yang dilakukan serta class yang digunakan sebagai ekstraktor Lexical Database Files.

- Bab V : EKSTRAKSI INFORMASI OLEH WORDNET BROWSER

  Pada bagian ini akan dijelaskan mengenai proses ekstraksi informasi yang dilakukan oleh WordNet Browser. Pembahasan akan meliputi pencarian kata oleh user, ekstraksi informasi yang dimiliki oleh kata serta menampilkan informasi tersebut kepada user.

- Bab VI : FITUR-FITUR TAMBAHAN WORDNET BROWSER

  Pada bagian ini akan dijelaskan mengenai fitur-fitur tambahan yang dimiliki oleh WordNet Browser. Fitur tersebut akan meliputi bookmark, history dan setting. Untuk setiap fitur tersebut akan dijelaskan mengenai fungsi dan proses yang dilakukan.

- Bab VII : UJI COBA APLIKASI

  Pada bagian ini akan ditunjukkan mengenai uji coba yang dilakukan terhadap aplikasi yang dibuat pada Tugas Akhir ini. Pembahasan akan meliputi analisa Lexical Database yang digunakan, fungsionalitas aplikasi serta kuesioner yang dilakukan.

- Bab VIII : PENUTUP

  Pada bagian ini berisi kesimpulan dari aplikasi yang dibuat pada Tugas Akhir ini. Pada bab ini juga disertakan saran mengenai kemungkinan pengembangan pada percobaan yang sudah dilakukan.

BAB II
TEORI DASAR

Pada bab ini akan dijelaskan mengenai iPhone, Xcode sebagai Software Development Kit yang dipergunakan pada Tugas Akhir ini dan struktrur dari WordNet sebagai sumber Lexical Database. Penjelasan dari iPhone akan mencakup perkembangan dari mobile device ini, dan juga fitur-fitur yang membuat iPhone memiliki nilai lebih tersendiri dibanding dengan mobile device lain. Penjelasan mengenai Xcode akan meliputi penjelasan dari perkembangan iPhone SDK dan fitur-fitur didalamnya. Sedangkan penjelasan mengenai WordNet akan meliputi struktur dari WordNet serta struktur penyimpanan data pada index dan data file.

2.1 iPhone

iPhone adalah sebuah mobile device yang dikembangkan dan dijual oleh Apple, Inc. Dalam perkembangannya iPhone membawa dampak perubahan yang besar dalam perkembangan mobile device. Fitur-fitur yang dimiliki oleh iPhone menaikkan standar fungsionalitas yang harus dimiliki oleh sebuah mobile device. Pada subbab ini akan dijelaskan mengenai perkembangan iPhone dan fitur-fitur yang dimilikinya.

2.1.1 Pengenalan iPhone

iPhone adalah mobile device yang dikenalkan pada publik pada Januari 2007, dan pertama kali dijual ke publik pada bulan Juli 2007. Sampai saat ini sudah ada 4 generasi iPhone yang diluncurkan oleh Apple, Inc. yaitu iPhone 2G yang diluncurkan pada tahun 2007, iPhone 3G yang diluncurkan pada tahun 2008, iPhone 3GS yang diluncurkan pada tahun 2009, dan yang terakhir iPhone 4 yang diluncurkan pada tahun 2010.

Apple memperkenalkan iPhone sebagai alat 3-in-1. Fitur-fitur utama yang dikenalkan oleh Apple tersebut adalah fitur telepon, iPod dan internet device.

Seperti mobile device pada umumnya, iPhone dapat melakukan dan menerima panggilan telepon. Disamping itu iPhone juga dapat digunakan sebagai pemutar musik dan video dengan memanfaatkan aplikasi iPod yang ada didalamnya. Sebagai tambahan, iPod adalah alat pemutar music berformat .mp3 milik Apple yang sangat terkenal. iPhone juga memiliki HTML browser yang bernama Safari. Para pengguna iPhone juga dapat menambahkan aplikasi-aplikasi yang dibutuhkan ke dalam iPhone dengan mengunduhnya melalui AppStore. Aplikasi tersebut dapat berupa permainan, kamus, gps, dan berbagai jenis aplikasi lain.

Gambar 2.1
iPhone 4

Generasi iPhone yang pertama, yaitu iPhone 2G masih memiliki fitur-fitur yang sangat minimal. Setahun berikutnya diperkenalkan iPhone 3G, yang memiliki fitur 3G sehingga kecepatan pengiriman data meningkat. Disamping itu pada tahun 2008, Apple juga meluncurkan iPhone SDK sehingga saat iPhone 3G diluncurkan, iPhone sudah mendukung aplikasi dari pihak ketiga. Pada tahun 2009, Apple mengenalkan iPhone 3Gs, yang memiliki tambahan kecepatan prosessor, dan memiliki kamera yang lebih baik sehingga mampu merekam video

dan juga memiliki beberapa fitur tambahan lainnya. iPhone 4 diperkenalkan pada tahun 2010. Memiliki perubahan jauh dibanding iPhone sebelumnya dengan memiliki fitur retina display, dimana kepadatan pixel didalam layar iPhone 4 mencapai angka 312 dpi, padahal mata manusia hanya mampu menangkap kepadatan sampai angka 300 dpi saja. Disamping itu iPhone 4 juga memiliki kamera didepan, yang memungkinkan pengguna untuk melakukan video call.

2.1.2 Fitur-fitur iPhone

Pada subbab ini akan dibahas spesifikasi hardware yang mendeskripsikan iPhone secara umum dan fitur-fitur yang dimiliki oleh iPhone. Berikut adalah spesifikasi iPhone 4 secara teknis.

- Berat : 137 gram
- Dimensi : 115.2mm x 58.6mm x 9.3mm
- Layar : TFT capacitive touchscreen, Retina display, 960 x 640 pixels
- RAM : 512MB
- OS : IPhonse OS (iOS)
- WLAN : Wi-Fi 802.11 b/g/n
- Koneksi Data : GPRS, EDGE, 3G, WLAN, Bluetooth, USB
- Kamera : 5MP
- Memori : 16GB / 32 GB

iPhone memilki touchscreen yang sudah menggunakan teknologi capacitive. Teknologi capacitive berbeda dengan teknologi sebelumnya, yaitu resistive. Pada resistive touchscreen, system akan mendeteksi sentuhan pengguna menggunakan semacam lapisan tipis yang akan menyentuh lapisan dibawahnya. Sentuhan antar kedua lapisan inilah yang akan dideteksi sebagai sentuhan dari pengguna. Sedangkan pada teknologi capacitive touchscreen, lapisan kaca dari layar akan dilapisi oleh konduktor, biasanya menggunakan Indium Tin Oxide (ITO). Karena tubuh manusia juga merupakan konduktor, maka saat menyentuh

lapisan konduktor tersebut akan menimbulkan arus listrik. Arus inilah yang akan dideteksi sebagai sentuhan dari pengguna.

Salah satu fitur revolusioner dari iPhone 4 adalah fitur retina display. Fitur retina display berhasil memasukkan resolusi 960 x 640 pixels ke dalam layar iPhone berukuran 3,5 inci dengan cara menggunakan pixel sebesar 78 micrometer. Mata manusia hanya dapat melihat sebuah pixel hingga 300dpi, dengan resolusi yang diberikan oleh iPhone 4 sebesar 326dpi (lebih dari 300 pixel per inci) mengakibatkan Tampilan pada layar iPhone 4 semakin tajam.

Gambar 2.2
Retina Display

iPhone 4 juga menggunakan teknologi IPS (in-plane-switching), yaitu teknologi yang memberikan sudut pandang yang lebih besar pada LCD. Para pengguna iPhone dapat melihat layar iPhone dari berbagai sudut dan tetap mendapatkan tampilan yang tajam. Teknologi ini juga diterapkan pada tablet komputer milik Apple,yaitu pada iPad 2. Detail mengenai spesifikasi dan kemampuan dari iPad dan iPad 2 akan dijelaskan pada subbab berikutnya beserta dengan fitur-fitur yang dimiliknya.

2.2 iPad

iPad adalah sebuah komputer tablet yang diproduksi oleh Apple, Inc. Produk ini dirancang sebagai sebuah perangkat digital yang berada diantara smart phone dan laptop. Mesikpun menggunakan OS yang sama dengan iPhone tetapi iPad memiliki kelebihan dan kekurangan tersendiri dibandingkan dengan iPhone. Pada subbab ini akan dijelaskan mengenai perkembangan iPad dan fitur-fitur yang dimilikinya.

2.2.1 Pengenalan iPad

iPad adalah sebuah tablet yang dikenalkan pada publik pada 27 Januari 2010, dan pertama kali dijual pada publik pada 3 April 2010. Sampai saat ini sudah ada 2 generasi ipad yang diluncurkan oleh Apple, Inc. yaitu iPad 1 yang diluncurkan pada 2010 dan iPad 2 yang diluncurkan pada tahun 2011.

Gambar 2.3
iPad

Generasi iPad pertama sudah memiliki fitur yang cukup lengkap, dimulai dari fitur telepon, internet device, iPod, video Player, ebook reader dan

sebagainya. Karena tidak memiliki sim card maka internet access hanya dapat diperoleh melalui wi-fi, hal ini telah diatasi oleh iPad generasi kedua yang telah dapat menggunkan sim card untuk mendapatkan internet access.

Sama seperti iPhone, iPad hanya memiliki 4 buah tombol, yaitu tombol home, volume, mute, dan sleep-wake. Hampir keseluruhan interaksi dengan pengguna dilakukan memalui touch screen. Berbeda dengan touch screen pada umumnya, iPhone dan iPad mendukung teknologi multi-touch yang memungkinkan iPhone dan iPad untuk mendeteksi dua atau lebih sentuhan pada saat yang bersamaan.

2.2.2 Fitur-Fitur iPad

Pada subbab ini akan dibahas spesifikasi hardware yang mendeskrpisikan iPad generasi Pertama dan generasi kedua serta fitur-fitur umum yang dimiliki oleh iPad. Tabel 2.1 menunjukkan spesifikasi iPad secara teknis.

Tabel 2.1
Spesifikasi iPad 1 & iPad2

	iPad 1	iPad 2
Berat	601 gram	613 gram
Dimensi	241,2 x 185,7 x 8.8 mm	241,2 x 185,7 x 8.8 mm
Layar	1024 x 768 pixels	1024 x 768 pixels
Processor	A4 processor	A5 processor
Ram	256MB	512MB
OS	iOS	iOS
WLAN	Wi-Fi 802.11 a/b/g/n	Wi-Fi 802.11 a/b/g/n
Koneksi Data	Bluetooth, USB	Bluetooth, USB, 3G
Kamera	Tidak ada	5x Digital Zoom
Memori	16GB / 32GB / 64GB	16GB / 32GB / 64GB

Generasi iPad pertama memiliki beberapa kekurangan dibandingkan dengan iPad dari generasi kedua. Dapat dikatakan bahwa iPad 2, lebih cepat dan

memiliki lebih banyak fitur dibandingkan dengan iPad 1. Salah satu perubahan besar dari iPad 2 adalah kekuatan dari processor dan ramnya. iPad 2 menggunakan 1 GHz dual core A5 processor dan ram sebesar 512 MB sedangkan iPad 1 menggunakan 1 GHz A4 processor dan ram sebesar 256 MB. Processor A5 memiliki clock speed yang dua kali lebih cepat dibandingkan dengan A4 dan 9 kali lebih baik ketika memproses grafik serta penggunaan baterei yang lebih hemat.

Perbedaan utama antara fitur iPhone dan iPad adalah pada Interface, fitur telepon dan resolusi layar. iPad memiliki beberapa interface yang tidak dimiliki oleh iPhone (sliding menu), iPad tidak dapat digunakan untuk menelpon walaupun pada iPad generasi kedua dapat menggunakan sim card. Resolusi layar yang dimiliki iPad adalah 1024 x 768 pixels, lebih besar dari pada iPhone.

2.3 Xcode

Xcode adalah IDE (Integrated Development Environment) utama untuk Mac. Xcode tidak hanya digunakan untuk mengembangkan aplikasi iPhone tetapi juga aplikasi untuk mac dan iPad. Didalam Xcode terdapat berbagai macam tool yang memiliki kegunaan berbeda-beda. iPhone SDK adalah Software Development Kit yang dibuat oleh Apple untuk membuat aplikasi iPhone. iPhone SDK diperkenalkan oleh Apple ke publik pada tahun 2008. Agar memudahkan pengguna Mac, maka iPhone SDK ditanamkan secara langsung ke dalam Xcode. Sejak diperkenalkan ke publik, iPhone SDK langsung mendapat tanggapan hangat dari para pengembang software. Setahun setelah diluncurkan, iPhone SDK telah didownload sebanyak lebih dari 1.000.000 kali. iPhone SDK kini telah dapat digunakan untuk membuat aplikasi untuk iPad dan iTouch yang juga menggunakan iOS.

iPhone SDK yang termasuk ke dalam Xcode terdiri atas beberapa aplikasi, yaitu: Xcode sebagai tempat pemrograman utama, Interface Builder untuk mendesain tampilan aplikasi yang dibuat, Instrument untuk menganalisa performa dari aplikasi yang dibuat, dan iPhone Simulator untuk mencoba aplikasi yang dibuat. Pada subbab ini akan dibahas mengenai Objective-C, yaitu bahasa yang

digunakan untuk membuat aplikasi di iPhone, dan macam-macam aplikasi yang ada di Xcode.

2.3.1 Objective-C

Objective-C adalah bahasa pemrograman yang merupakan pengembangan dari bahasa pemrograman C. Pertama kali diciptakan pada tahun 1980 dan digunakan oleh perusahaan komputer yang dikembangkan oleh Steve Jobs yaitu NeXT. Beberapa tahun kemudian Apple mengakuisisi NeXT dan menggunakan Objective-C sebagai bahasa pemrograman utama di Macintosh. Pada tahun 2008 Objective-C juga digunakan sebagai bahasa pemrograman untuk iPhone.

Karena Objective-C merupakan pengembangan dari bahasa pemrograman C maka syntax-syntax yang digunakan relatif sama. Yang membedakan keduanya adalah sifat Objective-C, yaitu Object Oriented. Pada Objective-C sebuah class terbagi menjadi 2 buah file, yaitu file dengan ekstensi *.m dan *.h. File dengan ekstensi *.h dikenal dengan file Interface yang berisi deklarasi dari property dan prosedur yang bersifat public. Segmen program 2.1 menunjukkan struktur interface dari Objective-C.

Segmen Program 2.1 Struktur Interface Objective-C

```
1: #import “otherClassName.h”
2: #import <Foundation/Foundation.h>

3: @interface className : superClassName {
4:    //variables declaration
5: }

6: +methodName;
7: +(returnType) methodName;
8: -(returnType) methodName:(paramType)paramName;
9: -(returnType) methodName:(param1Type)param1Name
                 param2Explain:(param2Type)param2Name;
10:@end
```

Struktur file interface pada Objective-C dapat dikelompokkan menjadi 3 bagian yaitu : bagian import, bagian deklarasi interface, dan bagian deklarasi method. Pada baris 1 dan 2 pada segmen program 2.1 merupakan contoh penggunaan import pada Objective-C. Terdapat 2 jenis import yang dapat

digunakan yaitu import dengan menggunakan tanda “ ” dan import dengan menggunakan tanda < >. Import dengan tanda < > digunakan apabila class yang akan diimport bukan merupakan class standar milik Xcode, contoh dari class standar milik Xcode adalah Foundation.h dan UIKit.h. Bila ingin menginmpor class yang bukan merupakan standar milik Xcode maka digunakan tanda “ “, misalnya #import “WordNet.h” digunakan untuk mengimport class WordNet.

Baris 3-5 pada segmen program 2.1 diatas merupakan bagian deklarasi interface. Pada bagian ini terdapat deklarasi superClassName, pada Objectice-C yang bersifat Object Oriented semua class harus memiliki super class. Bila class tersebut bukan merupakan turunan dari class manapun maka ia merupakan turunan dari class NSObject. Baris 6-9 merupakan bagian deklarasi method yang dimiliki oleh interface. Terdapat 2 jenis deklarasi method pada Objective-C yaitu deklarasi dengan diawali simbol + dan deklarasi dengan diawail simbol -. Apabila sebuah method dideklarasikan dengan diawali simbol + maka method itu adalah sebuah method yang bersifat statik, dapat dipanggil tanpa harus membuat instance dari class tersebut. Tetapi apabila sebuah method dideklarasikan dengan diawali tanda – maka instance dari class harus dibuat terlebih dahulu sebelum dapat memanggil method tersebut. Baris 10 merupakan penutup dari class interface.

Apabila file *.h berisi header dari method-method serta variable yang dimliki oleh sebuah class, maka file *.m atau yang dikenal dengan file implementation ini merupakan tempat penulisan keseluruhan program. Line of code yang berisi perintah yang akan dieksekusi oleh sebuah method akan pada file implementation ini. Segmen program 2.2 menunjukkan struktur dari file implementation.

Segmen Program 2.2 Struktur Implementation Objective-C

```
1: #import “headerFileName.h”
2: @implementation className
3: +methodName{
4:    //implementation
5: }
6: -(returnType) methodName:(paramType)paramName {
7: }
8: @end;
```

File implementation selalu diawali dengan baris 2 dan baris 8 segmen program 2.2, yaitu pendeklarasian implementation dan penutup dari class implementation. Baris 1 dari segmen program 2.2 merupakan line of code yang akan ditambahkan secara otomatis oleh Xcode. Setiap file implementation dari sebuah class harus mengimpor file header dari class tersebut. Semua class yang diimport pada file header akan dapat digunakan oleh file implementation, selain itu penambahan impor class juga dapat dilakukan secara langsung pada file implementation. Pada baris 3-5 dan 6-8 pada segmen program 2.2 menunjukkan contoh override dari method yang telah dideklarasikan pada file header. Method-method yang telah dideklarasikan pada file header harus dioverride pada file implementation. Method yang bersifat statik dan yang tidak bersifat statik memiliki cara override yang sama. Pada Objective-C juga dikenal istilah accessor dan mutator untuk mendapatkan dan mengubah nilai dari suatu variabel. Tidak seperti bahasa pemrograman berbasis object yang lain dimana accessor dan mutator harus dibuat secara manual, Obective-C memberikan kemudahan dengan membuatkan accessor dan mutator dari suatu variable secara otomatis dengan menambahkan sedikit line of code. Segmen program 2.3 menunjukkan contoh deklarasi accessor dan mutator untuk sebuah variable.

Segmen Program 2.3 Deklarasi Accessor dan Mutator Objective-C

```
1: @property(nonatomic,retain) variabelType variabelName;
2: @synthesize variabelName
```

Dengan menambahkan kedua line of code pada segmen program 2.3 maka Objective-C akan secara otomatis membuatkan accessor dan mutator untuk suatu variabel. Kedua baris line of code tersebut berada di dua buah file yang berbeda, baris pertama diletakkan pada file header di bagian deklarasi method sedangkan baris kedua diletakkan pada file implementation setelah deklarasi implementasi dilakukan. Nonatomic pada deklarasi property menandakan bahwa property tidak akan diakses secara thread-safe artinya terdapat kemungkinan akan terjadinya race-condition apabila dilakukan proses multi-threading. Sedangkan deklarasi retain menandakan bahwa property tersebut tidak akan dihapus dari memory oleh

garbage collector. Setelah kedua line of code tersebut dituliskan maka accessor dan mutator dari variable tersebut dapat diakses secara langsung. Terdapat dua cara untuk mengakses accessor dan mutator tersebut. Kedua cara tersebut adalah :

Segmen Program 2.4 Menggunakan Accessor dan Mutator Objective-C

```
1: NSString *x = [className variableName];
2: x = className.variableName;
3: [className setVariableName:x];
4: className.variableName = x;
```

Pada baris 1 segmen program 2.4 merupakan cara menggunakan accessor yang digunakan secara khusus oleh Objective-C, dimana variable dan class yang diakses menggunakan kurung siku. Sedangkan baris 2 pada segmen program 2.4 merupakan cara mengakses variable yang banyak dijumpai pada bahasa pemrograman yang lain, cara ini juga dapat digunakan pada Objective-C.

Pada Baris 3 segmen program 2.4 merupakan cara menggunakan mutator yang telah dibuatkan secara otomatis oleh Objective-C. Mutator ini memiliki nama method yang diawali oleh ‘set’ dan diikuti nama variable yang bersifat propercase. Baris 4 segmen program 2.4 merupakan cara kedua untuk menggunakan mutator pada Objective-C yaitu dengan mengakses nama variabel secara langsung, cara ini juga banyak dijumpai pada bahasa pemrograman yang lain.

Memory management merupakan hal yang sangat penting dalam pemrograman di iPhone, hal ini disebabkan karena memory iPhone yang sangat terbatas sehingga harus dimanfaatkan dengan sebaik mungkin. Objective-C memiliki garbage collector yang akan merelease variable dari memory apabila variabel tersebut sudah tidak lagi digunakan. Hal ini akan menjadi masalah apabila variabel tersebut telah direlease dan diakses oleh aplikasi, untuk mengatasi hal ini maka terdapat perintah retain.

Segmen Program 2.5 Retain dan Release Objective-C

```
1: NSString *x = [[NSString alloc]init];
2: [x retain];
3: [x release];
```

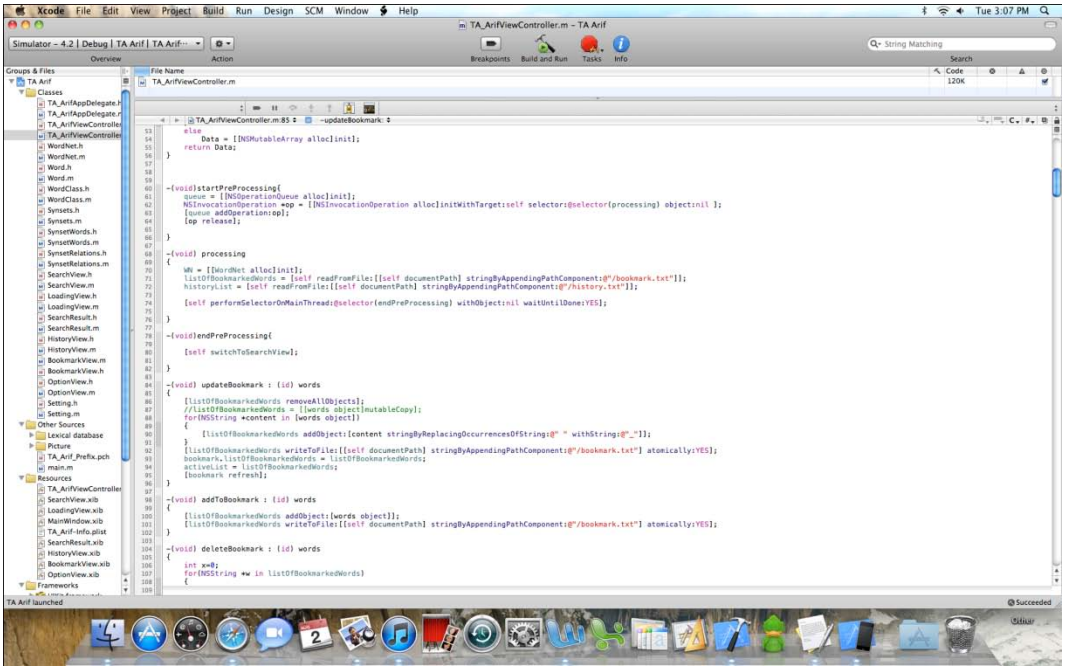
Pada baris 1 segmen program 2.5 ditunjukkan bahwa variabel x mendapatkan inisialisasi dari class NSString, hal ini akan menyebabkan Objective-C melakukan reservasi memory untuk variabel x. Agar variabel x tidak terpengaruh oleh garbage collector maka ditambahkan perintah retain seperti pada baris 2. Dengan adanya perintah retain ini maka saat pembersihan memory dilakukan variabel x tidak akan dihapus walaupun tidak digunakan. Ketika aplikasi sudah tidak lagi menggunakan variabel x maka variabel x dapat dibuang dari memory dengan menggunakan perintah release seperti pada baris 3. Variabel yang tidak digunakan tetapi tetap berada dalam memory disebut dengan memory leak. Ketika mencapai batas tertentu maka aplikasi dapat berhenti secara tiba-tiba tanpa adanya peringatan apapun. Untuk mengatasi hal ini penggunaan tool analyzer akan sangat membantu serta penggunaan perintah retain dan release secara berkala.

Kelebihan lain dari Objective-C adalah adanya fasilitas delegate. Fasilitas delegate memungkinkan class untuk menyerahkan method yang seharusnya ia jalankan sendiri, untuk dijalankan oleh class lain yang menjadi delegate. Fungsi ini banyak digunakan dalam pemrograman di iPhone, antara lain untuk mendapatkan lokasi sentuhan dilayar untuk pengguna.

2.3.2 Penulisan Program di Xcode

Xcode merupakan sebuah IDE yang dapat digunakan untuk membuat aplikasi pada MacOS dan iOS. Untuk membuat aplikasi pada kedua OS tersebut maka bahasa pemrograman yang harus digunakan adalah Objective-C tetapi bukan berarti Xcode tidak mendukung bahasa pemrograman lain. Xcode juga mendukung penggunaan bahasa pemrograman lain seperti C,C++,Java dan beberapa bahasa lainnya. Pada perkembangannya Xcode telah mencapai versi 4.0 dan masih dikembangkan hingga saat ini. Penulisan program pada Xcode memberi banyak kemudahan, diantaranya adapah pemberian warna yang berbeda pada setiap kata yang memiliki makna yang berbeda, sebuah nama variabel akan mendapat warna hijau sedangkan parameter akan berwarna pink. Selain itu pengetikkan suatu prosedur akan mendapat fasilitas auto complete dari Xcode,

dengan menekan tambol tab maka prosedur tersebut akan dengan sendirinya dituliskan secara lengkap oleh Xcode beserta dengan jenis parameter yang diperlukan oleh prosedur tersebut. Gambar 2.4 merupakan contoh tampilan dari Xcode.



Gambar 2.4
Tampilan Xcode

Dapat dilihat pada gambar 2.4 bahwa Xcode akan secara otomatis mengelompokkan file-file dengan tipe tertentu. File header dan implementation akan dikelompokkan ke dalam folder classes. Apabila programmer ingin mengelompokkan lagi file header dan implementation yang ada maka programmer juga dapat membuat folder lagi di dalam folder classes tersebut. File tampilan dan media akan dikelompokkan ke dalam folder other resources.

Xcode adalah sebuah aplikasi yang ditujukan hanya untuk menuliskan program bagi aplikasi Mac dan iOS, untuk mendesain tampilan maka digunakan aplikasi lain yang bernama interface builder. Interface builder dan Xcode terhubung melalui sebuah file dengan ekstensi *.xib yang dikenal dengan istilah file interface.

Untuk menjalankan aplikasi yang dibuat dapat dilakukan dengan memilih build and run atau dengan menekan command + return. Aplikasi yang dibuat dapat dijalankan pada simulator ataupun pada actual device seperti iPhone dan iPad. Apabila dijalankan memalui iPhone simulator maka dapat dilakukan debugging dengan mengaktifkan fitur breakpoints, fitur ini akan memberikan message apa bila terjadi error pada aplikasi yang sedang dijalankan. Sedangkan bila dijalankan melalui actual device maka fitur debugging atau yang lebih dikenal dengan istilah live debugging hanya bisa dilakukan apabila programmer terdaftar sebagai pengembang resmi di Apple dengan membayar $99 per tahun.

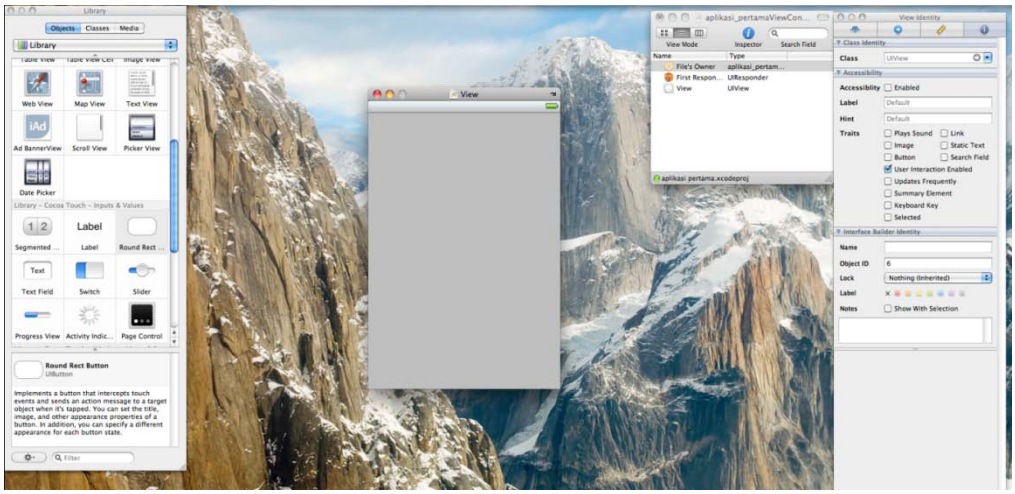
Selain fitur breakpoints Xcode juga menyediakan berbagai fitur lain seperti auto completion dan fitur analyzer. Fitur auto completion akan secara otomatis memberikan suggestion mengenai command yang akan ditulis dan melengkapi command tersebut apabila dipilih. Sedangkan fitur analyzer akan melakukan scanning pada line of code yang ada dan menunjukkan bagian-bagian yang memilki resiko terjadinya memory leak.

2.3.3 Interface Builder

Selain Xcode aplikasi lain yang sangat membantu dalam pembuatan program pada Mac dan iOS adalah Interface Builder. Interface Builder digunakan untuk mendesain tampilan dari aplikasi yang dibuat, aplikasi tersebut bisa ditujukan untuk Mac, iPhone / iTouch, maupun iPad. Interface builder akan menyediakan ukuran frame yang sesuai dengan target device yang dipilih.

Pada Gambar 2.5 ditunjukkan bahwa terdapat beberapa window pada interface builder. Window yang berada disebelah kiri adalah window library yang berisi komponen-komponen yang telah disediakan oleh interface builder. Untuk memasukkan komponen-komponen tersebut ke dalam view aplikasi cukup dengan melakukan drag and drop dari library ke dalam view aplikasi. Window yang berada ditengah layar adalah view dari file *.xib yang sedang aktif, sebuah file *.xib memiliki 1 buah view. Setelah Window view terdapat window controller yang mengandung File’s Owner dan First Responder. Sedangkan window yang

berada di paling kanan adalah window properties yang berisi properti-properti dari suatu komponen.



Gambar 2.5
Tampilan Interface Builder

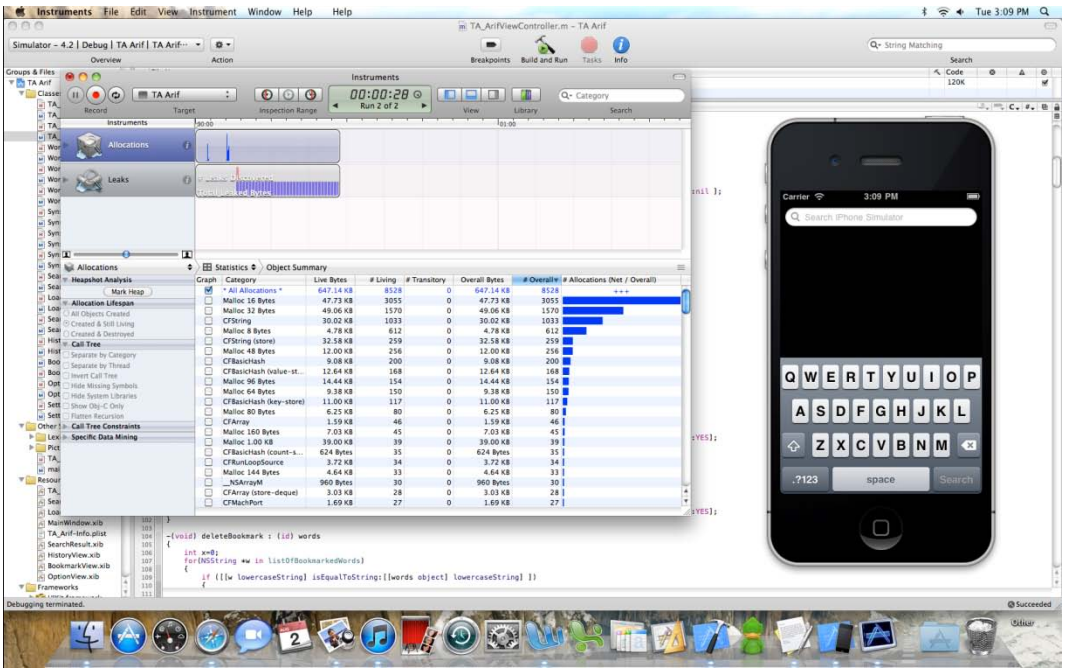
Suatu komponen dalam Interface Builder dapat dihubungkan dengan variabel bertipe IBOutlet yang dideklarasikan pada Xcode. Setiap komponen tersebut juga dapat diberi event yang didapatkan dari sebuah method yang dideklarasikan pada Xcode dengan tipe IBAction. Cara menghubungkan suatu komponen dengan Xcode adalah menarik garis dari komponen tersebut dengan controller yang bersangkutan. Penarikan garis dapat dilakukan dengan menahan tombol ctrl atau dengan menahan tombol mouse kanan dan melakukan drag dari komponen ke controller. Apabila garis ditarik dari controller File's Owner dan dihubungkan dengan sebuah komponen maka akan muncul daftar IBOutlet yang dapat dihubungkan dengan komponen tersebut, tetapi jika garis ditarik dari komponen dan dihubungkan ke controller File's Owner maka akan muncul daftar method yang bisa dilakukan. Sebuah komponen bisa memiliki lebih dari satu event, misalnya touch up inside, touch up outside,touch down inside,dl. Apabila sebuah action diberikan kepada komponen dengan cara menarik garis maka action tersebut akan masuk ke dalam event utama dari komponen tersebut.

Interface Builder juga menyediakan fasilitas untuk mendesain dalam portrait maupun landscape. Hal ini akan sangat berguna bagi aplikasi yang

mengijinkan terjadinya rotasi layar, apabila terjadi rotasi maka tidak perlu membuat view yang berbeda tetapi cukup dengan merubah posisi dari komponan-komponen yang ada pada view tersebut dan mengubah ukuran agar sesuai dengan ukuran layar yang baru.

2.3.4 Instruments

Salah satu tool yang disediakan oleh Xcode adalah tool Instruments, tool ini berguna untuk menganalisa kinerja dari aplikasi yang dibuat. Aplikasi ini dibuat berdasarkan DTrace tracing framework dari OpenSolaria dan sudah terintegrasi ke dalam Xcode. Gambar 2.6 menunjukkan tampilan Instruments.



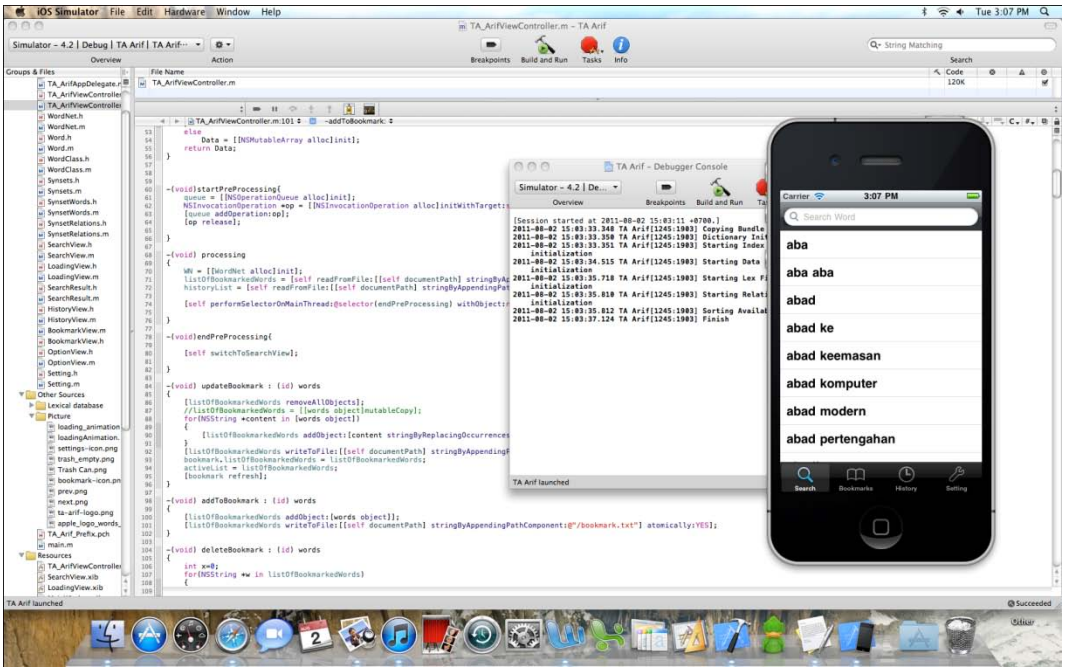
Gambar 2.6
Tampilan Instruments

Pada gambar 2.6 ditunjukkan bahwa instruments menampilkan timeline dari program yang disertai dengan grafik leaks. Grafik ini menunjukkan adanya memory leak pada aplikasi yang dibuat, semakin banyak leaks yang terjadi maka semakin tinggi grafik yang dihasilkan. Pada bagian bawah instruments akan ditunjukkan jumlah memory yang digunakan oleh class-class yang ada serta

penggunaan class yang menyebabkan terjadinya memory leak, sayangnya tidak ditunjukkan line of code yang menyebabkan terjadinya memory leak. Instruments sangat berguna untuk meningkatkan kualitas dari aplikasi yang dibuat agar lebih cepat dan efisien. Dalam pengembangan aplikasi penggunaan instruments sangat disarankan.

2.3.5 iPhone Simulator

iPhone Simulator merupakan bagian dari iPhone SDK yang terdapat didalam Xcode. Dengan adanya iPhone simulator maka programmer tidak perlu memiliki actual device untuk mencoba aplikasinya. Hampir semua fitur dari iPhone dapat dijalankan oleh simulator, misalnya penanganan touch dan multi touch pada layar, rotasi layar, akses internet dan berbagai fitur yang lain. Gambar 2.7 menunjukkan tampilan dari iPhone simulator.



Gambar 2.7
iPhone Simulator

Beberapa fitur yang tidak dapat dilakukan oleh simulator adalah kamera, accelerometer, GPS dan beberapa fasilitas lain yang membutuhkan hardware yang

tidak terdapat pada mac. Sehingga untuk uji coba fitur-fitur tersebut harus menggunakan iPhone dan memanfaatkan fitur live debugging. Tetapi dengan adanya simulator ini maka debug untuk fitur-fitur yang lain dapat dilakukan dengan memanfaatkan fitur breakpoints milik Xcode. Fitur breakpoint dapat melakukan trace terhadap jalannya program.

2.3.6 Arsitektur iPhone SDK

API pada iPhone memiliki banyak kesamaan dengan API milik Mac, keduanya terbagi menjadi 4 bagian. Yang membedakan kedua API tersebut adalah API lapisan teratas, apabila pada Mac lapisan teratasnya adalah Cocoa maka pada iPhone lapisan teratasnya adalah Cocoa Touch. Perbedaan ini disebabkan karena adanya perbedaan input pada kedua device, pada Mac input berupa pengetikkan pada keyboard sedangkan pada iPhone inputan berupa sentuhan pada layar device. Berikut ini adalah susunan arsitektur iPhone SDK:

- Cocoa Touch

  Cocoa Touch berisi fasilitas-fasilitas untuk berinteraksi dengan pengguna, seperti menerima inputan dan mengatur tampilan. Lapisan teratas dari arsitektur iPhone ini berisi: Multi-Touch events and controls, Accelerometer support, View hierarchy, Localization, dan Camera Support.

- Media

  Lapisan Media seperti namanya berisi beberapa kumpulan API dan framework untuk memutar dan menampilkan video, gambar dan suara. Isi dari lapisan Media adalah: Open AL, audio mixing and recording, Video Playback, Image file formats, Quartz, Core Animation, dan OpenGL ES.

- Core Services

  Lapisan Core Services berisi beberapa fasilitas-fasilitas dasar, dan sudah mulai berhubungan dengan hardware. Fasilitas-fasilitas yang ada dalam Core Services adalah: Networking, Embedded SQLite database, Core Location, Threads, dan Core Motion.

- OS X Kernel

  Pada dasarnya system operasi yang dijalankan di iPhone sama dengan yang dijalankan di Mac, yaitu OS X. Oleh karena itu lapisan paling bawah dari arsitektur iPhone SDK adalah OS X Kernel. Dalam lapisan ini, hampir semuanya berhubungan langsung dengan hardware. Isi dari lapisan ini adalah: TCP/IP, Sockets, Power Management, File System, dan Security.

2.4 WordNet

WordNet[1] adalah sebuah kamus yang didesain menyerupai sebuah thesaurus. Pada WordNet sebuah kata disimpan dalam bentuk sinonim set (synset). Synset adalah kata-kata yang memiliki bentuk yang berbeda tetapi memiliki arti kata yang sama. Tetapi berbeda dengan thesaurus, WordNet tidak hanya mengandung kumpulan kata-kata yang memiliki arti yang sama tetapi juga mengandung informasi lain berupa relasi dari kata-kata tersebut beserta glossnya.

2.4.1 Sejarah Pembentukan WordNet

Pada umumnya kamus dibuat dengan mengurutkan daftar kata yang ada secara alphabetical tanpa memperhatikan arti dari kata tersebut. Hal ini memberikan kemudahan bagi pembuat dan pengguna kamus. Tetapi sistem ini memiliki kekurangan yaitu pencarian kata secara alphabetical akan memakan waktu sehingga menyebabkan masyarakat malas mencari kata dalam suatu kamus.

Seiring dengan berkembangnya teknologi maka masalah pencarian kata yang memakan waktu telah dapat diatasi oleh komputer. Komputer dapat melakukan pencarian kata pada sebuah kamus elektronik dengan kecepatan yang sangat tinggi. Walaupun demikian sistem yang digunakan oleh kamus tersebut tetaplah sistem konvensional yang tidak memperhatikan arti dari kata tersebut. Para ahli bahasa akhirnya mengusulkan suatu sistem baru yang disebut WordNet

[1] Pangestu. S., *Pembuatan Prototype Database Lexical untuk Bahasa Indonesia yang mengacu pada Wordnet*. 2007

dimana sebuah kamus dikombinasikan dengan informasi lexical dan didukung oleh komputer berkecepatan tinggi. WordNet menyerupai sebuah kamus konvensional dengan memberikan informasi mengenai definisi besera contoh penggunaan kata yang ada pada synset. Tetapi kelebihan dari WordNet adalah kemampuannya menyajikan informasi lexical dengan memberikan relasi-relasi yang ada beserta sinonim yang dimiliki kata tersebut. Dapat dikatakan bahwa WordNet merupakan perkembangan dari kamus dan thesarurus dimana keduanya disatukan dan mendapat tambahan informasi berupa relasi kata.

2.4.2 Lexical Database

Lexical Database yang dimiliki oleh WordNet dituliskan dalam format ASCII. Hal ini dilakukan untuk memudahkan pengaksesan dalam berbagai bahasa pemrograman. File Lexical Database ini dapat dilihat dengan menggunakan text editor pada berbagai OS, misalnya TextEdit pada Mac dan WordPad pada Windows. Kelemahan dari sistem penyimpanan dalam format ascii ini adalah file Lexical Database akan sangat mudah diubah oleh orang awam.WordNet bahasa Indonesia yang digunakan pada Tugas Akhir ini mengandung 9 buah file dengan ukuran 11,4 MB. Tabel 2.2 menunjukkan detail Lexical Database Files.

Tabel 2.2
Daftar File Lexical Database

Nama File	Fungsi	Size (KB)
index.noun	Index file untuk kata noun	1,952
index.verb	Index file untuk kata verb	1,215
index.adj	Index file untuk kata adj	54
index.adv	Index file untuk kata adv	470
data.noun	Data file untuk kata noun	5,352
data.verb	Data file untuk kata verb	1,992
data.adj	Data file untuk kata adj	646
data.adv	Data file untuk kata adv	61
lexnames	Mencatat lexnames	1

Semua file Lexical Database yang ada pada tabel 2.2 ditulis dengan menggunakan format ASCII. Untuk pemisah setiap kata pada index file maupun synset pada data file digunakan karakter enter (newline). Sedangkan untuk pemisah antar field yang ada digunakan karakter spasi.

Pencarian sebuah kata akan melalui dua tahap proses yaitu: pencarian kata pada setiap index file dan ekstraksi synset data pada data file. Pencarian kata pada setiap index file dilakukan untuk mengatasi kemungkinan sebuah kata dapat berada pada lebih dari 1 class kata.

2.4.2.1 Index File

Index File berisi daftar semua kata pada WordNet sesuai dengan class kata masing-masing (noun,verb,adj,adv). Kata-kata tersebut disusun urut sesuai dengan alphabet dan ditulis dengan huruf kecil. Setiap kata yang ada dipisahkan oleh karakter enter (newline). Index File diawali dengan keterangan mengenai versi WordNet dan license agreement mengenai penggunaan WordNet tersebut, license agreement ini berjumlah 29 baris. Setelah license agreement tersebut selesai maka dimulailah baris-baris yang berisi data dari WordNet, format dari baris-baris tersebut adalah sebagai berikut:

```
lemma class synset_cnt p-cnt [ptr_symbol...] sense_cnt
tagsense_cnt synset_offset [synset_offset...]
```

Berikut ini adalah penjelasan dari format di file index:

- Lemma

  Kata yang ditulis dalam huruf kecil. Apabila kata ini terpisah oleh spasi maka spasi tersebut akan diganti dengan garis bawah (_), contohnya abad_ke, abah_abah dan sebagainya.

- Class

Kategori dari bentuk kata. Terdapat 4 kemungkinan untuk class ini yaitu: karakter “n” untuk class noun, karakter “v” untuk class verb, karakter “a” untuk class adjective dan class “r” untuk class adverb.

- Synset_cnt
  Merupakan jumlah dari synset yang dimiliki oleh lemma. Juga merupakan jumlah sense yang dimiliki.
- P_cnt
  Jumlah pointer yang dimiliki oleh sebuah lemma. Pointer yang dimaksud adalah pointer yang menunjukkan relasi-relasi yang dimiliki.
- Ptr_symbol
  Simbol-simbol pointer yang dimiliki oleh lemma. Setiap symbol pointer memiliki arti tersendiri, hal ini akan dijelaskan lebih detail pada subbab 2.4.2.3. Apabila lemma tidak memiliki relasi apapun maka p_cnt akan bernilai 0 dan field ini akan dihilangkan.
- Sense_ cnt
  Jumlah dari sense yang dimiliki oleh lemma.
- Tagsense_cnt
  Jumlah penggunaan lemma yang disorting berdasarkan frekuensi penggunaan pada teks *semantic concordance*.
- Synset_offset
  Byte offset dari synset yang dimiliki oleh lemma. Setiap synset_offset terhubung dengan synset yang berbeda data file. Synset offset terdiri dari 8 digit bilangan integer.

Berikut ini adalah contoh kata pada index file disertai dengan penjelasaan mengenai field-field yang dimiliki

```
abc n 3 2 #p %m 3 0 05321050 04785693 01743487
```

Field yang dimiliki:

- Lemma : abc
- Class : n (noun)
- Synset_cnt : 3
- P_cnt : 2
- Ptr_symbol : #p dan %m
  (sesuai dengan jumlah pada p_cnt)
- Sense_cnt : 3
- Tagsense_cnt : 0
- Synset_offset : 05321050, 04785693, 01743487
  (Sesuai dengan jumlah pada synset_cnt)

2.4.2.2 Data File

Data File berisi data-data yang dimiliki oleh sebuah synset. Dimulai dari sinonim, gloss, example dan relasi-relasi lexical yang dimiliki oleh synset tersebut. Pada awal setiap data file juga terkandung keterangan mengenai versi WordNet dan juga license agreement sejumlah 30 baris. Setelah itu barulah setiap baris mewakili satu buah synset. Format synset tersebut adalah sebagai berikut:

```
Synset_offset lex_filenum ss_type w_cnt word lex_id
[word lex_id...] p_cnt [ptr...] (frames..) | gloss
```

Berikut ini adalah penjelasan dari format di file data:

- Synset_offset

  Byte Offset dari synset yang berupa 8 digit bilangan integer.

- Lex_filenum

  Dua digit integer yang mencatat nomor file lexnames yang mengandung synset ini. Lexname akan dijelaskan pada subbab 2.4.2.4.

- Ss_type

  Berupa 1 karakter yang menunjukkan class kata. Memiliki nilai yang sama dengan field class pada index file.

- W_cnt

  Berupa 2 digit hexadesimal yang menunjukkan jumlah kata pada synset.

- Word

  Kata-kata yang berada didalam suatu synset dan ditulis dalam bentuk ASCII. Apabila kata tersebut terpisah oleh spasi maka akan digantikan dengan garis bawah (_). Word bersifat case-sensitive berbeda dengan lama pada index file yang ditulis dengan huruf kecil.

- Lex_id

  Berupa 1 digit hexadesimal integer yang jika ditambahkan pada lemma, akan memberikan karakteristik unik pada arti kata dalam lexical file. Biasanya berawal dari 0 dan bertambah sesuai dengan jumlah munculnya word yang ditambahkan pada file yang sama.

- P_cnt

  Berupa 3 digit bilangan integer yang menunjukkan jumlah pointer yang dimiliki oleh synset. Pointer yang dimaksud adalah pointer relasi. Apabila synset tidak memilki pointer apapun maka akan bernilai 000.

- Ptr

  Berfungsi menghubungkan suatu synset dengan synset lain dengan format sebagai berikut:

```
Ptr_symbol synset_offset class source/target
```

  Format tersebut memiliki kesamaan dengan format pada index file. Ptr_symbol menyatakan relasi yang dimiliki dengan synset_offset, class menunjuk class tempat synset_offset berada. Source/target berupa 4 digit bilangan hexadesimal, 2 digit pertama menunjukkan nomor word pada synset asal dan 2 digit terakhir menunjukkan nomor word pada synset target. Apabila Source/Target bernilai

0000 berarti kedua synset memiliki hubungan semantic. Tetapi apabila tidak bernilai 0000 maka kedua synset memiliki hubungan lexical. Nomor word berhubungan dengan field word pada synset, dari kiri ke kanan dimulai dari 1.

- Frames

  Frames hanya terdapat pada data.verb saja. Frames memiliki format:

  ```
  F_cnt + f_num w_num [+ f_num w_num.. ]
  ```

  F_Cnt menunjukkan jumlah dari frame yang terdaftar, F_cnt berupa 2 digit bilangan integer. F_num merupakan 2 digit bilangan desimal menunjukkan nomor frame yang digunakan. W_num berupa 2 digit bilangan hexadesimal yang menunjukkan word pada synset yang dijelaskan oleh field frame. Bila w_num bernilai 00 maka f_num mengacu untuk semua word pada synset.

- Gloss

  Gloss merupakan definisi dari sebuah synset dan dapat disertai oleh contoh kalimat. Gloss diawali dengan menggunakan sumbol "|".

Berikut ini dalah contoh data file dari data.noun:

```
05321050 03 n 0f abc 3 abece 0 abjad 2 aksara 2 alfabet 0
alif_ba_ta 0 alif_bata 0 fonem 0 hiroglif 0 huruf 4 ideograf
0 kritogram 0 lambang_bunyi 0 leter 2 piktograf 0 003 %m
03473223 n 0000 %m 05164323 n 0000 #p 05077790 n 0000 |
hurufhuruf yang dikembangkan oleh bangsa Inggris yang
didasarkan pada alfabet Romawi yang usianya kirakira 2.500
tahun
```

Data dari tiap field adalah sebagai berikut:

- Synset_offset : 05321050
- Lex_filenum : 03
- Ss_type : n (noun)
- W_cnt : 0f (15 desimal)
- Word : abc, abece, abjad, aksara, alphabet, alif_ba_ta, alif_bata, fonem, hiroglif, huruf, ideograf, kritogram, lambang_bunyi, leter, piktograf (15 word sesuai dengan w_cnt)
- Lex_id : 3, 0, 2, 2, 0, 0, 0, 0, 0, 4, 0, 0, 0, 2, 0
- P_cnt : 003

- Ptr
  - Ptr_symbol : %m, %m, %p
  - Synset_offset : 03473223, 05164323, 05077790
  - Class : n (noun), n (noun), n (noun)
  - Soucer/Target : 0000, 0000, 0000
- Frames : -
- Gloss : hurufhuruf yang dikembangkan oleh bangsa Inggris yang didasarkan pada alfabet Romawi yang usianya kirakira 2.500 tahun

Berikut ini adalah contoh data file dari data.verb:

```
00001740 29 v 01 mengerpus 0 000 01 + 01 00 | menghukum dengan cara memasukkan prajurit ke dalam kerpus menghukum dalam kurungan
```

Data dari tiap field adalah sebagai berikut:

- Synset_offset : 00001740
- Lex_filenum : 29

- Ss_type : v (verb)
- W_cnt : 01
- Word : mengerpus
  (1 word sesuai dengan w_cnt)
- Lex_id : 0
- P_cnt : 000
- Ptr : -
- Frames : 01 + 01 00
  - F_cnt : 01
  - F_num : 01
  - W_num : 00
- Gloss : menghukum dengan cara memasukkan prajurit ke dalam kerpus menghukum dalam kurungan

2.4.2.3 Pointer Symbol

WordNet menggunakan simbol-simbol untuk melambangkan hubungan lexical yang ada. Setiap relasi memilki simbol yang berbeda-beda, Tabel 2.3 menunjukkan relasi yang dimiliki oleh WordNet.

Tabel 2.3
Pointer Simbol pada WordNet

Relasi	Pointer	Bentuk kata
Antonim	!	Noun, Verb, Adjective
Hipernim	@	Noun, Verb
Instance Hipernim	@i	Noun
Hiponim	~	Noun, Verb
Instance Hiponim	~i	Noun
Member Holonim	#m	Noun
Substance Holonim	#s	Noun
Part Holonim	#p	Noun

Tabel 2.3
(Lanjutan)

Relasi	Pointer	Bentuk kata
Member Meronim	%m	Noun
Substance Meronim	%s	Noun
Part Meronim	%p	Noun
Atribut	=	Noun, Adjective
Derivationally Related Form	+	Noun
Domain of Synset - Topic	;c	Noun, Verb, Adjective
Member of this domain - Topic	-c	Noun
Domain of Synset - Region	;r	Noun, Verb, Adjective
Member of this domain - Region	-r	Noun
Domain of Synset - Usage	;u	Noun, Verb, Adjective
Member of this domain - Usage	-u	Noun
Entailment	*	Verb
Cause	>	Verb
Also See	^	Verb, Adjective
Verb Group	$	Verb
Similar To	&	Adjective
Participle of Verb	<	Adjective
Pertainim (Perains to Noun)	\	Adjective

Pada WordNet Bahasa Indonesia yang digunakan pada Tugas Akhir ini hanya mengandung antonim, hipernim, hiponim, holonim, dan meronim. Antonim adalah sebuah relasi yang dimiliki oleh sebuah kata yang merujuk pada kata yang memiliki arti yang berlawanan, contohnya kehidupan dan kematian. Hipernim adalah relasi yang merujuk pada kata yang sifat nya lebih umum sedangkan hiponim merupakan relasi yang merujuk pada kata yang sifatnya lebih spesifik. Contoh hipernim adalah : gurun pasir dan gurun sedangkan contoh hiponim adalah gurun dan gurun pasir, kedua relasi tersebut saling berhubungan satu sama lain.

Holonim adalah sebuah sebuah relasi dimana sebuah kata merupakan bagian yang lebih besar yang disusun oleh relasi meronim sedangkan meronim adalah sebuah relasi dimana sebuah kata merupakan bagian dari sesuatu. Contoh holonim: mata dan retina, mata merupakan sesuatu yang tersusun oleh retina. Contoh dari meronim adalah kebalikan dari holonim yaitu retina dan mata, retina merupakan bagian dari mata. Holonim dan meronim juga merupakan relasi yang tidak dapat dipisahkan.Relasi-relasi tersebut saling berhubungan satu sama lain dan diwakilkan dengan menggunakan simbol. Setiap simbol bersifat unik dah hanya mewakilkan satu buah relasi saja.

2.4.2.4 Lexnames

Lexnames merupakan hasil proses grinder terhadap lexicographer file. Lexnames ini mengandung informasi class dari kata diikuti dengan kelompok kata. Format dari lexnames adalah :

```
File_number Nama_File Kategori_Sintaksis
```

Field file_number berupa 2 digit integer yang merupakan nomor urut file dan disusun secara ascending. Angka yang hanya 1 digit harus menambahkan angka 0 didepannya. Nama_file merupakan field dengan tipe string yang memiliki struktur class kata diikuti dengan karakter “.” dan kategori kelompok yang dikelompokkan berdasarkan kategori tertentu. Gambar 2.8 menunjukkan isi dari file lexnames.

```
00      adj.all    3
01      adj.pert 3
02      adv. all 4
03      noun.Tops 1
04      noun.act   1
05      noun.animal      1
06      noun.artifact    1
07      noun.atribut     1
08      noun.body 1
09      noun.cognation   1
```

Gambar 2.8
10 baris pertama dari lexnames

Seperti pada gambar 2.8 nama_file berupa tipe class yang diikuti dengan karakter "." dan grup kata. Misalnya noun.animal berarti kata dengan class noun yang termasuk ke dalam class hewan. Field kategori_sintaksis akan diisi sesuai dengan tipe class kata, bila noun maka field akan bernilai 1, bila verb maka field akan bernilai 2, bila adjective maka field akan bernilai 3, bila adverb maka field akan bernilai 4.

2.5 Pembuatan WordNet Bahasa Indonesia

WordNet Bahasa Indonesia merupakan hasil penelitian dari para mahasiswa STTS. Para mahasiswa tersebut melakukan pembentukan synset, mendapatkan definisi dan membuat aplikasi untuk melakukan perubahan pada WordNet. Terdapat empat orang mahasiswa yang melakukan pembuatan WordNet bahasa Indonesia, yaitu: Andy Saputra, Erick Pranata, Heru Raharjo, Theofilus Christyawan. Pembuatan synset dilakukan dengan memanfaatkan resource yang berasal dari KBBI (Kamus Besar Bahasa Indonesia), Tesaurus dan kamus bilingual bahasa Indonesia dan Inggris. Pembuatan synset tersebut ditangani oleh Andy Saputra dan dalam penelitiannya terdapat beberapa kendala yang dihadapi ketika synset dibentuk dengan menggunakan resource monolingual :

1. Data dalam Tesaurus tidak lengkap dan tidak konsisten sehingga membutuhkan proses lebih lanjut dan asumsi yang ditentukan sendiri. Inkonsistensi data dari Tesaurus ini dapat terlihat pada tercampurnya sinonim dan hiponim pada pasangan dalam Tesaurus. Hal ini harus ditangani dengan konsep hubungan sinonim yang digunakan. Ketidaklengkapan Tesaurus dapat terlihat pada beberapa kata yang tidak memiliki entry di dalam Tesaurus, namun disebutkan pada pasangan dari kata lain sehingga tidak bisa dilakukan proses pemeriksaan hubungan sinonimnya.
2. Tidak kongruennya data dari KBBI dan Tesaurus sehingga mempengaruhi proses penggabungan data dari KBBI dan Tesaurus. Contoh dari tidak kongruennya data dari resource yang digunakan adalah misalkan dalam KBBI, lema 'abad' memiliki 4 sense yang

berbeda. Sedangkan dalam Tesaurus, lema ‘abad’ hanya memiliki 1 pasangan. Karena kurangnya informasi, maka tidak diketahui lema ‘abad’ yang dimaksud dalam Tesaurus ini merujuk pada sense yang mana dari lema ‘abad’ pada KBBI. Hal ini mengurangi akurasi dari proses penggabungan.

3. Proses ekstraksi resource yang kurang sempurna menyebabkan beberapa kata yang tidak dapat diproses dengan akurat. Misalkan saja, terdapat kata ‘panji-panji’, ‘panjipanji’ dan ‘panjipan-ji’ yang akan dianggap kata yang berbeda oleh program, sementara sebenarnya ketiga contoh tersebut merujuk pada satu kata yang sama.

Sedangkan pembentukan synset dengan memanfaatkan resource bilingual dengan menggunakan word sense disambiguation pun memiliki kendala tersendiri. Beberapa kendala tersebut antara lain adalah:

1. Keterbatasan kualitas dan ketersediaan kamus yang memadai, sehingga hasil terjemahan dari Bahasa Indonesia menjadi Bahasa Inggris dan sebaliknya kurang baik. Tidak semua terjemahan dari kamus yang digunakan tersusun dalam bentuk kata, namun terdapat juga terjemahan dalam bentuk definisi yang memerlukan proses parsing. Hal ini cukup rentan menimbulkan kesalahan penerjemahan, terutama untuk penerjemahan kata Bahasa Indonesia menjadi Bahasa Inggris.
2. Terdapat perbedaan kelas kata antara kata dalam Bahasa Indonesia (dalam KBBI) dan Bahasa Inggris (dalam kamus) sehingga ada beberapa kata yang tidak ditemukan terjemahannya (karena proses penerjemahan dilakukan hanya terhadap kata yang kelas katanya sama saja). Contohnya adalah kata ‘hamil’ yang memiliki kelas kata ‘verb’ di KBBI, sedangkan terjemahannya, ‘pregnant’ memiliki kelas kata ‘adjective’. Hal ini mengakibatkan ‘pregnant’ tidak dianggap terjemahan yang valid untuk ‘hamil’.

Untuk definisi suatu kata ditangani oleh Theofilus Christyawan. Definisi dari suatu kata didapatkan dari melakukan crawling / fetching web-web site berbahasa Indonesia yang dihasilkan oleh google. Kata yang ingin dicari akan

dimasukkan ke dalam google search dan hasil web site yang didapatkan akan diambil sebagai sumber data. Hasil yang didapatkan meliputi dari huruf A hingga Z. Resource tersebut kemudian akan diproses untuk didapatkan definisi dari kata-kata yang ada dan dimasukkan ke dalam sense yang bersangkutan.

Erick Pranata menangani relasi hipernim dan hiponim, sedangkan Heru Raharjo menangani relasi holonim dan meronim. Keduanya menggunakan KBBI dan wikipedia sebagai sumber datanya. Hasil penelitian keempat orang tersebut menjadi dasar terbentuknya WordNet bahasa Indonesia. Penelitian tersebut kemudian diteruskan oleh Jessica Felani Wijoyo dengan membuat editor untuk WordNet.

2.6 Levenshtein Distance

Levenshtein Distance atau yang lebih dikenal dengan istilah edit distance adalah sebuah algoritma yang digunakan untuk menghitung perbedaan antara dua buah sequence. Pada Tugas Akhir ini levenshtein distance digunakan untuk menghitung biaya yang digunakan untuk merubah sebuah kata ke kata yang lain dengan cara menambahkan atau mengurangi huruf dan menukar posisi huruf yang ada pada sebuah kata. Hasil perhitungan dari levenshtein distance akan digunakan untuk menangani kemungkinan terhadinya kesalahan pengetikan oleh pengguna dan memberikan daftar kata yang mungkin ingin diketik oleh pengguna.

Setiap terjadi penambahan atau pengurangan huruf dan perubahan posisi huruf pada suatu kata maka levenshtein akan menambahkan biaya sebesar 1. Algoritma ini akan mencoba semua kemungkinan yang dapat dilakukan untuk merubah suatu kata menjadi kata yang lain dan mengambil biaya terkecil yang bisa didapatkan.

BAB III
ANALISA DAN DESAIN SISTEM PADA WORDNET BROWSER

Pada bab ini akan dijelaskan mengenai analisa sistem dari yang dibuat pada Tugas Akhir ini. Analisa dan desain sistem akan menjelaskan mengenai WordNet Browser secara umum, perbandingan Tugas Akhir dengan aplikasi lain yang sejenis, analisa permasalahan beserta solusinya dan software yang digunakan. Perbandingan dengan aplikasi sejenis akan meliputi perbandingan dengan aplikasi pada iPhone / iPad serta aplikasi pada desktop. Selain itu akan dijelaskan juga mengenai desain interface yang dimiliki oleh WordNet Browser beserta dengan kegunaan dari masing-masing halaman tersebut.

3.1 Gambaran Umum WordNet Browser

WordNet sebagai suatu sistem lexical database dapat diakses dengan dua cara, yaitu melalui browser dan melalui library. Pengaksesan melalui browser lebih ditujukan kepada end-user yang menggunakan WordNet sebagai kamus maupun lexical resource lainnya, sedangkan pengaksesan melalui library lebih ditujukan kepada developer maupun peneliti yang membutuhkan database WordNet untuk aplikasi ataupun penelitian yang mereka kerjakan.

Pada Tugas Akhir ini dikembangkan sebuah library WordNet menggunakan Objective-C yang dapat mengekstrak informasi-informasi yang terkandung dalam sebuah lexical database WordNet. Library ini diharapkan dapat membantu pengembang aplikasi berbasis Objective-C yang memerlukan informasi-informasi yang terkandung dalam WordNet. Tugas Akhir ini juga mengembangkan sebuah WordNet Browser yang dapat dijalankan pada sistem operasi iOS. Pada perkembangannya diharapkan browser ini dapat mengekstrak informasi dari lexical database bahasa lain selain bahasa Indonesia, karena itu digunakan bahasa Inggris sebagai bahasa universal pada interface browser.

Sebuah WordNet browser mampu menyediakan berbagai informasi mengenai sebuah kata, dimulai dari definisi kata hingga relasi-relasi yang dimiliki

oleh kata tersebut. Tetapi karena WordNet memiliki daftar kata yang sangat banyak maka tidak mungkin digunakan sistem konvensional dimana user harus mencari kata yang diinginkan dengan membaca kata-kata yang ada secara alphabetical. WordNet browser menyediakan fitur pencarian dimana user dapat langsung mengetikkan kata yang ingin dicari, dalam proses pengetikan tersebut browser akan menampilkan daftar kata-kata yang mungkin merupakan kata yang ingin diketikkan oleh user.

Selain memiliki fitur pencarian kata, browser juga akan dapat menangani kesalahan pengetikan kata oleh user. Apabila user mencari sebuah kata yang memiliki kesalahan pengetikkan maka browser akan secara otomatis mencari kata yang memiliki tingkat kemiripan paling tinggi dengan kata yang diketikkan oleh user dan menampilkannya ke layar. Pada layar hasil pencarian kata user dapat melakukan pencarian kata secara langsung dengan memilih kata tersebut. Tetapi hal ini hanya berlaku pada kata-kata tersebut saja, yaitu pada sinonim atau kata yang memiliki relasi dengan kata yang sedang dicari. Kata yang bisa dipilih memiliki warna yang berbeda dengan kata yang tidak bisa dipilih.

Secara default browser akan menampilkan informasi mengenai kata yang dicari user dengan menggunakan struktur kamus secara umum, yaitu gloss atau definisi dari suatu kata terlebih dahulu diikuti dengan sinonim dan relasi yang dimiliki. Apabila user sudah terbiasa dengan struktur tampilan dari Princeton WordNet (PWN) maka browser juga menyediakan fitur untuk merubah struktur tampilan sesuai dengan PWN. Apabila sebuah kata berada pada lebih dari satu class kata maka informasi dari kata tersebut akan dikelompokkan berdasarkan class katanya. Setiap class akan akan menempati satu buah layar iPhone, jika user ingin melihat class kata yang lain maka dapat dilakukan dengan melakukan sliding gesture. Fitur ini dapat dinonaktifkan sehingga semua class kata akan berada pada sebuah layar iPhone saja.

Browser memiliki fitur bookmark. User yang merasa memiliki ketertarikan dengan suatu kata dapat melakukan bookmark pada kata tersebut. Pada halaman bookmark yang dimiliki oleh browser user dapat melihat informasi mengenai kata yang sudah dibookmark maupun melakukan perubahan posisi bookmark serta

menghapus kata yang dirasa sudah tidak diperlukan. Selain fitur bookmark browser juga memiliki fitur history. Semua kata yang pernah dicari oleh user akan dicatat oleh browser. User dapat melihat semua list kata yang pernah dicari dan melihat informasi mengenai kata tersebut. Tombol back dan next juga tersedia pada history dan bookmark, kedua tombol ini akan sangat berguna jika user ingin melihat kata secara urut sesuai dengan daftar kata pada bookmark atau history. Pecatatan history secara default sejumlah 50 kata apabila suda melebihi batas tersebut maka kata yang paling lama akan dihapuskan dari history. Jumlah kata yang dicatat pada history dapat diatur pada halaman option dengan maksimal 100 kata dan minimal 10 kata. Apabila user tidak menginginkan pencatatan history maka fitur history juga dapat dinonaktifkan.

Apabila user ingin melakukan perubahan pada fitur-fitur yang disediakan oleh browser, tersedia halaman option. Pada halaman ini user dapat menonaktifkan fitur yang tidak diinginkan, misalnya user ingin menonaktifkan fitur pengelompokkan berdasarkan class kata.

3.2 Tinjauan terhadap Aplikasi Sejenis

Pada subbab ini akan dibahas perbandingan WordNet browser yang dibuat pada Tugas Akhir ini dengan aplikasi lain yang sejenis. Kedua aplikasi tersebut adalah aplikasi Dictionary.com dan Advanced English Dictionary and Thesaurus. Kedua aplikasi terbust memiliki beberapa fitur yang sama seperti auto complete text, group by class dan history (pada Dictionary.com fitur ini bernama recent). Tetapi kedua aplikasi tersebut juga memiliki beberapa perbedaan, misalnya : Dictionary.com memiliki fitur audio pronouncation tetapi tidak mengandung relasi hipernim dan hiponim. Sedangkan Advanced English Dictionary and Thesaurus memiliki kedua relasi tersebut tetapi tidak memiliki fitur audio pronouncation. Selain membandingkan dengan aplikasi mobile akan juga dibandingkan dengan aplikasi WordNet pada desktop dan Web milik Princeton.

Aplikasi WordNet browser yang dibuat pada Tugas Akhir ini mencakup hampir seluruh fitur dari aplikasi-aplikasi tersebut, fitur yang tidak dimiliki oleh Tugas Akhir ini adalah fitur audio pronouncation. Untuk fitur-fitur lain yang

dimiliki oleh kedua aplikasi-aplikasi tersebut seperti auto complete text dan history akan dibuat pada Tugas Akhir ini beserta dengan tembahan fitur-fitur yang tidak terdapat aplikasi lain. Pada tabel 3.1 ditampilkan fitur-fitur yang dimiliki dan tidak dimiliki oleh Tugas Akhir ini dan aplikasi-aplikasi yang telah disebutkan.

Tabel 3.1
Perbandingan Fitur dengan Aplikasi Sejenis

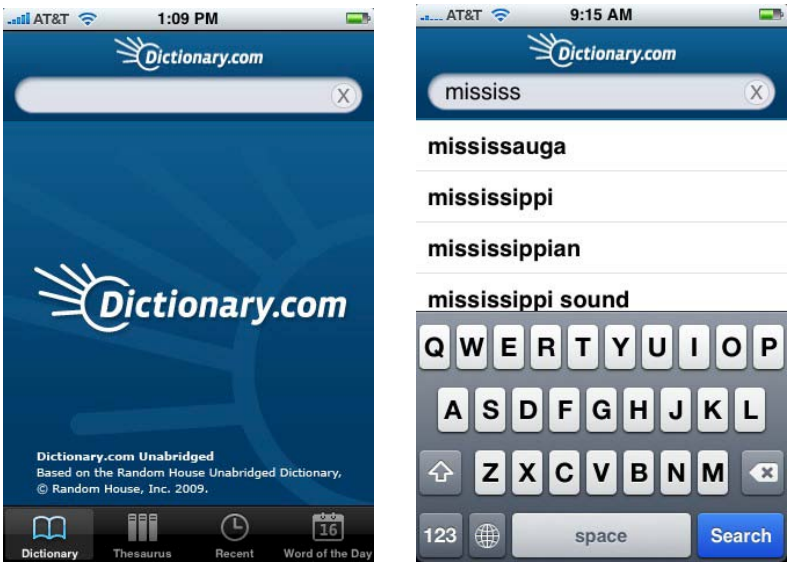
Pembanding	Dictionary.com	Advanced English Dictionary and Thesaurus	Tugas Akhir	Princeton WordNet
Show/hide gloss	×	×	✓	✓
Show only Class	×	×	✓	✓
Group by Class	✓	✓	✓	✓
Auto Complete Text	✓	✓	✓	×
History	✓	✓	✓	×
Bookmark	×	✓	✓	×
audio pronouncation	✓	×	×	×
Synset	✓	×	✓	✓
Hipernim	×	✓	✓	✓
Hiponim	×	✓	✓	✓
Holonim	×	×	✓	✓
Meronim	×	×	✓	✓
Suggestion	✓	×	✓	×

Pada subbab 3.2.1 akan dibahas mengenai fitur-fitur yang dimiliki oleh Dictionary.com dan Advanced English Dictionary and Thesaurus.

3.2.1 Dictionary.com

Dictionary.com adalah sebuah aplikasi gratis pada iPhone yang dikembangkan oleh Ask.com. Dictionary.com merupakan aplikasi kamus terbaik pada tahun 2009, aplikasi ini menyediakan gloss, sense, audio pronouncation serta synset dari kata yand dicari. Sampai saat ini Dictionary.com masih terus dikembangkan. Tampilan awal dari Dictionary.com sangat sederhana, hanya berupa sebuah textbox dan pada bagian bawah terdapat logo dari Dictionary.com seperti yang ditunjukkan gambar 3.1(a).

Pada saat user mulai mengetikkan kata yang ingin dicari maka aplikasi tersebut akan menampilkan kata-kata yang mungkin akan diketikkan oleh user. Fitur ini ditunjukkan pada gambar 3.1(b) dimana user mendapatkan sejumlah sugesti untuk kata ‘mississ’. User dapat mencari kata tersebut dengan menekan tombol “search” yang terdapat pada keyboard atau dengan melakukan tap pada daftar kata yang disediakan.

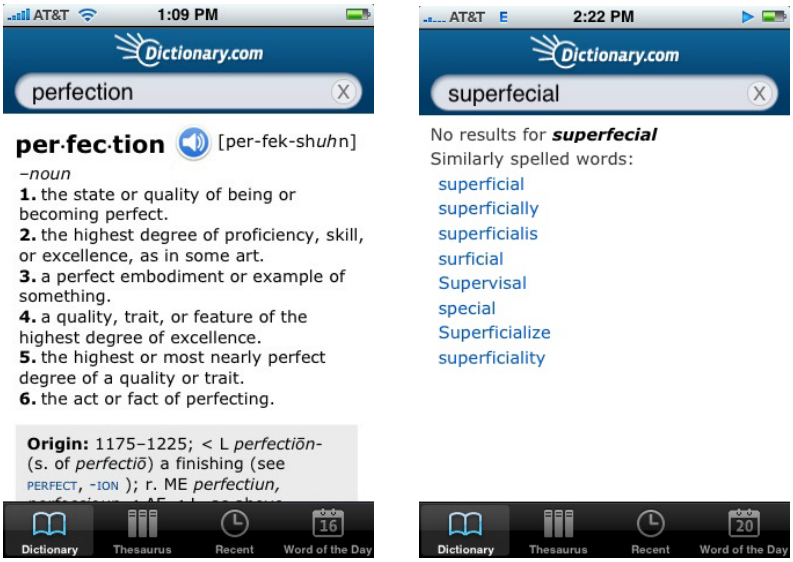


(a) **(b)**

Gambar 3.1
Tampilan Dictionary.com
(a). Tampilan Awal
(b). Fitur Auto Complete Text

Apabila kata yang dicari terdapat pada Lexical Database maka seluruh informasi seperti gloss, sense, class dan audio pronouncation akan ditampilkan. Apabila kata yang dicari berada pada lebih dari satu class kata maka informasi akan dikelompokkan berdasarkan class kata. Dictionary.com juga menyediakan informasi tambahan mengenai kata tersebut yang terletak pada bagian paling bawah. Pada gambar 3.2(a) ditunjukkan hasil pencarian kata ‘perfection’ , terdapat sebuah icon speaker yang akan menyuarakan kata tersebut apa bila ditekan oleh user. Tetapi apabila kata tersebut tidak terdapat pada Lexical Database maka aplikasi akan memberikan sejumlah kata yang mungkin ingin dicari oleh user. Hal

ini untuk mewaspadai apabila terjadi kesalahan pengetikan oleh user. Fitur tersebut ditunjukkan pada gambar 3.2(b). Untuk memanfaatkan fitur suggestion milik Dictionary.com maka iPhone harus memiliki koneksi internet, apa bila tidak tersedia koneksi internet maka fitur ini akan dinonaktifkan. Apabila suatu kata tidak ditemukan maka Dictionary.com akan meminta user mengaktifkan koneksi internet dan Dictionary.com tidak akan menampilkan kata apapun.



(a) **(b)**

Gambar 3.2
Hasil Pencarian Dictionary.com
(a). Tampilan Informasi Kata
(b). Tampilan Sugesti Kata

Apabila user ingin mengetahui mengenai sinonim dari kata yang dicari. Maka user dapat memilih tab Thesaurus yang terdapat pada bagian bawah aplikasi. Tampilan dari thesaurus ini ditunjukkan pada gambar 3.3. Dictionary.com mengikuti layout dari kamus conventional dimana informasi utama yang ditampilkan dari suatu kata adalah gloss dan examplenya. Sedangkan untuk informasi mengenai sinonim dari kata tersebut maka user harus mengakses halaman yang berbeda. Selain itu relasi-relasi yang dimiliki oleh kata juga tidak dimiliki oleh Dictionary.com. Dapat disimpulkan bahwa aplikasi Dictionary.com

tidak ditujukan untuk user yang sudah terbiasa menggunakan WordNet pada desktop application maupun pada web application.

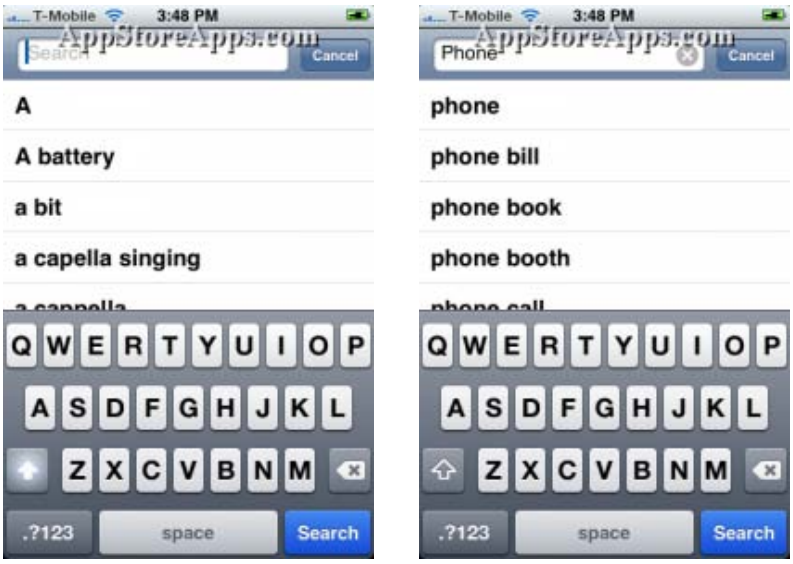
Gambar 3.3
Thesaurus Dictionary.com

Hingga saat ini Dictionary.com trs mengalami perkembangan. Dengan kualitas data yang cukup lengkap dan tampilan yang baik, Dictionary.com menjadi salah satu aplikasi kamus terfavorit di iPhone.

3.2.2 Advanced English Dictionary and Thesaurus

Advanced English Dictionary and Thesaurus merupakan contoh lain dari WordNet Browser pada iPhone. Aplikasi ini memiliki fitur-fitur yang hampir sama dengan Dictionary.com. Salah satu perbedaan utama dengan Dictionary.com adalah pada kelengkapan informasi yang ditampilkan. Pada Dictionary.com informasi yang disajikan hanyalah berupa gloss dan sinonim, tetapi Advanced English Dictionary and Thesaurus menyediakan informasi mengenai relasi dari kata tersebut, tetapi relasi yang disajikan hanyalah berupa relasi hipernim dan hiponim. Perbedaan yang kedua adalah pada layoutnya, apabila Dictionary.com menyajikan layout kamus conventional maka Advanced English Dictionary and Thesaurus menyediakan layout mengikuti layout milik Princeton WordNet.

Dimana pada layout ini sinonim dari kata ditampilkan terlebih dahulu baru diikuti dengan gloss dari kata tersebut. Setelah itu relasi dari kata akan ditampilkan pada bagian akhir.

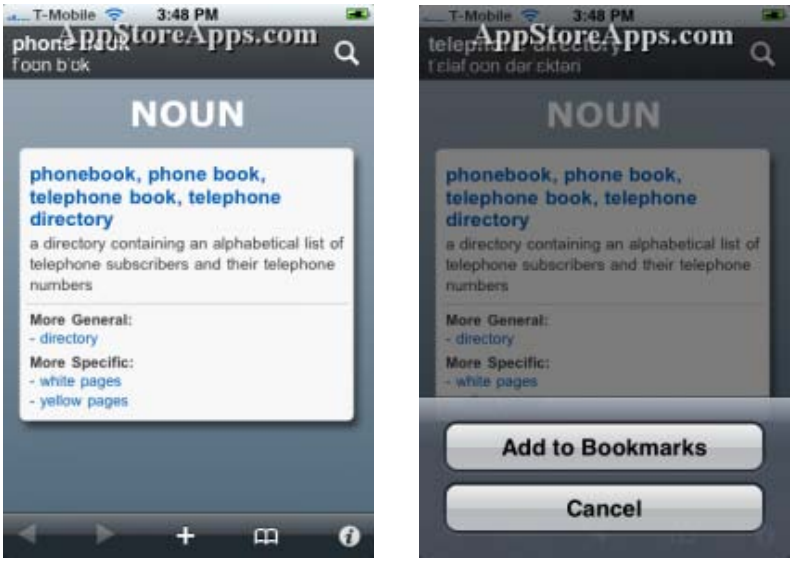


(a) **(b)**

Gambar 3.4
Tampilan Pencarian Advanced English Dictionary and Thesaurus
(a). Tampilan Awal
(b). Tampilan Fitur Auto Complete Text

Tampilan awal dari Advanced English Dictionary and Thesaurus akan langsung mengarahkan user pada halaman pencarian seperti pada gambar 3.4(a). Pada awal pencarian daftar semua kata akan ditampilkan pada user, daftar kata ini ditampilkan sesuai dengan urutan abjad sama seperti Dictionary.com. Ketika user mulai mengetikkan kata maka fitur auto complete akan mulai dijalankan, daftar kata akan menampilkan kata-kata yang memiliki kemiripan dengan yang diketikkan oleh user. Fitur ini ditunjukkan oleh gambar 3.4(b). Pencarian kata dapat dilakukan dengan menekan tombol search yang ada pada keyboard atau dengam melakukan tap pada kata yang ditampilkan oleh aplikasi. Setiap pencarian kata akan dicatat oleh aplikasi. Kata-kata yang pernah dicari tersebut dapat dilihat oleh user pada halaman history. Selain itu user juga dapat melakukan bookmark apabila suatu kata dianggap penting. Kata-kata yang dibookmark akan

ditampilkan pada halaman bookmark. Fitur bookmark dan history juga menyediakan fasilitas penghapusan kata yang tercatat.



(a) **(b)**

Gambar 3.5
Tampilan Informasi Kata dan Penambahan Bookmark
(a). Hasil Pencarian Kata
(b). Menambahkan Bookmark

Hasil dari pencarian kata pada Lexical Database seperti yang ditunjukkan pada gambar 3.5(a) menunjukkan class dari kata tersebut disertai dengan gloss dan relasi semantiknya, sinonim, hipernim dan hiponim. Bila user merasa perlu untuk melakukan bookmark terhadap kata tersebut maka dapat menekan tombol plus pada bagian bawah aplikasi. Penekanan tombol tersebut akan menampilkan konfirmasi pada user seperti pada gambar 3.5(b).

Pada halaman bookmark akan ditampilkan kata-kata yang telah dibookmark oleh user. Apabila kata tersebut diklik oleh user, maka class, gloss, sinonim, hipernim dan hiponimnya akan ditampilkan. Gambar 3.6(a) menunjukkan halaman bookmark. Pada Gambar 3.6(b) ditampilkan halaman history kata-kata yang pernah dicari oleh user. Sama seperti halaman bookmark, apabila kata tersebut diklik maka class, gloss, sinonim, hipernim dan hiponimnya akan ditampilkan.

(a) **(b)**

Gambar 3.6
Tampilan Halaman Bookmark dan Halaman History
(a). Halaman Bookmark
(b). Halaman History

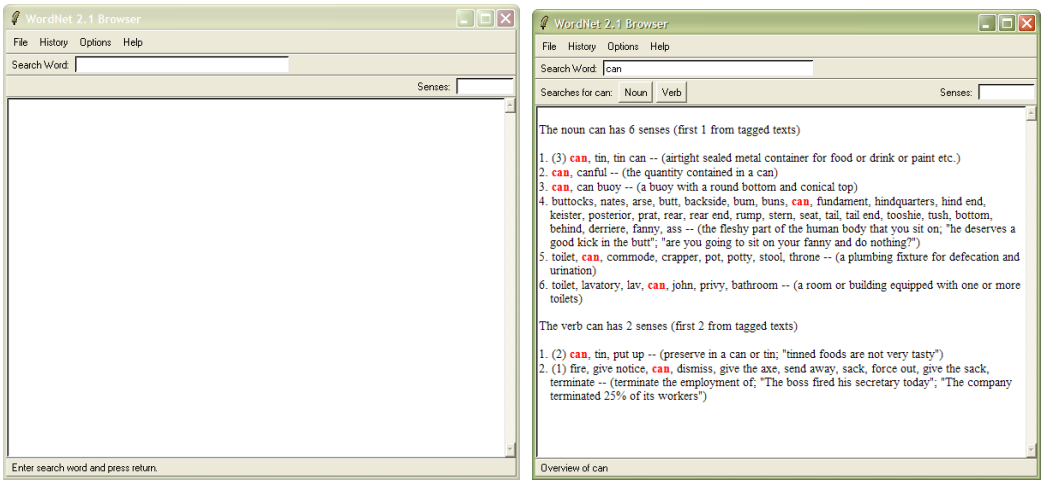
Daftar kata yang tercatat pada halaman history dapat dihilangkan dengan menekan tombol clear yang telah disediakan. Sedangkan pada halaman bookmark kata yang tercatat dapat dihapus satu-persatu dengan menekan tombol edit, setelah tombol edit ditekan maka akan muncul tombol delete di sebelah kiri setiap kata. Apabila tombol tersebut ditekan maka kata pemilik tombol delete tersebut akan dihapus.

3.2.3 Princeton WordNet

Princeton WordNet (PWN) merupakan lexical database pertama dalam Bahasa Inggris yang mendapat pengakuan luas atas peran pentingnya dalam perkembangan ilmu linguistik, termasuk juga ilmu komputer linguistik. PWN menyediakan 2 aplikasi yang dapat digunakan untuk mengakses lexical database. Aplikasi pertama berupa sebuah aplikasi desktop dan yang kedua berupa sebuah aplikasi web. Kedua aplikasi tersebut merupakan sebuah WordNet Browser yang mampu menampilkan berbagai informasi yang dimiliki oleh sebuah kata termasuk definisi dan relasi yang dimiliki oleh kata.

3.2.3.1 PWN Desktop Aplication

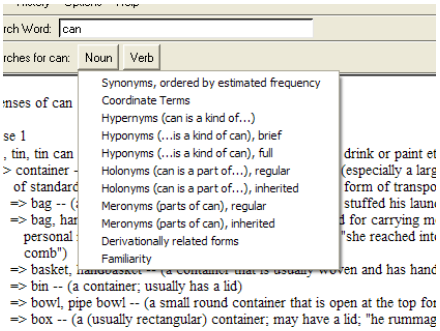
Browser ini ditujukan untuk end-user yang hanya menggunakan WordNet sebagai referensi saja, bukan untuk digunakan dalam program yang dikembangkan sendiri. Hingga saat ini, browser WordNet yang tersedia adalah browser versi 2.1 (untuk Windows) yang dapat didownload oleh user dari halaman web berikut http://WordNet.princeton.edu/WordNet/download/ seperti yang ditunjukkan pada gambar 3.7(a) di bawah ini.



(a) **(b)**

Gambar 3.7
Tampilan Desktop Application PWN Browser
(a). Tampilan Awal PWN Browser
(b). Tampilan Hasil Pencarian

Gambar 3.7(b) di atas menunjukkan proses selanjutnya, yaitu jika user sudah memasukkan kata yang ingin dicari synsetnya, maka browser akan memproses kata tersebut dan mengeluarkan hasil ekstraksi yang disusun berdasarkan class katanya (jika kata tersebut memiliki lebih dari 1 class kata). Kata yang dicari akan ditandai dengan diberi warna merah pada browsernya. Data yang ditampilkan oleh browser secara default adalah daftar synset yang mengandung kata tersebut beserta gloss (penjelasan tentang sense dari suatu synset).



Gambar 3.8
Memilih Relasi Semantik yang Lain

Jika user ingin melihat relasi lainnya, maka user bisa menekan tombol Noun/Verb (sesuai dengan kelas kata yang dikehendaki) pada browser, maka akan keluar pilihan seperti yang ditunjukkan pada Gambar 3.8. Relasi lainnya tersebut antara lain adalah hubungan hipernim/hiponim, hubungan holonim/meronim, bentuk turunan derivasi dari kata tersebut, dan lain-lainnya.

3.2.3.2 PWN Web Application

Selain versi offline yang dapat didonwload, Princeton juga menyediakan halaman web yang dapat digunakan untuk melihat synset dalam PWN tersebut (untuk versi website, tidak disediakan fasilitas untuk melihat berbagai hubungan semantik lainnya). Berikut adalah link yang dapat digunakan untuk mengakses PWN via website: http://WordNetweb.princeton.edu/perl/webwn. Tampilan web application dan desktop application miliki Princeton tidak jauh berbeda. Keduanya Hanya mediakan sebuah textbox tempat user mengetikkan kata yang ingin dicari dan opsi mengenai informasi yang ingin ditampilkan. Tetapi pada web application pemilihan opsi menggunakan sebuah combobox yang berisi opsi-opsi yang tersedia dan juga disediakan sebuah button “change” yang berfungsi mengaktifkan opsi yang telah dipilih.

Tampilan awal aplikasi browser WordNet via website dapat dilihat pada Gambar 3.9(a) di bawah ini. User dapat memasukkan kata yang ingin dicari pada bagian ‘Word to Search for’ dan dapat memilih Display Options untuk mengatur tampilan yang diinginkan.

(a) **(b)**

Gambar 3.9
Tampilan Web Application PWN Browser
(a). Tampilan Awal Browser
(b). Hasil Pencarian oleh Browser

Jika user sudah memasukkan kata yang ingin dicari, maka website akan menampilkan data yang sesuai seperti ditunjukkan pada Gambar 3.9(b). Data hasil pencarian akan ditampilkan sesuai dengan kelas katanya. Sebagai contoh untuk kata ‘can’, terdapat dua kelas kata yang berbeda, yaitu Noun dan Verb. Maka synset untuk Noun akan dipisahkan dengan synset untuk verb.

3.3 Arsitektur Sistem WordNet Browser

Dalam bab ini akan dijelaskan mengenai arsitektur sistem dari aplikasi WordNet Browser. Arsitektur yang dibahas mengenai input dan output serta arsitektur sistem dari aplikasi WordNet Browser yang dibuat pada Tugas Akhir ini.

3.3.1 Input Sistem

Pada WordNet browser terdapat 3 jenis input, input yang pertama berupa Lexical Database Files, input yang kedua adalah sebuah kata yang diketikkan atau dipilih oleh user dan input yang terakhir adalah pengaturan milik user. Lexical Database Files adalah input yang menyediakan informasi berupa kata-kata yang dimiliki oleh sebuah bahasa (dalam Tugas Akhir ini digunakan Bahasa Indonesia)

disertai dengan definisi, contoh useran kata, dan relasi-relasi semantik yang dimiliki oleh kata. Untuk input yang kedua yaitu kata, dapat diketikkan oleh user melalui halaman search. Pada halaman bookmark dan history user cukup memilih kata yang diinginkan dan browser akan menganggap kata tersebut sebagai input. Ketika informasi mengenai kata yang dicari oleh user ditampilkan, akan terdapat sejumlah kata yang dapat dipilih oleh user misalnya sinonim dari kata tersebut. Kata-kata tersebut juga akan menjadi input bagi browser. Input yang terakhir yaitu berupa pengaturan milik user berasal dari sebuah file setting. Pengaturan ini berupa penanda apakah fitur-fitur yang dimiliki oleh browser akan diaktifkan atau tidak, misalnya apakah browser akan mencatat kata-kata yang pernah dicari oleh user.

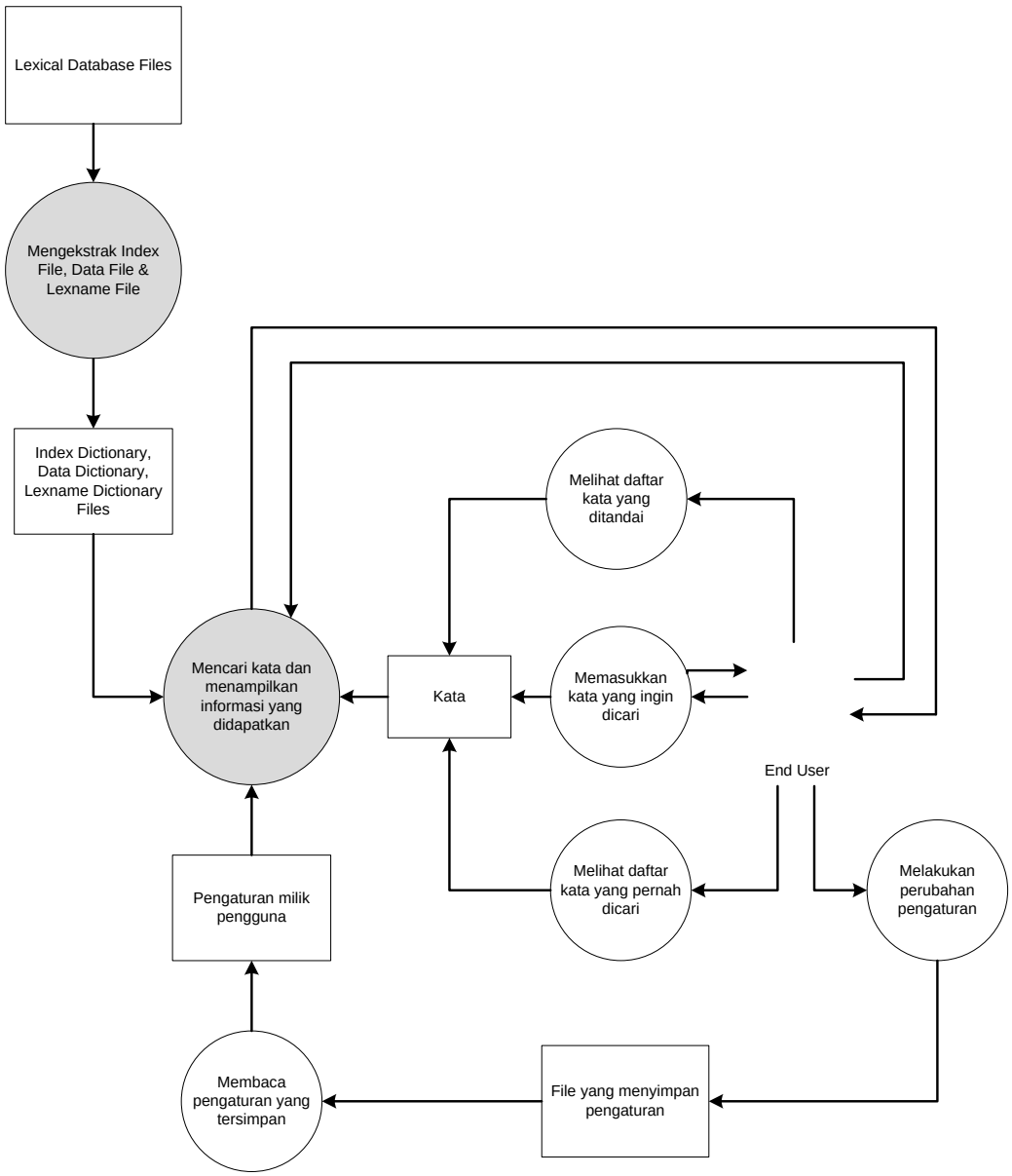
3.3.2 Output Sistem

Setiap input yang telah dijelaskan pada bagian input sistem akan memiliki jenis output yang berbeda-beda. Input pertama yaitu Lexical Database Files akan mengalami proses ekstraksi dimana setiap atribut yang tersimpan pada index files, data files, lexnames file akan diambil dan disimpan ke dalam sebuah dictionary. Output dari proses ini adalah dictionary yang mengandung semua informasi lexical.

Input yang kedua yaitu kata yang dicari akan mengalami proses pencarian dan ekstraksi informasi. Kata yang dicari tersebut akan dicari pada index dictionary, apabila kata tersebut ditemukan maka informasi mengenai kata tersebut akan diambil dari data dictionary jika tidak ditemukan maka browser akan menampikan kata-kata yang memiliki tingkat kemiripan tinggi dengan kata yang diinputkan. Output dari proses ini dapat berupa informasi mengenai kata tersebut atau daftar kata yang memiliki tingkat kemiripan dengan kata yang diinputkan. Input yang ketiga yaitu berupa file setting. File ini akan mengalami ekstraksi dimana status aktif dari setiap fitur WordNet akan diambil dan disimpan ke dalam suatu class. Output dari proses ini adalah sebuah class yang mengandung property-property dengan nilai yang mempresentasikan kondisi suatu fitur.

3.3.3 Penjelasan Sistem

Sistem yang akan dibuat adalah sebuah aplikasi WordNet browser yang mampu berjalan pada iOS (dimiliki oleh iPhone,iPad dan iTouch). WordNet browser ini mampu menampilkan informasi berupa sinonim, definisi kata, relasi antonim, hipernim, holonim, meronim, hiponim dari kata yang diinputkan user.



Gambar 3.10
Aristektur Sistem WordNet Browser

WordNet Browser memiliki dua proses utama, yaitu proses mengekstrak lexical database serta proses pencarian kata. Lexical database terdiri dari 3 jenis file yaitu index files, data files, dan lexnames file. Setiap file tersebut akan mengalami ekstraksi data dan informasi yang didapatkan akan disimpan ke dalam dictionary, setiap file akan memilki dictionarynya sendiri. Proses pencarian kata akan memanfaatkan dictionary yang sudah berisi informasi tersebut. Pengecekan ketersediaan kata serta pengecekan class kata akan memanfaatkan index dictionary sedangkan proses ekstraksi informasi yang dimiliki oleh kata tersebut akan memanfaatkan data dictionary dan lexnames dictionary. Pada gambar 3.10 kedua proses utama dari WordNet Browser ditandakan dengan sebuah lingkaran yang memiliki warna yang berbeda. Selain kedua proses utama tersebut WordNet browser juga memili proses yang lain seperti proses pembacaan pengaturan dan proses melihat daftar kata yang pernah dicari. Proses-proses ini akan dibahas secara mendetail pada bab yang IV, V dan VI.

3.4 Desain Interface

Pada subbab ini akan dibahas mengenai desain interface dari WordNet Browser yang dibuat pada Tugas Akhir ini. Desain interface yang akan dibahas meliputi halaman loading, halaman searching, halaman result, halaman bookmark dan halaman setting.

3.4.1 Halaman Loading

Halaman loading merupakan halaman yang pertama kali ditampilkan ketika browser dijalankan. Halaman ini akan muncul selama browser masih melakukan pre-processing Lexical Database Files serta pembacaan file setting, history, dan bookmark. Pada halaman ini terdapat sebuah activity indicator yang menandakan bahwa aplikasi ini sedang berjalan dan tidak mengalami error. Untuk melambangkan WordNet Browser yang ditujukan untuk iOS maka disediakan sebuah gambar apel yang tersusun dari banyak kata. Selain itu juga terdapat tulisan WordNet yang menunjukkan nama aplikasi.

Gambar 3.11
Tampilan Halaman Loading

Pada gambar 3.11 ditunjukkan tampilan halaman loading, apa bila proses pre-processing telah selesai dilakukan maka halaman ini akan digantikan oleh halaman searching. Halaman ini tidak akan tampil kembali setelah halaman search muncul.

3.4.2 Halaman Search

Halaman Search ini akan ditampilkan setelah pre-processing selesai dilakukan, halaman ini akan muncul menggantikan halaman loading. Pada halaman ini user dapat mengetikkan kata yang ingin dicari atau memilih kata yang ditampilkan oleh browser. Selain itu user juga dapat mengakses halaman lain melalui tab bar yang tersedia. Pada gambar 3.12(a) ditunjukkan tampilan dasar dari halaman search. Terdapat sebuah textbox pada bagian atas, apabila user mengetikkan huruf atau kata pada bagian ini maka browser akan menampilkan daftar kata yang mungkin akan diketikkan oleh user seperti yan ditunjukkan pada gambar 3.12(b). Apabila user ingin mencari salah satu kata yang ditampilkan oleh user maka pencarian dapat dilakukan dengan melakukan tap pada kata tersebut.

Sedangkan bila user ingin menghilangkan keyboard yang muncul maka user dapat menekan tombol cancel.

(a) **(b)**

Gambar 3.12
Tampilan Halaman Search
(a). Tampilan Awal
(b). Tampilan Fitur Auto Complete

Pencarian kata secara manual juga dapat dilakukan dengan menekan tombol search yang ada pada keyboard. Apabila user ingin mengakses halaman lain (bookmark, history, setting) maka dapat dilakukan dengan melakukan tap pada tab bar yang berada pada bagian bawah. Icon tab bar yang memiliki tampilan yang menyala menandakan halaman yang saat ini sedang aktif.

3.4.3 Halaman Result

Halaman Result akan ditampilkan ketika pencarian suatu kata dilakukan, kata tersebut dapat berasal dari halaman search, bookmark, history, maupun result itu sendiri. Halaman ini akan menampilkan informasi berupa gloss, sinonim, dan relasi-relasi yang dimiliki oleh suatu kata. Halaman 3.13(a) menunjukkan halaman result yang dimiliki browser. Pada halaman ini informasi dari kata akan

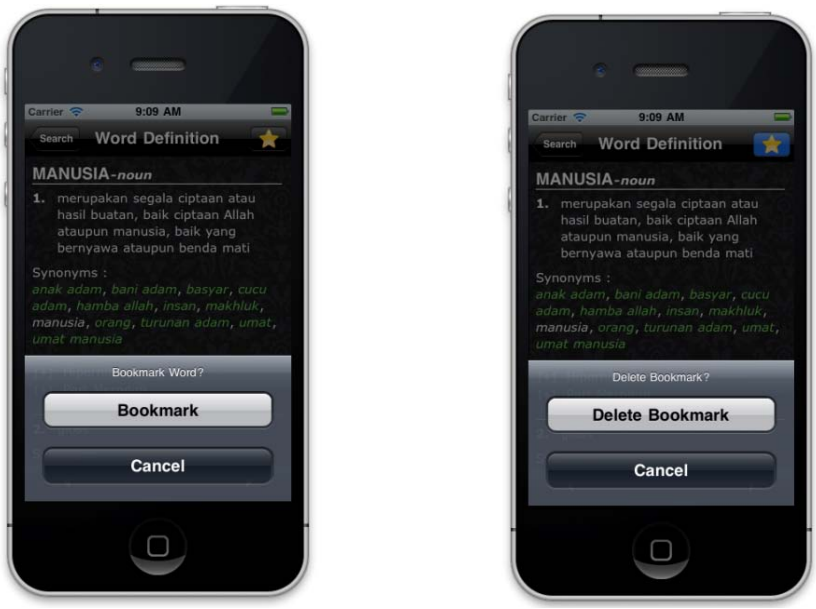
ditampilkan. Pertama kali kata itu sendiri akan ditampilkan diikuti dengan class yang dimiliki oleh kata tersebut, setelah class kata ditampilkan berikutnya gloss dari kata akan ditampilkan diikuti dengan sinonim dan relasi yang dimiliki.

(a) **(b)**

Gambar 3.13
Tampilan Halaman Result
(a).Tampilan Default
(b).Tampilan 15 synonym pertama

Perbedaan sense dari kata akan dibedakan menggunakan sebuah garis putih yang menandakan pergantian sense. Browser hanya akan menampilkan 15 sinonim pertama yang dimiliki, apabila user ingin melihat sinonim secara lengkap maka dapat menekan simbol ”[...]” sepert pada gambar 3.13(b) . Relasi yang dimiliki akan dikelompokkan berdasarkan jenisnya dan akan secara otomatis hanya menampilkan jenis relasi yang dimiliki ketika pertama kali ditampilkan, untuk melihat kata yang memiliki relasi tersebut user dapat melakukan tap pada relasi yang diinginkan. Apabila sebuah kata terdapat pada lebih dari satu class kata maka informasi yang ditampilkan akan dikelompokkan berdasarkan class kata nya. Setiap class kata akan memiliki layar tersendiri, bila ingin melihat class kata lain dapat dilakukan scoll horizontal ke kiri atau ke kanan. Jumlah dari class

kata ditunjukkan dengan adanya page control pada bagian bawah, ditunjukkan pada gambar 3.13(b). Apabila fitur group by class dinonaktifkan maka semua class kata akan berada pada satu layar dan disusun ke bawah dengan urutan noun-verb-adjective-adverb.



(a) **(b)**

Gambar 3.14
Tampilan Add Bookmark
(a). Tampilan Add Bookmark
(b). Tampilan Delete Bookmark

Apabila user merasa perlu untuk melakukan bookmark pada kata yang sedang dilihat maka user dapat menambahkan bookmark pada kata tersebut dengan melakukan tap pada tombol dengan simbol bintang yang berada di kanan atas. Apabila tombol ini mengalami penekanan maka browser akan memunculkan konfirmasi apakah kata tersebut ingin dibookmark atau tidak seperti pada gambar 3.14(a). Suatu kata yang tercatat pada catatan bookmark ditandai dengan tombol bintang yang menyala seperti pada gambar 3.14(b). Bila tombol bintang yang menyala tersebut mengalami penekanan maka akan muncul konfirmasi apakah bookmark ingin dihilangkan. Daftar kata yang telah dibookmark oleh user dapat dilihat pada halaman bookmark.

WordNet Browser menyediakan dua jenis layout yang dapat dipilih oleh user. Jenis layout pertama ditunjukkan oleh gambar 3.13. Layout ini mengikuti layout kamus konventional dimana informasi yang ditampilkan berupa gloss terlebih dahulu barulah diikuti dengan sinonim yang dimiliki oleh kata tersebut, layout ini ditujukan bagi para user yang tidak terbiasa menggunakan WordNet. Jenis layout yang kedua ditunjukkan oleh gambar 3.15(a).

(a) **(b)**

Gambar 3.15
Tampilan PWN Layout dan Suggestion
(a). Tampilan Result (PWN)
(b). Tampilan Suggestion

Layout ini mengikuti layout yang dimiliki oleh Princeton WordNet (PWN), layout ini ditujukan untuk para user yang sering menggunakan WordNet dan sudah terbiasa dengan layout dari PWN. Pada layout kedua jumlah sinonim yang ditampilkan hanya dibatasi sebanyak 3.Apabila kata yang dicari oleh user tidak terdapat pada lexical database maka browser akan menampilkan sejumlah kata yang memiliki tingkat kemiripan tinggi dengan yang kata diinputkan oleh user. Fitur ini ditunjukkan pada gambar 3.15(b). Sejumlah kata akan memiliki warna yang berbeda (tidak berwarna putih), kata-kata tersebut akan dapat dicari

secara langsung dengan melakukan tap pada kata tersebut. Kata yang dapat di tap pada merupakan bagian dari suatu relasi atau sinonim, bisa juga kata yang merupakan sugesti dari browser ketika kata yang diinputkan tidak ditemukan. Pencarian kata dengan cara ini juga akan tercatat pada history.

Pada posisi di sebelah kiri atas, terdapat sebuah tombol yang berfungsi untuk kembali ke halaman yang terakhir dikunjungi sebelum halaman result. Halaman tersebut dapat berupa halaman search, history, atau bookmark. Pada halaman result, tab bar yang pada awalnya berfungsi untuk berpindah halaman akan berubah menjadi sebuah toolbar yang berfungsi untuk melihat kata yang pernah dicari atau dibookmark. Bila user ingin melihat kata yang dicari sebelumnya maka user dapat menekan tombol back yang terdapat pada bagian bawah, tombol next juga disediakan oleh browser. Apabila user berasal dari halaman search dan menekan tombol back atau next maka kata yang ditampilkan berasal dari daftar kata yang berada pada history. Sedangkan bila user melihat kata pada halaman bookmark maka tombol back dan next akan menampilkan kata-kata yang berada pada daftar bookmark. Bila posisi kata berada pada kata terakhir atau pertama maka tombol next atau back tidak akan dapat ditekan.

3.4.4 Halaman Bookmark

Halaman bookmark akan menampilkan daftar kata yang dibookmark oleh user. User dapat mengakses halaman ini dengan menekan tombol bookmark pada tab bar ketika sedang tidak berada pada halaman result. Halaman bookmark hanya terdiri dari sebuah tombol edit pada bagian kiri atas, daftar kata bookmark pada bagian tengah dan tab bar navigasi pada bagian bawah. Tampilan awal bookmark ditunjukkan pada gambar 3.16(a).

Apabila user ingin melihat informasi dari kata yang sudah dibookmark maka user cukup melakukan tap pada kata yang ingin dilihat. Kata yang ditampilkan akan urut secara descending dimulai dari yang terakhir kali dibookmark oleh user. Kata yang dilihat oleh user pada halaman bookmark juga akan tercatat pada history. Apabila user merasa perlu untuk merubah urutan dari bookmark yang ada maka disediakan juga fasilitas untuk merubah posisi kata

yang ada dan menghapus kata. User dapat menghapus lebih dari satu kata pada saat yang bersamaan. Untuk menggunakan fitur ini maka user harus terlebih dahulu menekan tombol edit yang berada disebelah kanan atas.

(a) **(b)**

Gambar 3.16
Tampilan Halaman Bookmark
(a). Tampilan Mode View
(b). Tampilan Mode Edit

Pada gambar 3.16(b) ditunjukkan tampilan bookmark ketika user ingin melakukan perpindahan atau penghapusan kata. Pada mode edit, tab bar navigasi akan berubah menjadi tab bar miliki bookmark yang berisi sebuah tombol delete. Setiap kata akan memiliki sebuah checkbox disebelah kiri dan icon pergeseran di sebelah kanan (berupa tiga buah garis bertumpuk). Untuk melakukan perubahan posisi kata, user cukup melakukan drag and drop pada icon pergeseran tersebut. Sedangkan untuk melakukan penghapusan kata maka user harus terlebih dahulu menandai kata mana yang ingin dihapus. Penandaan kata dapat dilakukan dengan melakukan tap pada lingkaran yang berada di sebelah kiri kata. Kata yang ditandai akan memiliki tanda centang pada lingkaran tersebut. Pada gambar 3.16(b) ditunjukkan kata yang ditandai dan kata yang tidak ditandai. Ketika semua kata

telah ditandai maka user cukup menekan tombol delete yang berada di sebelah kiri bawah (berupa tong sampah). Untuk kembali ke mode awal maka user dapat menekan tombol done yang berada di sebelah kanan atas.

3.4.5 Halaman History

Halaman history akan menampilkan semua kata yang pernah dicari oleh user. Daftar kata akan ditampilkan secara descending dimulai dari kata yang terakhir kali dicari oleh user. Halaman History ini hanya berupa daftar kata yang pernah dicari oleh user seperti yang ditampilkan pada gambar 3.17.

Gambar 3.17
Tampilan Halaman History

Ketika memasuki halaman history kata yang terakhir kali dilihat oleh user akan memiliki background berwarna biru, apabila user melihat kata lain yang ada pada halaman history maka secara otomatis kata tersebut akan memiliki background berwarna biru. Secara default browser akan mencatat kata-kata yang pernah dicari oleh user. Jumlah kata yang dicatat oleh browser adalah 50, apabila pencarian telah melebihi jumlah tersebut maka penghapusan akan mengikuti prinsip LFU (Least Frequently Used) dimana kata yang terlama akan dihapus. Posisi kata pada history bersifat dinamis, apabila sebuah kata yang berada di

posisi bawah mengalami pencarian ulang maka kata tersebut akan berada di posisi paling atas. Pengaturan tersebut akan dapat diubah pada halaman setting.

3.4.6 Halaman Setting

Halaman setting berisi semua pengaturan fitur yang dimiliki oleh browser. Pengaturan tersebut antara lain adalah : show gloss, show class , group by class, auto complete, PWN layout, Record history dan number of record.

Gambar 3.18
Tampilan Halaman Setting

Halaman setting terdiri dari on off switch dan sebuah slider, pada gambar 3.18 ditampilkan pengaturan-pengaturan yang menggunakan on off switch. Fitur show gloss akan menentukan apakah gloss suatu kata akan ditampilkan, fitur show class akan menentukan class kata apa saja yang ditampilkan. Apabila semua class dinonaktifkan maka show gloss secara otomatis akan dinonaktifkan. Group by class akan membagi setiap class kata ke dalam layar tersendiri, satu class kata akan memiliki satu layar. Fitur auto complete akan menentukan apakah browser akan menampilkan daftar kata yang mungkin ingin diketikkan user pada halaman search. Fitur PWN layout akan menentukan apakah layout dari PWN akan

digunakan. Fitur record history akan menentukan apakah kata yang dicari oleh user akan dicatat, bila fitur ini diaktifkan akan muncul slider untuk menentukan jumlah kata yang akan dicatat dengan nilai minimum 10 dan maximal 100.

3.5 Software yang Dibutuhkan

Untuk dapat mengimplementasikan aplikasi yang telah dibuat ini maka dibutuhkan software-software yang tepat. Oleh karena itu, dalam pembuatannya diperlukan beberapa software pendukung, antara lain yaitu:

- Xcode

  Xcode adalah IDE utama dari iPhone, dalam IDE ini disediakan banyak fasilitas-fasilitas lain untuk menunjang pengembangan aplikasi pada iPhone, diantaranya adalah iPhone simultor, yaitu suatu aplikasi yang dapat digunakan untuk mensimulasikan aplikasi yang dibuat. Ada juga aplikasi Interface Builder yang digunakan untuk merancang user interface dari aplikasi yang dibuat. Semua aplikasi tersebut adalah bagian dari Xcode.

- Adobe Photoshop CS5

  Software digital image processing yang digunakan untuk membuat icon dan background yang digunakan dalam aplikasi ini. Software ini juga dapat berfungsi untuk menyesuaikan format gambar yang disimpan agar dapat dibaca oleh iPhone, yaitu format .PNG.

BAB IV
EKSTRAKSI INFORMASI LEXICAL DATABASE FILES

Lexical Database Files merupakan sekumpulan text files yang menyimpan informasi mengenai gloss, sinonim, dan relasi yang dimiliki oleh kata-kata pada suatu bahasa. Lexical Database Files ini dikembangkan oleh Princeton University sebagai bagian dari PrincetonWordNet (PWN), yaitu sebuah WordNet berbahasa Inggris. Dalam perkembangannya Princeton University menyediakan sebuah library untuk mengekstrak informasi yang terdapat dalam Lexical Database Files agar PWN dapat dimanfaatkan oleh para programmer yang memiliki kepentingan terhadap PWN. Sayangnya library tersebut hanya tersedia untuk bahasa pemrograman Java dan .NET. Oleh karena itu pada Tugas Akhir ini dikembangkan sebuah library untuk bahasa pemrograman Objective-C yang berperan untuk melakukan ekstraksi informasi pada Lexical Database Files. Library yang dikembangkan pada Tugas Akhir ini berupa sebuah class bernama WordNet.

Class WordNet berfungsi sebagai ekstraktor Lexical Database serta penghubung antara Lexical Database dengan WordNet Browser. Lexical Database Files akan mengalami pre-processing dimana informasi-informasi pada Lexical Database Files tersebut akan disimpan ke dalam sekumpulan dictionary. Ketika WordNet Browser melakukan pencarian kata maka class WordNet akan mencari kata tersebut pada dictionary yang dimilikinya dan mengembalikan informasi yang dimiliki oleh kata tersebut. Agar pengguna class WordNet dapat melakukan pencarian tersebut maka terdapat beberapa prosedur yang disediakan oleh class WordNet. Terdapat lima buah prosedur bersifat public yang dapat digunakan untuk melakukan ekstraksi informasi milik sebuah kata, selain kelima prosedur tersebut juga terdapat prosedur private yang membantu dalam melakukan ekstraksi informasi. Tabel 4.1 Menunjukkan daftar prosedur yang dimiliki oleh libary WordNet.

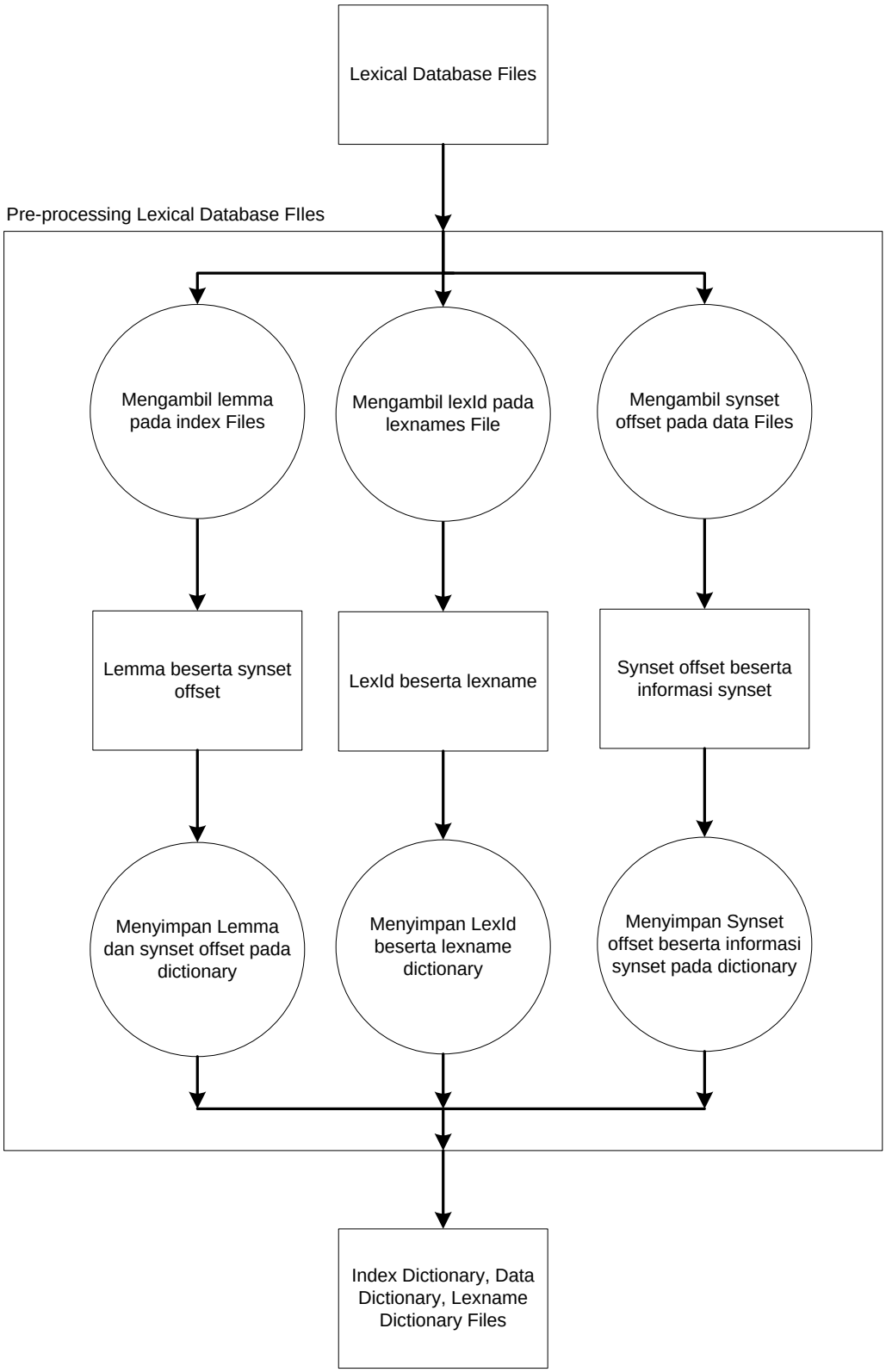
Tabel 4.1
Daftar Prosedur Class WordNet

Nama Prosedur	Sifat	Fungsi
getWordData	Public	Mendapatkan synset offset sebuah kata.
getSynsetData	Public	Mendapatkan informasi yang dimiliki sebuah synset offset.
getSynsetLemma	Public	Mendapatkan sinonim pada index tertentu.
getLexName	Public	Mendapatkan lexname dari lexFileNum tertentu.
getRelationName	Public	Mendapatkan jenis relasi dari relation pointer tertentu.
readFromFile	Private	Memindahkan isi Lexical Database ke dalam dictionary yang bersangkutan.
getWordClassData	Private	Melakukan Ekstraksi berdasarkan String Data yang diterima.

Setiap prosedur tersebut memiliki fungsi yang spesifik dan setiap prosedur tersebut akan dijelaskan pada subbab yang berhubungan dengan proses yang dilakukan. Walaupun begitu semua prosedur tersebut memiliki ketergantungan satu sama lain. Misalnya prosedur getSynsetLemma memerlukan synset offset yang didapatkan dengan menggunakan prosedur getWordData. Hasil ekstraksi akan disimpan ke dalam sebuah struktur penyimpanan yang khusus, struktur-struktur yang digunakan akan dijelaskan bersama dengan proses yang dilakukan. Proses-proses yang dilakukan untuk melakukan ekstraksi informasi akan dijelaskan secara detail pada bab ini.

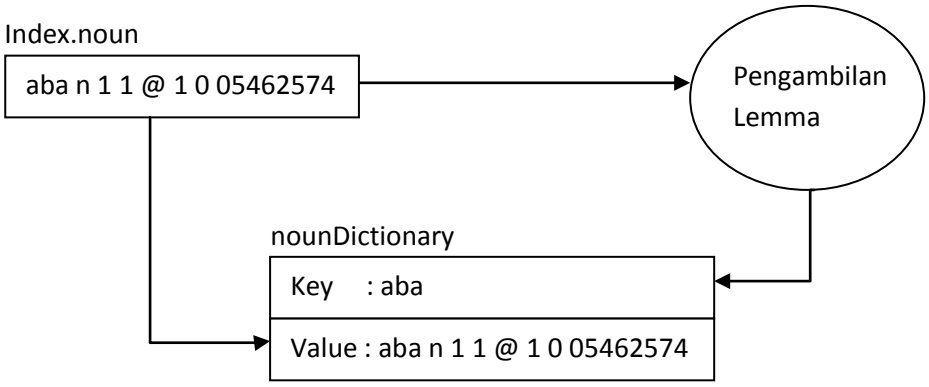
4.1 Pre-Processing Lexical Database Files

Sebelum ekstraksi informasi yang dimiliki oleh suatu kata dapat dilakukan oleh class WordNet, terlebih dahulu harus dilakukan pre-processing terhadap Lexical Database Files. Tahap pre-processing ini mencakup pengambilan data pada kelompok index files, data files, dan lexname file dan menyimpannya ke dalam dictionary yang dimiliki oleh class WordNet.



Gambar 4.1
Pre-Processing Lexical Database Files

Pada gambar 4.1 ditunjukkan bahwa tahap pre-processing Lexical Database Files dilakukan dua buah tahap proses. Tahap pertama adalah pengambilan string dari index pertama hingga ditemukannya spasi pertama, aturan ini berlaku unuk kelompok index file dan data file. Untuk file lexnames pemotongan string dilakukan berdasarkan tab atau "\t" karena pada file lexnames pemisah antar fieldnya bukanlah spasi melainkan tab. Tahap yang kedua adalah menyimpan string hasil ekstraksi tersebut ke dalam dictionary, string yang didapatkan dari proses pemotongan berdasarkan spasi atau tab akan menjadi key dari dictionary sedangkan keseluruhan string tersebut akan menjadi value bagi dictionary. Output dari proses ekstraksi ini berupa dictionary yang berisi data hasil ekstraksi Lexical Database Files. Gambar 4.2 menunjukkan ilustrasi penyimpanan data pada file index.noun ke dalam noun dictionary.



Gambar 4.2
Ilustrasi Penyimpanan pada Dictionary

Pre-processing hanya bertujuan memindahkan isi dari Lexical Database Files ke dalam dictionary dengan lemma / synset offset / lexical id sebagai key dari dictionary dengan tujuan mempercepat proses pencarian pada Lexical Database Files. Apabila ekstraksi informasi pada setiap kata dilakukan pada tahap ini maka akan pre-processing akan memakan waktu yang lama. Tahap Pre-processing ini ditangani oleh prosedur readFromFile milik class WordNet. Prosedur tersebut bersifat private dan hanya ditujukan sebagai inisialisasi class WordNet yang dijalankan ketika instrance dari class WordNet dibuat.

Segmen Program 4.1 Prosedur ReadFromFile

```
1:   #include <stdio.h>
2:   #include <string.h>
3:   -(void) readFromFile :(NSString *)path
             forDict : (NSMutableDictionary *) dict{
4:   if ([[NSFileManager defaultManager] fileExistsAtPath:path])
5:    {
6:      const char *target = [path
       StringUsingEncoding:NSASCIIStringEncoding];
7:      char str[10000];
8:      FILE *fp = fopen(target, "r");
9:      int lineNumber=0;
10:     while(fgets(str, 10000, fp))
11:     {
12:      lineNumber++;
13:      char keys[] = "lexnames";
14:      char *j;
15:      j = strstr (target,keys);
16:      if (j == nil)
17:      {
18:       if (lineNumber>29)
19:       {
20:        char keys[] = " ";
21:        int i;
22:        i = strcspn (str,keys);
23:        char str2[i];
24:        strncpy(str2,str,i);
25:        str2[i] = '\0';
26:        NSString *key = [NSString stringWithUTF8String:str2];
27:        NSString *value = [NSString stringWithUTF8String:str];
28:        [dict setObject:value forKey:key];
29:       }
30:      }else {
31:        char keys[] = "\t";
32:        int i;
33:        i = strcspn (str,keys);
34:        char str2[i];
35:        strncpy(str2,str,i);
36:        str2[i] = '\0';
37:        NSString *key = [NSString stringWithUTF8String:str2];
38:        NSString *value = [NSString stringWithUTF8String:str];
39:        [dict setObject:value forKey:key];
40:      }
41:   }
42:   fclose(fp);
43:   }
44:  }
```

Segmen Program 4.1 menunjukan bahwa prosedur readFromFile ini menggunakan bahasa pemrograman Objective-C dan C++. Bahasa pemrograman C++ digunakan untuk mempercepat proses pembacaan dan pemotongan text file pada Lexical Database Files. Pada baris 1 dan baris 2 ditunjukkan include yang

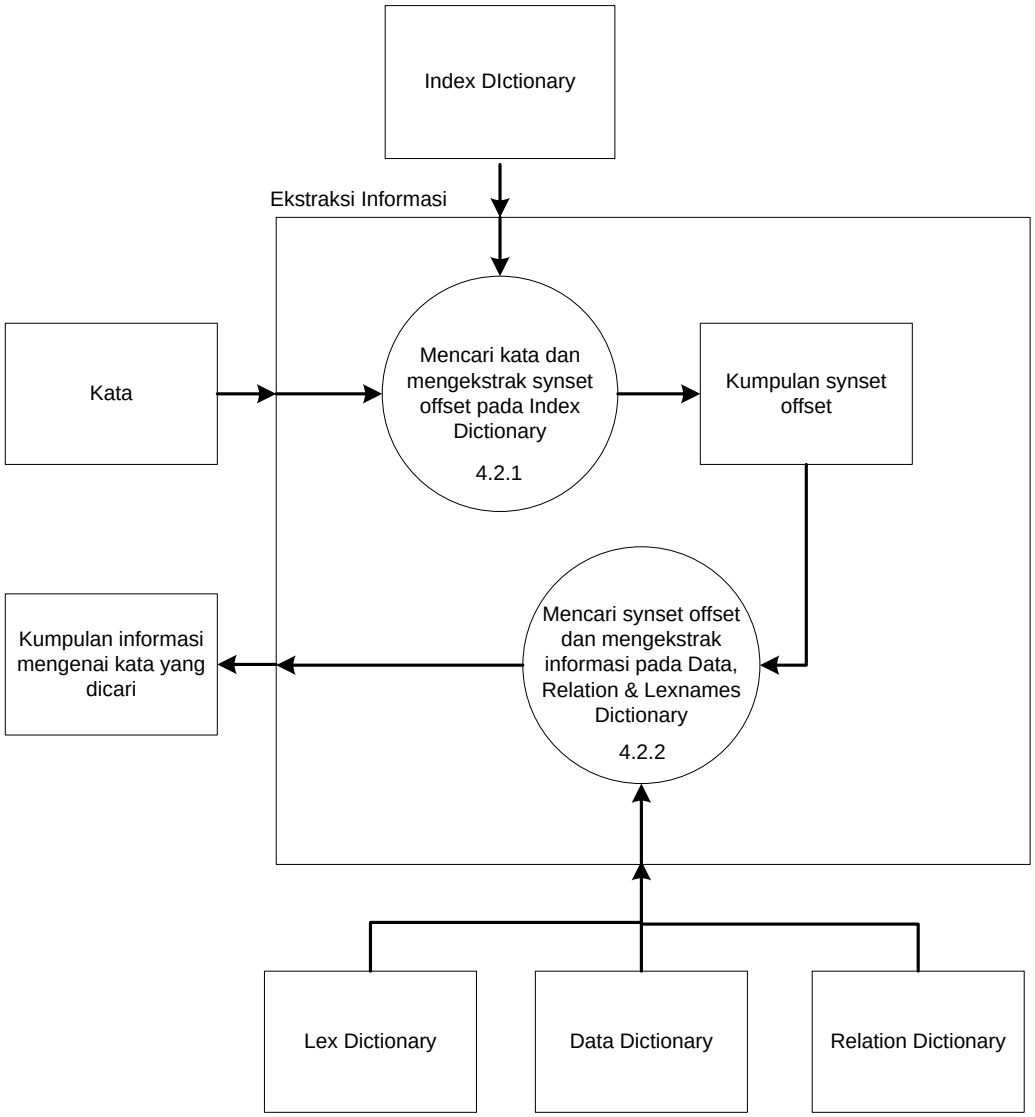
digunakan oleh C++ kedua include tersebut mengandung fungsi-fungsi string dan file. Pada baris 4 dilakukan pemeriksaan apakah file benar-benar ada pada path yang dikirimkan sebagai parameter. Baris 6-8 merupakan insialisasi yang diperlukan untuk melakukan pembacaan file pada C++. Pada baris 9 dilakukan insialisasi variabel lineNumber, variabel ini bertujuan untuk mengetahui apakah license agreement dari file sudah terlewati. Pembacaan file dilakukan pada baris 10 dimana satu baris data akan diambil atau higga data mencapai 10000 byte, hasil pembacaan data akan disimpan pada variabel str. Pada baris 12-16 akan dilakukan pemeriksaan apakah file yang dibaca merupakan file lexnames, jika benar maka program akan melanjutkan ke baris 31 dimana dilakukan pemisahan data berdasarkan tab atau "\t" dan memasukkannya ke dalam dictionary.

Jika ternyata file tersebut bukanlah file lexnames maka program akan melanjukan ke baris 18 dimana akan dilakukan pemeriksaan apakah license agreement sudah terlewati. Pada baris 20-25 dan 31-36 program akan melakukan pemotongan string, dimana string pertama hingga ditemukannya spasi atau tab pertama kali akan dianggap sebagai key dan keseluruhan string akan disimpan sebagai value pada dictionary. Hal tersebut berlaku baik untuk index file maupun data file. Konversi dari tipe data char * milik C++ menjadi tipe data NSString milik Objective-C dilakukan pada baris 26-27 dan 37-38. Seteleh konversi selesai dilakukan maka NSString akan dimasukkan ke dalam NSMutableDictionary yang juga didapatkan melalui parameter, proses ini terjadi pada baris 28 dan 38. Prosedur ini diakhiri dengan menutup koneksi dengan file yang dilakukan pada baris 42.

4.2 Ekstraksi Informasi Sebuah Kata

Setelah Lexical Database Files mengalami pre-processing, kini class WordNet telah siap melakukan ekstraksi informasi pada Lexical Database Files. Proses ekstraksi informasi ini mengalami dua tahap proses, tahap pertama adalah pencarian kata pada index dictionary dan tahap kedua ekstraksi informasi pada data dictionary. Tahap pertama akan menghasilkan kumpulan synset yang dimiliki oleh kata dan pada tahap kedua akan dihasilkan informasi mengenai gloss,

sinonim, dan relasi yang dimiliki oleh setiap synset tersebut. Kedua tahap proses ekstraksi informasi ditunjukkan berupa dua buah lingkaran pada gambar 4.3.



Gambar 4.3
Proses Ekstraksi Informasi

Pada gambar 4.3 ditunjukkan bahwa terdapat empat buah kelompok dictionary yang berperan dalam ekstraksi informasi kata yaitu index dictionary, lex dictionary, data dictionary, dan relation dictionary. Keempat kelompok dictionary tersebut merupakan variabel-variabel yang dimiliki oleh class

WordNet, pada tabel 4.2 ditunjukkan penjelasan mengenai masing-masing kelompok dictionary.

Tabel 4.2
Dictionary milik Class WordNet

Nama dictionary	Kelompok Dictionary	Fungsi
noun	Index Dictionary	Menampung data index.noun
verb	Index Dictionary	Menampung data index.verb
adj	Index Dictionary	Menampung data index.adj
adv	Index Dictionary	Menampung data index.adv
nounData	Data Dictionary	Menampung data data.noun
verbData	Data Dictionary	Menampung data data.verb
adjData	Data Dictionary	Menampung data data.adj
advData	Data Dictionary	Menampung data data.adj
lex	Lex Dictionary	Menampung data lexnames
relation	Relation Dictionary	Mencatat relation pointer

Kelompok index dictionary dan kelompok data dictionary tediri dari empat buah dictionary, dimana setiap dictionary mewakili satu buah class kata sedangkan kelompok lex dictionary dan relation dictionary hanya terdiri dari satu buah dictionary. Kelompok index dictionary akan menampung data yang didapat dari kelompok index dictionary sedangkan kelompok data dictionary akan menampung data yang didapat dari kelompok data dictionary dan lex dictionary akan menampung data yang didapat dari lexnames file. Ketiga kelompok dictionary tersebut mendapatkan nilai yang berasal dari Lexical Database Files Berbeda dengan kelompok relation dictionary yang nilainya tidak berasal dari Lexical Database Files. Seperti yang telah dijelaskan pada bab II bahwa setiap jenis relasi akan memiliki sebuah simbol yang mewakili jenis relasi tersebut. Karena relasi ini berlaku untuk setiap bahasa maka relation dictionary memiliki nilai yang statis dan pengisian dari dictionary ini dilakukan secara otomatis oleh

class WordNet. Pengisian nilai relation dictionary ditunjukkan oleh segmen program 4.2.

Segmen Program 4.2 Pengisian Relation Dictionary

```
1:  [relation setObject:@"Antonim"               forKey:@"!"];
2:  [relation setObject:@"Hipernim"              forKey:@"@"];
3:  [relation setObject:@"Insctance Hipernim"    forKey:@"@i"];
4:  [relation setObject:@"Hiponim"               forKey:@"~"];
5:  [relation setObject:@"Instance Hiponim"      forKey:@"~i"];
6:  [relation setObject:@"Member Holonim"        forKey:@"#m"];
7:  [relation setObject:@"Substance Holonim"     forKey:@"#s"];
8:  [relation setObject:@"Part Holonim"          forKey:@"#p"];
9:  [relation setObject:@"Member Meronim"        forKey:@"%m"];
10: [relation setObject:@"Substance Meronim"     forKey:@"%s"];
11: [relation setObject:@"Part Meronim"          forKey:@"%p"];
12: [relation setObject:@"Atribut"               forKey:@"="];
13: [relation setObject:@"Derivationally Related Form"
                forKey:@"+"];
14: [relation setObject:@"Domain of synset - Topic"
                forKey:@";c"];
15: [relation setObject:@"Member of this domain - Topic"
                forKey:@"-c"];
16: [relation setObject:@"Domain of synset - Region"
                forKey:@";r"];
17: [relation setObject:@"Member of this domain - Region"
                forKey:@"-r"];
18: [relation setObject:@"Domain of synset - Usage"
                forKey:@";u"];
19: [relation setObject:@"Member of this domain - Usage"
                forKey:@"-u"];
20: [relation setObject:@"Entailment"            forKey:@"*"];
21: [relation setObject:@"Cause"                 forKey:@">"];
22: [relation setObject:@"Also See"              forKey:@"^"];
23: [relation setObject:@"Verb Group"            forKey:@"$"];
24: [relation setObject:@"Similar To"            forKey:@"&"];
25: [relation setObject:@"Participle of Verb"    forKey:@"<" ];
26: [relation setObject:@"Pertainim (Perains to Noun)"
                forKey:@"\\"];
```

Segmen program 4.2 menunjukkan proses pengisian relation dictionary. Key pada dictionary ini berupa simbol yang mewakili suatu relasi, sedangkan valuenya adalah relasi yang diwakilkan oleh simbol tersebut. Baris 1 menunjukkan proses menyimpan simbol “!” sebagai key pada dictionary dan relasi antonim pada valuenya, baris 2-26 menunjukkan proses penyimpanan berbagai relasi yang lain. Pada Tugas Akhir ini relasi yang disediakan adalah antonim, hipernim, hiponim, holonim dan meronim, tetapi semua jenis relasi akan

disimpan pada dictionary. Hal ini dilakukan untuk mempersiapkan apabila relasi tersebut tersedia pada Lexical Database Files pada bahasa lain.

4.2.1 Mencari Kata dan Mengekstrak Synset Offset pada Index Dictionary

Tahap pertama dari pengekstrakan informasi milik kata adalah melakukan ekstraksi synset yang dimiliki oleh kata tersebut. Ekstraksi ini dilakukan pada kumpulan index dictionary yang dimiliki oleh class WordNet. Kata yang akan dicari akan mengalami pemeriksaan pada setiap class kata, perlu diingat bahwa sebuah kata bisa berada pada lebih dari satu class kata. Apabila kata tersebut ditemukan pada sebuah class kata maka synset offset yang dimiliki oleh kata tersebut akan disimpan. Selain synset offset juga terdapat informasi lain yang dapat diekstrak dari index dictionary, oleh karena itu class WordNet menggunakan struktur penyimpanan berupa class bernama Word untuk menampung hasil ekstraksi pada index dictionary.

Tabel 4.3
Struktur Cass Word

Nama Variabel	Tipe Variabel	Fungsi
lemma	NSString	Menyimpan lemma kata
noun	WordClass	Menyimpan hasil ekstraksi noun dictionary
verb	WordClass	Menyimpan hasil ekstraksi verb dictionary
adj	WordClass	Menyimpan hasil ekstraksi adj dictionary
adv	WordClass	Menyimpan hasil ekstraksi adv dictionary

Pada tabel 4.3 ditunjukkan bahwa terdapat 2 tipe variabel pada class Word yaitu NSString dan WordClass. Variabel dengan tipe NSString digunakan untuk menyimpan lemma dari kata sedangkan keempat variabel dengan tipe WordClass digunakan untuk meyimpan informasi dari setiap synset yang dimiliki oleh lemma yang terdapat pada setiap class kata. Setiap variabel pada class Word juga dideklarasikan sebagai property dengan parameter nonatomic dan retain.

Parameter nonatomic berarti property tersebut tidak akan mengatasi masalah berupa *race-condition* apabila dilakukan proses multi-threading. Karena WordNet Browser yang dibuat pada Tugas Akhir ini tidak menggunakan multi-threading maka parameter yang digunakan adalah nonatomic. Sedangkan property retain menandakan bahwa property tersebut akan terus tersimpan didalam memory. Dengan adanya deklarasi sebagai property maka variabel-variabel yang dimiliki oleh class Word secara otomatis akan memiliki accessor dan mutator.

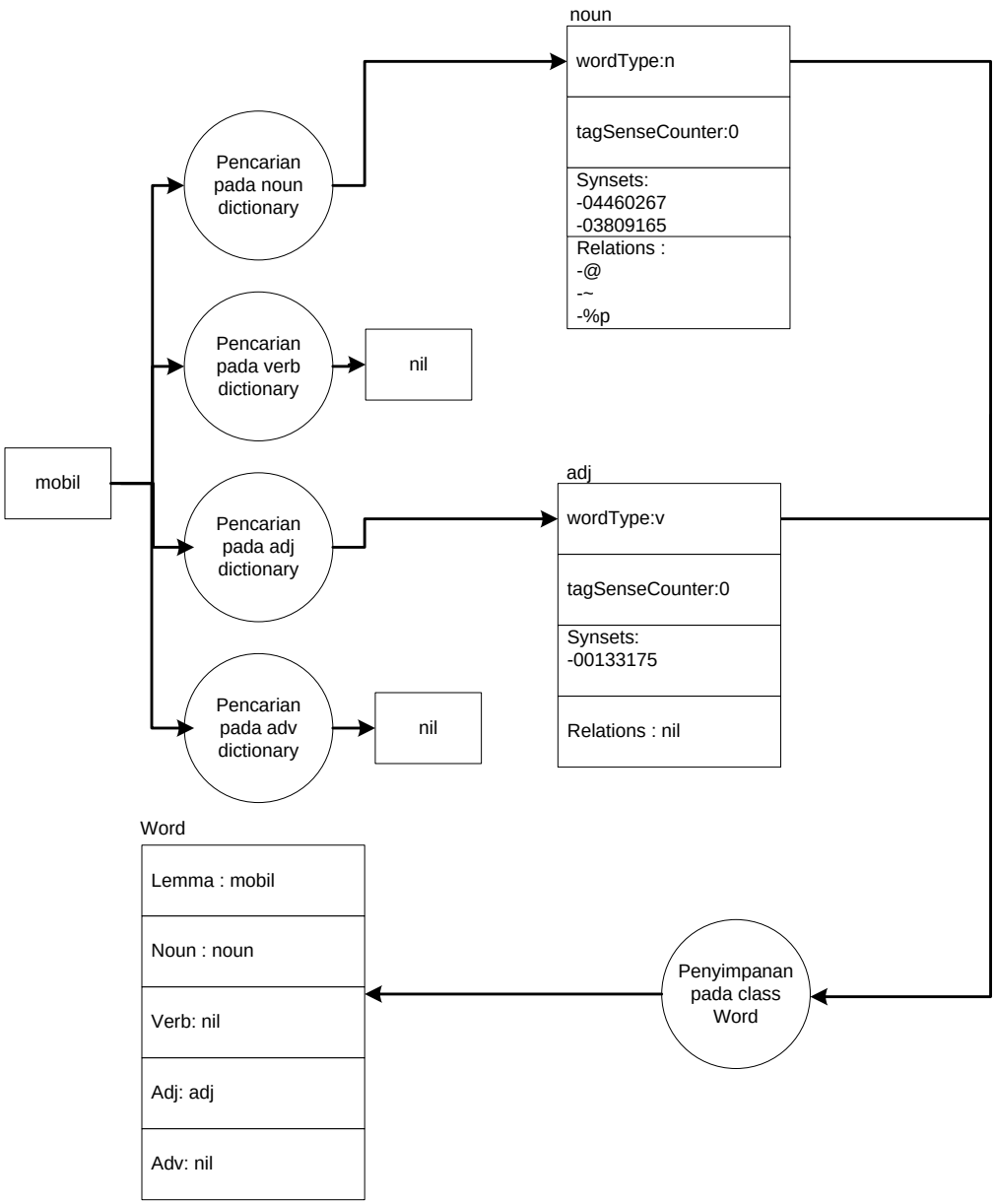
Pada struktur class Word ditunjukkan bahwa class Word Menggunakan sebuah class lain sebagai bagiannya, class tersebut adalah class WordClass. Peranan dari class WordClass adalah untuk menampung hasil ekstraksi synset pada setiap class kata, karena itu terdapat empat buah variabel dengan tipe WordClass pada class Word dimana setiap variabel melambangkan satu buah class kata.

Tabel 4.4
Struktur Class WordClass

Nama Variabel	Tipe Variabel	Fungsi
wordType	NSString	Menyimpan class kata
tagSenseCounter	NSString	Menyimpan jumlah penggunaan lemma
synsets	NSMutableArray	Menyimpan synset offset yang dimiliki
relations	NSMutableArray	Menyimpan pointer relation yang dimiliki

Pada tabel 4.4 ditunjukkan bahwa variabel synsets dan relations memiliki tipe NSMutableArray hal ini dilakukan untuk mengatasi kondisi dimana suatu kata memiliki lebih dari satu buah synset dan suatu kata memiliki lebih dari satu buah relasi. Tetapi pada tahap ini masih belum diketahui synset manakah yang memiiliki relasi tersebut. Jumlah sense yang dimiliki oleh kata tidak dicatat karena jumlah sense dari suatu kata pasti bernilai sama dengan jumlah synset yang dimiliki.Setiap variabel yang terdapat pada class WordClass juga dideklarasikan sebagai property dengan parameter nonatomic dan retain.

Proses ekstraksi yang terjadi meliputi pencarian pada keempat index history, pencarian kata akan dilakukan pada setiap index dictionary dan hasilnya akan disimpan dengan struktur class Word. Gambar 4.4 menunjukkan ilustrasi proses ekstraksi yang terjadi.



Gambar 4.4
Ilustrasi Pencarian Kata pada Index Dictionary

Pada gambar 4.4 ditunjukkan contoh pencarian terhadap kata mobil. Kata tersebut akan dicari pada keempat index dictionary. Hasil dari pencarian tersebut akan diproses dan disimpan dengan menggunakan struktur class WordClass, tetapi apabila tidak ditemukan maka hasil pencarian akan berupa nil. Hasil pencarian yang tidak bernilai nill akan disimpan ke dalam variabel dengan struktur class Word.

Algoritma 4.1 menunjukkan algoritma yang digunakan untuk melakukan ekstraksi synset yang dimiliki oleh suatu kata. Algoritma tersebut sesuai dengan proses yang ditunjukkan oleh gambar 4.4.

Algoritma 4.1 Ekstraksi Informasi Pada Index Dictionary

```
[Digunakan sebagai algoritma dasar untuk ekstraksi synset offset]

1. [Lakukan iterasi untuk mengambil semua synset offset pada semua
   index dictionary]
    1.1  FOR EACH dictionary in indexDictionary
         1.1.1   IF SearchKata in dictionary
              1.1.1.1  StringData = GET_DATA(SearchKata)

2. [String Data yang didapatkan akan mengalami proses ekstraksi]
    2.1 tempSplit = SPLIT_WITH_DELIMITER(StringData,” “)
    2.2 Class = GET_INDEX(tempSplit,1)
    2.3 SynsetNumber = GET_INDEX(tempSplit,2)
    2.4 RelationNumber = GET_INDEX(tempSplit,3)
    2.5 TagSense = GET_INDEX(tempSplit,RelationNumber+5)

3. [Lakukan ekstraksi synset offset]
    3.1 FOR (i=0;i<SynsetNumber;i++)
         3.1.1   Synset[i] =
                 GET_INDEX(tempSplit,i+RelationNumber+6)

4. [Lakukan ekstraksi relasi]
    4.1 FOR (i=0;i<RelationNumber;i++)
         4.1.1   Relation[i] = GET_INDEX(tempSplit,i+4)
```

Alogitma 4.1 tersebut diwujudkan ke dalam dua buah prosedur. Langkah satu dari algoritma 4.1 ditangani oleh prosedur getWordData sedangkan langkah 2-4 ditangani oleh prosedur getWordClassData. Prosedur getWordClassData adalah sebuah prosedur private yang berperan mengekstrak informasi dari sebuah string yang berasal dari index dictionary. Prosedur ini digunakan oleh prosedur getWordData untuk mengekstrak string yang didapatkan dari suatu index dictionary. Ekstraksi yang dihasilkan berupa synset offset, relasi, class synset dan

tag sense counter synset. Sedangkan prosedur getWordData adalah sebuah prosedur bersifat public yang dapat diakses oleh pengguna class WordNet untuk melakukan ekstraksi kata.

Segmen Program 4.3 Prosedur getWordClassData

```
1:  -(id) getWordClassData:(NSString *)stringData
2:  {
3:   id wordDataType=[[WordClass alloc]init];

4:   NSArray *tempSplit = [stringData
     componentsSeparatedByString:@" "];
5:   int numOfSynsetsID = [[tempSplit objectAtIndex:2] intValue];
6:   int numOfRelations = [[tempSplit objectAtIndex:3] intValue];

7:   NSMutableArray *arrayOfSynsets = [[NSMutableArray alloc]
     init];
8:   for(int i=0;i<numOfSynsetsID;i++)
9:   {
10:   [arrayOfSynsets addObject:[tempSplit
      objectAtIndex:i+numOfRelations+6]];
11:  }

12:  NSMutableArray *arrayOfRelations = [[NSMutableArray alloc]
     init];
13:  for(int i=0;i<numOfRelations;i++)
14:  {
15:   [arrayOfRelations addObject:[tempSplit objectAtIndex:i+4]];
16:  }
17:
18:  [wordDataType setWordType:[tempSplit objectAtIndex:1]];
19:  [wordDataType setSynsets:arrayOfSynsets];
20:  [wordDataType setRelations:arrayOfRelations];

21:  [wordDataType setTagSenseCounter:[tempSplit
     objectAtIndex:numOfRelations+5] ];

22:  return [wordDataType autorelease];
23: }
```

Pada baris 3 segmen program 4.3 disiapkan sebuah variable dengan tipe data WordClass untuk menampung informasi hasil dari ekstraksi string yang menjadi parameter prosedur. Pada baris 4-6 dilakukan pemotongan string berdasarkan spasi dan dilakukan ekstraksi untuk jumlah synset dan jumlah relasi yang dimiliki kata tersebut. Kemudian ekstraksi synset yang dimiliki akan dilakukan berdasarkan jumlah synset yang telah didapatkan, proses ini ditunjukkan pada baris 8-11. Setelah synset didapatkan maka relasi juga akan

diekstrak, proses ini ditunjukkan pada baris 13-16. Pada baris 18-21 dilakukan penyimpanan informasi hasil ekstraksi ke dalam variabel penampung.

Prosedur getWordData adalah sebuah prosedur bersifat public yang berperan melakukan ekstraksi pada index dictionary, dengan bantuan prosedur getWordClassData. Prosedur getWordData akan menerima inputan berupa kata dan mencari kata tersebut pada semua index dictionary, apabila sebuah kata terdapat pada suatu index dictionary maka string yang didapatkan akan dikirim kepada prosedur getWordClassData. Hasil ekstraksi akan disimpan pada variabel dengan tipe data class Word. Segmen program 4.4 akan menunjukkan prosedur getWordData.

Segmen Program 4.4 Prosedur getWordData

```
1:  -(id)getWordData:(NSString *)word
2:  {
3:     id data = [[Word alloc]init];

4:     [data setLemma:word];

5:     NSString *stringData=[[NSString alloc]init];
6:     if (stringData =
       [noun objectForKey:[[data lemma]lowercaseString]])
7:        [data setNoun:[self getWordClassData:stringData]];
8:     if (stringData =
       [verb objectForKey:[[data lemma]lowercaseString]])
9:        [data setVerb:[self getWordClassData:stringData]];
10:    if (stringData =
       [adj  objectForKey:[[data lemma]lowercaseString]])
11:       [data setAdj:[self getWordClassData:stringData]];
12:    if (stringData =
       [adv  objectForKey:[[data lemma]lowercaseString]])
13:       [data setAdv:[self getWordClassData:stringData]];

14:    return data;
15: }
```

Input dari prosedur ini adalah sebuah NSString yang merupakan kata yang ingin dicari, dan mencarinya pada semua class kata. Pada baris 1 segmen program 4.4 disiapkan sebuah variabel dengan tipe Word untuk menampung data hasil ekstraksi. Pada baris 4 kata yang diterima sebagai parameter akan disimpan sebagai lemma kata. Pemeriksaan tersebut akan dilakukan pada setiap index dictionary, apa bila index dictionary tertentu tidak mengembalikan nilai nil untuk

key kata yang diinputkan maka string yang diterima akan dikirimkan kepada prosedur getWordClassData dan hasil ekstraksi akan disimpan pada variabel penampung. Proses ini ditunjukkan pada baris 6-13.

Bagi para pengguna class WordNet dapat menggunakan prosedur getWordData untuk melakukan ekstraksi pada index dictionary. Prosedur getWordClass data merupakan sebuah prosedur private yang tidak dapat oleh pengguna class WordNet.

4.2.2 Mencari Synset Offset dan Mengekstrak Informasi pada Data, Relation dan Lexname Dictionary

Tahap kedua dari proses ekstraksi informasi adalah mengekstrak informasi yang dimiliki oleh synset tersebut pada data dictionary, relation dictionary dan lex dictionary. Setiap synset yang dimiliki oleh kata akan dicari pada data dictionary yang sesuai dengan synset tersebut. Pada subbab 4.2.1 dijelaskan bahwa ekstraksi synset offset juga akan menghasilkan class kata tempat synset tersebut berada. Pencarian pada data dictionary akan menghasilkan gloss, sinonim, lexFileNum dan pointer relation yang dimiliki oleh synset. Pointer relation yang didapatkan akan menjadi key dicari pada relation dictionary untuk mengetahui jenis relasi yang disimbolkan. Sedangkan lexFileNum akan menjadi key yang dicari pada lex dictionary untuk mengetahui lexname dari synset tersebut. Untuk menampung seluruh informasi yang dimiliki oleh synset maka class WordNet menggunakan struktur class bernama Synset. Tabel 4.5 menunjukkan struktur data yang dimiliki oleh class Synset.

Tabel 4.5
Struktur Class Synset

Nama Variabel	Tipe Variabel	Fungsi
lexFileNum	NSString	Menyimpan lexFileNum
ssTypes	NSString	Menyimpan class kata synset
synsetWords	SynsetWord	Menyimpan sinonim synset
synsetRelations	NSMutableArray	Menyimpan relasi synset

Untuk menyimpan informasi mengenai sinonim yang dimiliki oleh synset maka class Synset menggunakan bantuan dari class SynsetWord. Hal ini dilakukan karena dalam proses ekstraksi sinonim terdapat beberapa informasi lain yang bisa didapatkan. Sedangkan untuk menyimpan informasi mengenai relasi yang dimiliki oleh synset digunakan bantuan dari class SynsetRelation, variabel-variabel dengan tipe SynsetRelation akan disimpan pada array dengan nama synsetRelations milik class Synset. Setiap variabel pada class Synset dideklarasikan sebagai property dengan tipe nonatomic dan retain, hal ini juga terjadi pada class SynsetWord dan SynsetRelation.

Tabel 4.6
Struktur Class SynsetWord

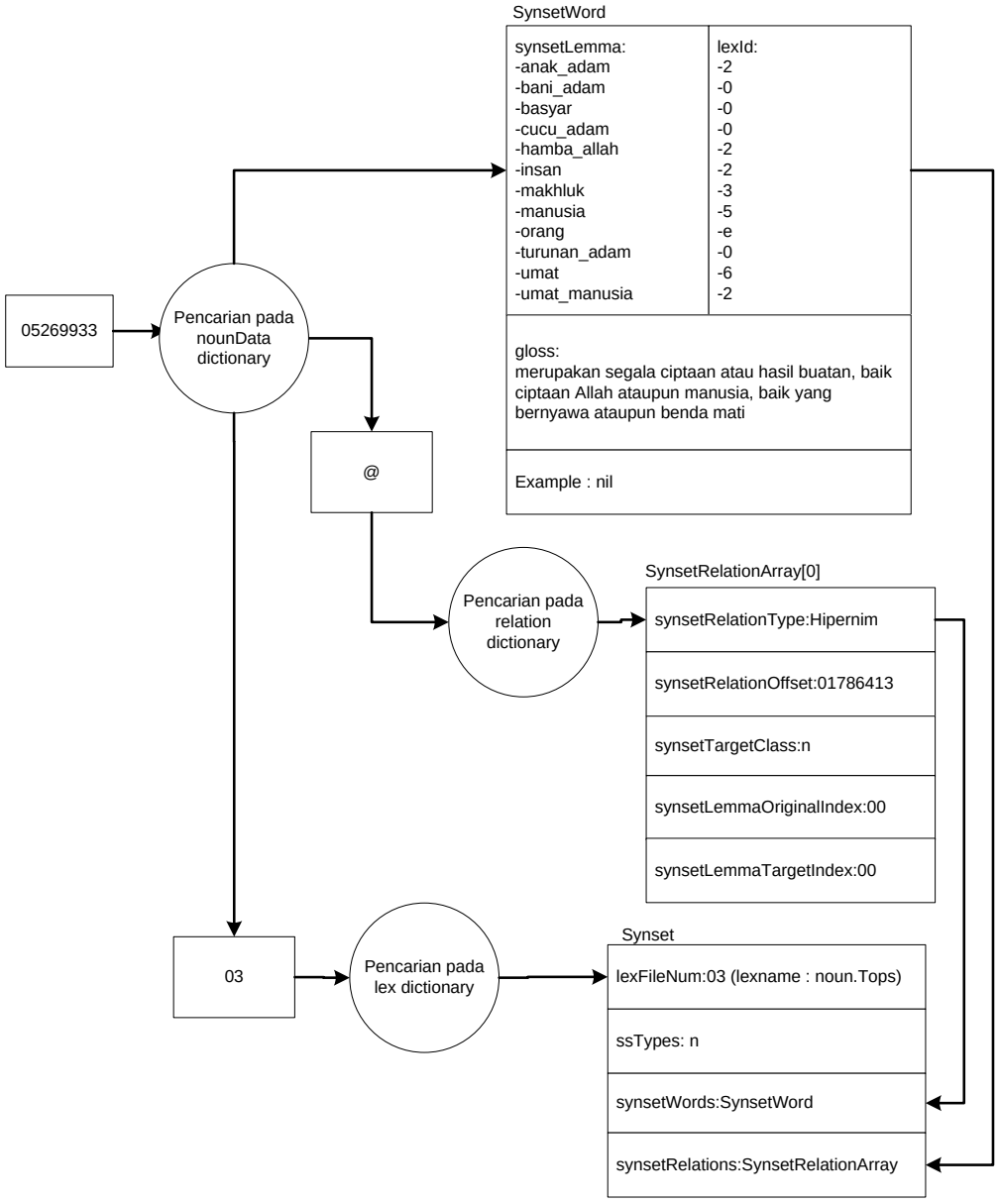
Nama Variabel	Tipe Variabel	Fungsi
gloss	NSString	Menyimpan gloss synset
example	NSString	Menyimpan example
synsetLemma	NSMutableArray	Menyimpan lemma sinonim synset
lexId	NSMutableArray	Menyimpan lexId milik lemma sinonim

Pada tabel 4.6 ditunjukkan bahwa selain menyimpan informasi mengenai sinonim class SynsetWord juga menyimpan informasi mengenai gloss dan example yang dimiliki oleh synset. Class SynsetWord juga mampu mengatasi kondisi dimana apabila synset memiliki lebih dari satu buah sinonim dengan memanfaatkan array untuk menyimpan lemma yang didapatkan.

Tabel 4.7
Struktur Class SynsetRelation

Nama Variabel	Tipe Variabel	Fungsi
synsetRelationType	NSString	Menyimpan tipe relasi
synsetRelationOffset	NSString	Menyimpan synset offset relasi
synsetTargetClass	NSString	Menyimpan class kata relasi
synsetLemmaOriginalIndex	NSString	Menyimpan id lemma asal
synsetLemmaTargetIndex	NSString	Menyimpan id lemma tujuan

Tabel 4.7 menunjukkan variabel synsetLemmaOriginalIndex dan synsetLemmaTargetIndex akan menyimpan index dari lemma asal dan index lemma dari synset pemilik relasi yang saling berhubungan. Apabila relasi bersifat sematik maka kedua variabel akan bernilai 00.



Gambar 4.5
Ilustrasi Ekstraksi Informasi

Proses ekstrasi informasi yang dimiliki oleh synset dimulai dengan melakukan ekstraksi pada data dictionary. Pada gambar 4.5 ditunjukkan proses ekstraksi dari synset 05269933. Synset tersebut akan dicari ke dalam nounData dictionary, hasil dari ekstraksi ini berupa sinonim, gloss, lexFileNum dan relasi yang dimiliki. Sinonim, gloss dan example akan disimpan dengan menggunakan struktur class SynsetWords. Ekstraksi sinonim, gloss dan example ini tidak melibatkan lex dan relation dictionary, berbeda dengan ekstraksi lexFileNum dan ekstraksi relasi. Pada ekstraksi relasi akan didapatkan relation pointer yang dimiliki oleh synset, untuk mengetahui jenis dari relasi tersebut maka perlu dilakukan pencarian pada relation dictionary, hasil pencarian itulah yang kemudian akan disimpan ke dalam variabel synsetRelationType milik class SynsetRelation. Pada ekstraksi relasi terdapat juga ekstraksi index pemilik relasi apabila nlai dari index tidak 00 maka akan terjadi pencarian ke dalam synsetlemma untuk mendapatkan lemma pemilik index. Demikian juga pada ekstraksi lexFileNum, ekstraksi ini hanya mendapatkan kode angka dari lexnames.

Setiap proses yang ditunjukkan pada gambar 4.4 ditangani oleh sebuah prosedur yang disediakan oleh class WordNet. Untuk melakukan ekstraksi pada data dictionary digunakan prosedur getSynsetData, untuk melakukan ekstraksi pada relation dictionary digunakan prosedur getRelationName dan untuk melakukan ekstraksi pada lex dictionary digunakan prosedur getLexName. Penerapan gambar 4.4 ditunjukkan oleh algoritma 4.2.

Algoritma 4.2 menunjukkan algoritma yang digunakan untuk melakukan ekstraksi pada data dictionary. Algoritma ini ditangani oleh sebuah prosedur bernama getSynsetData. Prosedur getSynsetData memerlukan dua buah parameter dengan tipe string. Parameter pertama merupakan synset offset yang ingin diekstrak sedangkan parameter kedua merupakan class kata tempat synset tersebut berada. Prosedur getSynsetData terbagi menjadi tiga tahap, tahap pertama adalah ekstraksi lexFileNum dan sinonim. Tahap kedua adalah ekstraksi gloss dan example, tahap ketiga adalah ekstraksi relasi. Setiap tahap tersebut akan dibahas secara detail disertai dengan contoh data yang dihasilkan.

Algoritma 4.2 Ekstraksi Informasi Pada Data Dictionary

```
[Digunakan sebagai algoritma dasar untuk ekstraksi informasi yang
dimiliki oleh synset offset]

1. [Periksa semua dictionary dan ambil data milik synset offset]
    1.1  FOR EACH dictionary in indexDictionary
         1.1.1   IF SynsetOffset in dictionary
              1.1.1.1  StringData = GET_DATA(SearchKata)

2. [String Data yang didapatkan akan mengalami proses ekstraksi]
    2.1  dataSplit = SPLIT_WITH_DELIMITER(StringData,"|")
    2.2  tempSplit = SPLIT_WITH_DELIMITER(StringData," ")
    2.3  LexFileNum = GET_INDEX(tempSplit,1)
    2.4  Type = GET_INDEX(tempSplit,2)
    2.5  SynonymNum = GET_INDEX(tempSplit,3)
    2.6  RelationIndex= SynonymNum*2+4
    2.7  RelationNum = GET_INDEX(tempSplit,RelationIndex)

3.  [Lakukan ekstraksi lemma synonym]
    3.1 Index=0
    3.2 FOR (i=0;i<SynonymNum;i++)
         3.2.1   Lemmas[i] = GET_INDEX(tempSplit,index+4)
         3.2.2   ID[i] = GET_INDEX(tempSplit,i+5)
         3.2.3   Index+=2

4. [Lakukan ekstraksi gloss dan example]
    4.1 glossSplit = SPLIT_WITH_DELIMITER(stringData,";")
    4.2 FOR (i=0;i<glossSplit Size;i++)
         4.2.1   IF (glossSplit[i] contain "\")
               4.2.1.1   Gloss+=glossSplit[i]
         4.2.2   ELSE
               4.2.2.1   Example+=glossSplit[i]

5. [Lakukan ekstraksi relasi synonym]
    5.1 Index=0
    5.2 FOR (i=0;i<RelationNum;i++)
         5.2.1   tempIndex = index+RelationIndex
         5.2.2   RelationPointer = GET_INDEX(tempSplit, tempIndex
                 +1)
         5.2.3   RelationName = GET_REL_NAME(RelationPointer)
         5.2.4   RelationOffset = GET_INDEX(tempSplit, tempIndex
                 +2)
         5.2.5   OffsetClass = GET_INDEX(tempSplit tempIndex +3)
         5.2.6   RelationTemp = GET_INDEX(tempSplit tempIndex +4)
         5.2.7   RelationOri = SUBSTRING_FROM_TO(RelationTemp,0,1)
         5.2.8   RelationTar = SUBSTRING_FROM_TO(RelationTemp,2,3)
         5.2.9   Index+=4
```

Segmen program 4.5 menunjukkan tahap pertama dari prosedur getSynsetData yaitu ekstraksi sinonim dan lexFileNum. Pada tahap ini digunakan class WordData untuk menampung informasi hasil ekstraksi, selain itu juga disiapkan class Synsets sebagai penampung keseluruhan informasi.

Segmen Program 4.5 Ekstraksi Sinonim & LexFileNum

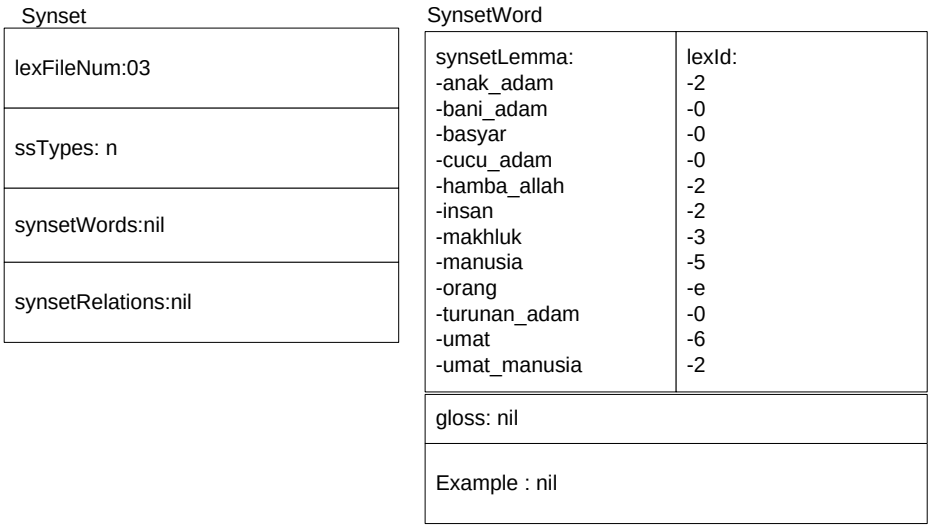
```
1:   id data = [[Synsets alloc]init];
2:   NSString *stringData;

3:   if ([Dict isEqualToString:@"noun"])
4:     stringData = [nounData objectForKey:synsetOffset];
5:   if ([Dict isEqualToString:@"verb"])
6:     stringData = [verbData objectForKey:synsetOffset];
7:   if ([Dict isEqualToString:@"adj"])
8:     stringData = [adjData objectForKey:synsetOffset];
9:   if ([Dict isEqualToString:@"adv"])
10:    stringData = [advData objectForKey:synsetOffset];

11:  NSArray *synsetGlossExampleSplit =
     [stringData componentsSeparatedByString:@"|"];

12:  NSArray *dataSplit = [[synsetGlossExampleSplit
     objectAtIndex:0]componentsSeparatedByString:@" "];

13:  [data setLexFileNum:[dataSplit objectAtIndex:1]];
14:  [data setSsTypes:[dataSplit objectAtIndex:2]];

15:  id wordsData = [[SynsetWords alloc]init];

16:  NSMutableArray *lemmas = [[NSMutableArray alloc]init];
17:  NSMutableArray *lexIds = [[NSMutableArray alloc]init];
18:  int x=0;
19:  NSString *hexStr = [dataSplit objectAtIndex:3];
20:  NSScanner *scanner = [NSScanner scannerWithString:hexStr];
21:  unsigned int outVal ;
22:  [scanner scanHexInt:&outVal];
23:  for(int j=0;j<outVal;j++)
24:  {
25:    [lemmas addObject:[dataSplit objectAtIndex:x+4]];
26:    [lexIds addObject:[dataSplit objectAtIndex:x+5]];
27:    x+=2;
28:  }

29:  [wordsData setLexId:lexIds];
30:  [wordsData setSynsetLemma:lemmas];
```

Pada baris 3-10 dari segmen program 4.5 dilakukan pemeriksaan dari class kata manakah synset offset berasal lalu dilakukan pencarian pada data dictionary yang sesuai dan hasil pencarian disimpan pada variabel stringData. Seteleh data didapatkan maka langkah pertama adalah memisahkan bagian yang mengandung gloss dan example dengan bagian yang mengandung sinonim dan relasi kata. Kedua bagian tersebut dipisahkan oleh sebuah simbol “|” proses ini ditunjukkan pada baris 11. Contoh dari data adalah : *00001740 03 n 01 sepelarian 0 000* |

sejauh berlari tentang jarak . Kumpulan string yang berada seteleh simbol "|" merupakan gloss yang dimiliki oleh kata.

Setelah pemisahan bagian dilakukan maka pertama-tama pengekstrakan sinonim akan dilakukan. Pada baris 12 bagian yang mengandung sinonim dan relasi akan dipisah berdasarkan karakter spasi lalu disimpan ke dalam variabel dataSplit. Seperti yang telah dijelaskan pada bab 2 bahwa setiap field pada data file (dalam Tugas Akhir ini telah dipindahkan ke dalam data dictionary) bahwa setiap field akan dipisahkan berdasarkan karakter spasi. Hasil ekstraksi yang bisa didapatkan adalah kode lexnames dari synset dan class kata dari synset, proses ini ditunjukkan pada baris 13 dan 14. Proses selanjutnya adalah melakukan ekstraksi sinonim yang dimiliki oleh synset beserta id dari tiap lemma sinonim tersebut, id pada kata tersebut melambangkan jumlah kata. Untuk melakukan ekstraksi sinonim pertama-tama perlu diketahui jumlah dari sinonim yang dimiliki, pada baris 29-22 dilakukan ekstraksi jumlah sinonim yang dimiliki (jumlah diwakilkan 2 digit bilangan hexadecimal) serta konversi dari bilangan hexadecimal ke bilangan integer. Konversi ini memanfaatkan variabel dengan tipe NSScanner.

Pada baris 20 scanner akan menerima inputan berupa sebuah string yang merupakan bilangan hexadesimal. Pada baris 22 scanner melakukan konversi dari bilangan hexadesimal dan menyimpannya ke dalam variabel outVal yang bertipe integer. Setelah jumlah dari sinonim telah didapatkan maka langkah selanjutnya adalah mengambil setiap lemma beserta id yang dimiliki lalu menyimpannya ke dalam array. Proses ini ditunjukkan pada baris 23-28, pengambilan lemma sinonim dan id akan berselang 2 field karena posisi lemma dan id bersebelahan misalnya *0c anak_adam 2 bani_adam 0*, 0c adalah jumlah sinonim sedangkan anak_adam dan bani_adam adalah lemma, 2 dan 0 merupakan id dari masing-masing lemma. Pada saat tahap ini selesai dilakukan maka data yang sudah didapatkan adalah lexFileNum, sinonim, dan ssType. Ilustrasi dari kondisi informasi ditunjukkan oleh gambar 4.6 dimana ditampilkan informasi berupa lemma sinonim, id, lexFileNum dan ssTypes yang disimpan oleh class Synset dan SynsetWord. Ditunjukkan pada gambar 4.6 bahwa informasi berupa gloss dan example masih belum didapatkan.



Gambar 4.6
Hasil ekstraksi tahap pertama

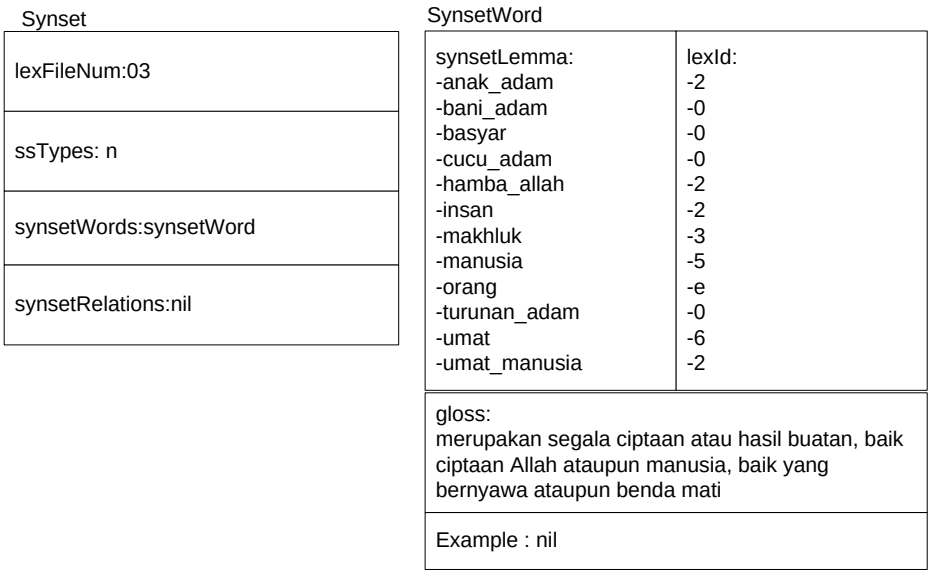
Untuk melengkapi data dari synsetWords maka tahap kedua dari prosedur akan melakukan ekstraksi gloss dan example yang dimiliki oleh synset. Setelah ekstraksi selesai dilakukan maka informasi pada synsetWords telah lengkap dan langkah akhir yang harus dilakukan adalah menyimpan synsetWords ke dalam class Synset. Tahap ketiga dari ekstraksi pada data dictionary baru akan dilaksanakan setelah gloss, example, sinonim telah didapatkan dan disimpan pada class Synset.

Proses ekstraksi yang ditunjukkan oleh segmen program 4.6 adalah ekstraksi gloss dan example apabila tersedia. Pada baris 3 ditunjukkan bahwa bagian yang mengandung gloss dan example akan dipisah berdasarkan simbol “;”. Tetapi sebuah synset dapat memiliki lebih dari satu example dan setiap example tersebut juga dipisahkan oleh simbol “;”. Kemungkinan kedua adalah gloss tersebut terbagi menjadi beberapa bagian dan setiap bagiannya dipisahkan oleh simbol “;” juga. Untuk membedakan apakah suatu field merupakan bagian dari gloss atau bagian dari example maka dilakukan sebuah pemeriksaan, sebuah example selalu diawali dan diakhiri oleh tanda petik (“) oleh karena itu apabila ditemukan tanda petik pada suatu field maka field tersebut merupakan bagian dari example tetapi jika tidak ditemukan maka field tersebut merupakan bagian dari gloss.

Segmen Program 4.6 Ekstraksi Gloss & Example

```
1:   NSString *gloss=[[NSString alloc]init];
2:   NSString *example=[[NSString alloc]init];

3:   NSArray *glossSplit = [[synsetGlossExampleSplit
     objectAtIndex:1]componentsSeparatedByString:@";"];
4:   for(int i=0;i<[glossSplit count];i++)
5:   {
6:    if ([[glossSplit
      objectAtIndex:i]rangeOfString:@"\""].location == NSNotFound)
7:    {
8:      gloss=[gloss stringByAppendingFormat:@"%@ ",[glossSplit
        objectAtIndex:i]];
9:    }
10:   else {
11:     example=
        [example stringByAppendingFormat:@"%@ ",[glossSplit
        objectAtIndex:i]];
12:   }
13:  }
14:  [wordsData setGloss:gloss];
15:  [wordsData setExample:example];

16:  [data setSynsetWords:wordsData];
```

Proses ekstraksi gloss dan example ini ditunjukkan pada baris 3-13 segmen program 4.6. Pada baris 6 dilakukan pemeriksaan apakab tanda petik berada pada field yang sedang diperiksa, pemeriksaan ini memanfaatkan prosedur rangeOfString milik class NSString. Karena sebuah string pada Objective-C harus diawali simbol @” dan diakhiri simbol “ maka untuk melakukan pencarian terhadap tanda petik harus diawali dengan sebuah garis miring (@”\””). Hasil ekstraksi gloss dan example ditampung pada sebuah variabel string dan variabel string tersebut akan disimpan oleh variabel wordData, wordData tersebut pada akirnya akan ditampung oleh variable synsetWords pada class Synset. Setelah tahap ini selesai dilakukan maka sinonim, gloss dan example dari synset telah berhasil didapatkan, ilustrasi kondisi ekstraksi ditunjukkan oleh gambar 4.7. Pada saat tahap kedua selesai dilakukan, informasi yang belum diekstrak hanyalah relasi yang dimiliki oleh synset. Ekstraksi ini akan melibatkan pencarian pada relation dictionary. Hasil dari ekstraksi tahap ketiga adalah array yang mengandung variabel dengan tipe class synsetRelation. Variabel-variabel tersebut akan mengandung informasi mengenai relation pointer, synset pemilik relasi dan juga index lemma pemilik relasi.



Gambar 4.7
Hasil ekstraksi tahap kedua

Detail dari ekstraksi relation dictionary ditunjukkan oleh segmen program 4.8. Relation Pointer akan menjadi parameter yang digunakan untuk melakukan pencarian pada relation dictionary. Jenis relasi yang didapat dari hasil pencarian itulah yang akan disimpan ke dalam variabel synsetRelation. Segmen program 4.7 menunjukkan tahap ketiga ekstraksi.

Proses ekstraksi yang ditunjukkan oleh segmen program 4.7 adalah proses ekstraksi relasi yang dimilki oleh synset, proses ini akan memanfaatkan class SynsetRelation untuk menampung hasil ekstraksi. Untuk melakukan ekstraksi relasi maka terlebih dahulu harus diketahui jumlah relasi yang dimiliki oleh synset, jumlah relasi dilambangkan dengan tiga digit bilangan integer. Pada baris 3 ditunjukkan sebuah variabel *y* dengan nilai (outVal*2+4) . Variabel outVal dikalikan dengan 2 karena outVal merupakan jumlah sinonim yang dimiliki sedangkan setiap sinonim terdiri dari lemma dan LexId. Sedangkan ditambahkan dengan empat karena index array yang mengandung sinonim dan relasi dimulai dari index keempat dari array dataSplit. Variabel y ini merupakan index field yang mengandung jumlah relasi yang dimiliki, jika tidak memiliki relasi apapun maka field ini akan bernilai 0. Pada baris 7-9 dilakukan ekstraksi tipe relasi yang

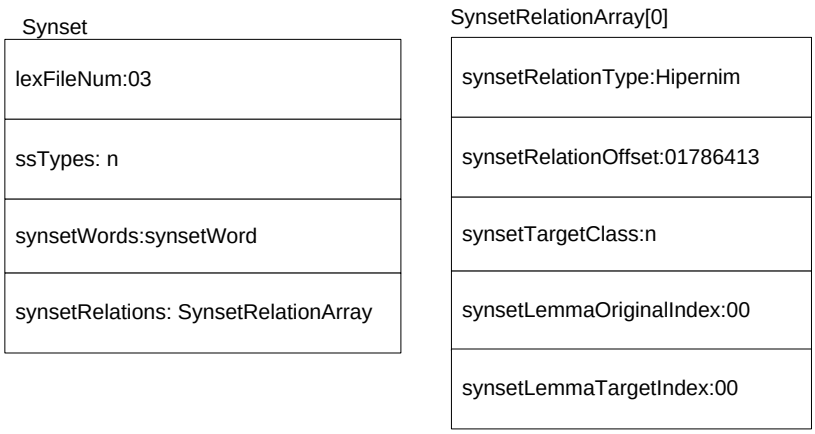
dimilki, synset offset pemilik relasi tersebut dan class kata dari synset tersebut. Sedangkan pada baris 10-14 dilakukan ekstraksi index asal dan target lemma. Apabila field ini bernilai 0000 berarti relasi bersifat semantik dan semua lemma dari synset asal dan tujuan memiliki relasi ini. Tetapi bila bernilai selain 0000 berarti relasi bersifat lexical dimana relasi ini hanya dimiliki oleh salah satu lemma dalam synset asal dengan salah satu lemma pada target synset. Lemma pertama pada synset memiliki index satu. Ekstraksi relasi mengakhiri proses ekstraksi pada data dictionary, kondisi informasi yang telah diekstrak ditunjukkan oleh gambar 4.8.

Segmen Program 4.7 Ekstraksi Relasi Synset

```
1:   NSMutableArray *relations = [[NSMutableArray alloc]init];
2:   x=0;
3:   int y=(outVal*2+4);

4:   for(int j=0;j<[[dataSplit objectAtIndex:y]intValue];j++)
5:   {
6:     SynsetRelations  *synsetRelation = [[SynsetRelations
       alloc]init];

7:     [synsetRelation setSynsetRelationType:[self
       getRelationName:[dataSplit objectAtIndex:x+y+1]]];
8:     [synsetRelation setSynsetRelationOffset:[dataSplit
       objectAtIndex:x+y+2]];
9:     [synsetRelation setSynsetTargetClass:[dataSplit
       objectAtIndex:x+y+3]];

10:    NSString *ori, *tar;
11:    ori=[[dataSplit objectAtIndex:x+y+4] substringToIndex:2];
12:    tar = [[dataSplit objectAtIndex:x+y+4]
       substringWithRange:NSMakeRange(2,2)];
13:    [synsetRelation setSynsetLemmaOriginalIndex:ori];
14:    [synsetRelation setSynsetLemmaTargetIndex:tar];

15:    [relations addObject:synsetRelation];
16:    x+=4;
17:  }
18:  [data setSynsetRelations:relations];
19:  return data;
```

Pada gambar 4.8 ditunjukkan salah satu relasi milik synset yang merupakan hasil ekstraks tahap ketiga. Array yang menampung relasi-relasi hasil ekstraksi akan disimpan ke dalam variabel synsetRelations milik class Synset. Pada tahap ini semua relasi milik synset telah berhasil diekstrak. Untuk lexname

milik synset akan disimpan dalam bentuk lexFileNum, apabila user memerlukan lexname synset maka user dapat memanfaatkan prosedur getSynsetName milik class WordNet.



Gambar 4.8
Hasil ekstraksi tahap ketiga

Pada baris 7 segmen program 4.7 ditunjukkan bahwa prosedur getSynsetData memanfaatkan prosedur getRelationName untuk mengetahui jenis relasi yang dimiliki. Prosedur tersebut memanfaatkan relation pointer yang didapatkan untuk melakukan pencarian pada relation dictionary. Segmen program 4.8 menunjukkan detail dari prosedur getRelationName.

Segmen Program 4.8 Prosedur getRelationName

```
1: -(id) getRelationName:(NSString   *) relID
2: {
3:  NSString *relName = [[NSString alloc]init];
4:  relName = [relation objectForKey:relID];
5:  return   relName;
6: }
```

Prosedur getRelationName merupakan sebuah prosedur bersifat public yang berperan untuk mendapatkan jenis relasi berdasarkan relation pointer yang diinputkan. Input dari prosedur ini berupa string yang merupakan relation pointer dan output dari prosedur ini berupa sebuah string yang merupakan jenis relasi dari relation pointer yang diinputkan. Baris 4 dari segmen program 4.7 menunjukkan

proses mendapatkan value dari dictionary dimana key yang digunakan adalah input yang diterima. Pada baris 5 value tersebut akan diberikan pada user.

Untuk melakukan ekstraksi lexname maka class WordNet menyediakan prosedur getLexName. Prosedur getLexName merupakan sebuah prosedur bersifat public dan berperan untuk mendapatkan lexnames dari suatu lexId tertentu. Input dari prosedur ini berupa string yang merupakan lexId dan output dari prosedur ini berupa sebuah string yang merupakan lexname dari lexId yang diinputkan. Segmen program 4.9 menunjukkan detail dari prosedur getLexName.

Segmen Program 4.9 Prosedur getLexName

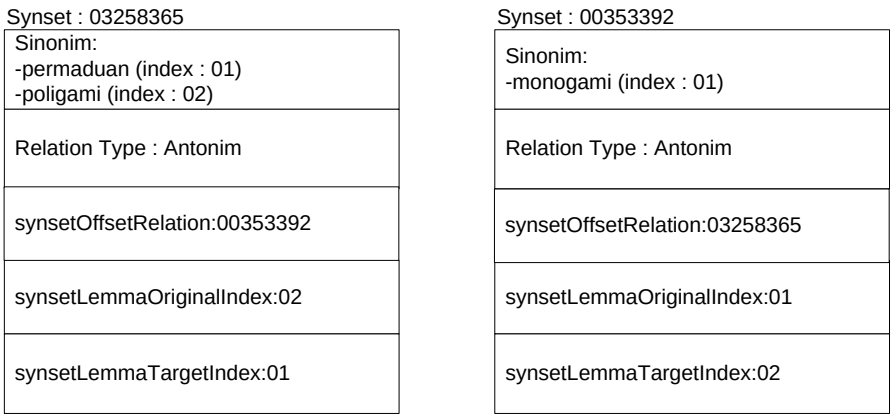
```
1: -(id) getLexName:(NSString *) lexId
2: {
3:  NSString *temp = [[NSString alloc]init];
4:  temp=[lex objectForKey:lexId];
5:  NSArray *lexSplit = [temp componentsSeparatedByString:@"\t"];
6:  return [lexSplit objectAtIndex:1];
7: }
```

Prosedur getLexName akan mengakses lex dictionary dan mengambil value yang dimiliki oleh dengan nilai lexId, proses ini ditunjukkan pada baris 4 segmen program 4.9. Setelah mendapatkan value tersebut maka dilakukan pemotongan string berdasarkan tab atau “\t” dab disimpan pada sebuah array lexSplit. Prosedur akan mengembalikan value yang berada pada index 1 dari array. Index pertama dari array akan mengandung lexId dari lexname tersebut, lexId ini bernilai sama dengan lexId yang diinputkan. Format dari file lexnames dijelaskan secara detail pada bab II. Proses pemotongan string dan pengembalian data ditunjukkan oleh baris 5 dan 6.

4.2.3 Mendapatkan Lemma Sinonim pada Index Tertentu

Apabila suatu synset memiliki hubungan lexical maka perlu diketahui lemma sinonim yang memiliki relasi tersebut. Dalam suatu relasi semantik tidak semua lemma pada sinonim mengandung hubungan tersebut hanya suatu lemma tertentu, hal ini juga berlaku pada synset tujuan dari relasi. Pada gambar 4.9 ditunjukkan relasi antonim antara synset 03258365 dengan synset 00353392.

Synset 03258365 memiliki dua buah sinonim tetapi karena relasi semantik hanya dimiliki oleh salah satu lemma maka perlu diketahui lemma yang dimaksud.



Gambar 4.9
Relasi Leksikal

Untuk mengatasi hal ini maka class WordNet menyediakan sebuah prosedur bernama getSynsetLemma untuk mendapatkan lemma yang memiliki relasi tersebut, index lemma didapatkan pada saat ekstraksi relasi dan tersimpan pada class SynsetRelation. Prosedur getSynsetLemma merupakan sebuah prosedur bersifat public dan berperan untuk mengambil sebuah lemma pada index tertentu yang dimiliki oleh sebuah synset. Prosedur ini digunakan ketika user ingin mengetahui kata yang memiliki suatu relasi tertentu dengan kata yang sedang dicari. Apabila field source/target dari suatu relasi bernilai 0000 maka prosedur ini akan mengambil lemma sinonim pertama dari synset tetapi bila field tersebut memiliki index tertentu maka prosedur akan mengambil lemma pada posisi yang dimaksud. Segmen program 4.10 menunjukkan detail dari prosedur getSynsetLemma.

Input dari prosedur ini berupa synset offset, class kata tempat synset berada dan index lemma yang diinginkan. Output yang dihasilkan adalah sebuah string yang merupakan lemma sinonim yang berada pada yang diinginkan pada suatu synset. Prosedur getSynsetLemma memiliki kemiripan proses dengan prosedur getSynsetData hanya saja proses yang dijalankan tidak sebanyak proseudr getSynsetData.

Segmen Program 4.10 Prosedur getSynsetLemma

```
1:  -(id) getSynsetLemma:(NSString *) synsetOffset
          forDict:(NSString *)Dict forIndex:(NSString *)index;
2:  {
3:   id data = [[NSString alloc]init];
4:   NSString *stringData;

5:   if ([Dict isEqualToString:@"noun"])
6:    stringData = [nounData objectForKey:synsetOffset];
7:   if ([Dict isEqualToString:@"verb"])
8:    stringData = [verbData objectForKey:synsetOffset];
9:   if ([Dict isEqualToString:@"adj"])
10:   stringData = [adjData objectForKey:synsetOffset];
11:  if ([Dict isEqualToString:@"adv"])
12:   stringData = [advData objectForKey:synsetOffset];

13:  NSArray *synsetGlossExampleSplit = [stringData
     componentsSeparatedByString:@"|"];
14:  NSArray *dataSplit =[[synsetGlossExampleSplit
     objectAtIndex:0]componentsSeparatedByString:@" "];

15:  int x=0;
16:  NSString *hexStr = [dataSplit objectAtIndex:3];
17:  NSScanner *scanner = [NSScanner scannerWithString:hexStr];
18:  unsigned int outVal ;
19:  [scanner scanHexInt:&outVal];

20:  if([index isEqualToString:@"00"])
21:   index=@"01";

22:  for(int j=0;j<outVal;j++)
23:  {
24:   if (j+1 == [index intValue])
25:    data=[[dataSplit objectAtIndex:x+4]retain];
26:   x+=2;
27:  }
28:  return data;
29: }
```

Baris 4-19 segmen program 4.10 juga diilakukan pada prosedur getSynsetData. Perbedaan dari kedua prosedur dimulai dari baris 20, proses-proses tersebut hanya dilakukan oleh prosedur getSynsetLemma. Baris 20 merupakan proses pemeriksaan apakah index yang diberikan bernilai 00, jika benar maka prosedur akan menetapkan bahwa index yang akan diambil adalah index yang pertama. Pada baris 22 dilakukan iterasi sejumlah sinonim yang dimiliki oleh synset, pada baris 24 dilakukan pengecekan apakah iterasi berada pada index yang diinginkan jika benar maka lemma sinonim pada posisi tersebut akan diambil.

4.3 Mendapatkan Daftar Kata pada Lexical Database Files

Daftar kata yang terdapat pada Lexical Database Files dapat didapatkan dengan memanfaatkan kelompok dictionary. Daftar kata yang didapatkan tersebut akan menjadi daftar kata yang ditampilkan oleh browser pada halaman search serta untuk menjalankan fitur auto complete. Karena proses pengurutan daftar kata memakan waktu yang cukup lama maka hasil daftar kata yang didapatkan akan disimpan ke dalam sebuah plist. Segmen program 4.15 menunjukkan proses yang terjadi.

Segmen Program 4.11 Proses Mendapatkan Daftar kata

```
1:  NSFileManager *fileManager = [NSFileManager defaultManager];

2:  if(![fileManager fileExistsAtPath:listPlistFilePath])
3:  {
4:   listOfAvailableWords = [[NSMutableArray alloc]init];
5:   NSMutableDictionary *tempList = [[[NSMutableDictionary
     alloc]init]retain];

6:   for(NSString *str in noun)
7:     [tempList setObject:str forKey:str];
8:   for(NSString *str in verb)
9:     [tempList setObject:str forKey:str];
10:  for(NSString *str in adj)
11:    [tempList setObject:str forKey:str];
12:  for(NSString *str in adv)
13:    [tempList setObject:str forKey:str];

14:  for(NSString *str in tempList)
15:  {
16:   str=[str stringByReplacingOccurrencesOfString:@"_"
      withString:@" "];
17:   [listOfAvailableWords addObject:str];
18:  }

19:  NSArray *sorted = [listOfAvailableWords
     sortedArrayUsingSelector:@selector(compare:)];

20:  listOfAvailableWords = [sorted mutableCopy];

21:  [tempList release];

22:  [listOfAvailableWords writeToFile:listPlistFilePath
     atomically:YES];
23: }
24: else {
25:  listOfAvailableWords     = [[NSMutableArray
     alloc]initWithContentsOfFile:listPlistFilePath];
26: }
```

Pada baris 1 dan 2 segmen program 4.11 ditunjukkan bahwa pada proses mendapatkan daftar kata melakukan pemeriksaan terhadap sebuah file plist bernama listPlist. File dengan ekstensi plist adalah sebuah text file yang digunakan oleh Objective-C, struktu penyimpanan file ini berbentuk sebuah xml. File listPlist berperan menampung daftar kata yang menjadi hasil proses ini. Apabila file tersebut tidak ditemukan pada folder document milik device berarti proses mendapatkan daftar kata belum pernah terjadi, tetapi apabila file tersebut ditemukan maka class WordNet akan melakukan pembacaan daftar kata dari file plist tersebut. Proses pembacaan daftar kata dari file plist ditunjukkan oleh baris 25. Tetapi bila file plist tidak ditemukan proses akan berlanjut pada baris 4. Pada baris 5 disiapkan sebuah dictionary yang akan menapung semua kata yang terdapat pada keempat index dictionary.

Proses pembacaan keempat dictionary ditunjukkan pada baris 6-13, setiap index dictionary akan mengalami iterasi dan key pada setiap index dictionary akan menjadi key dan value pada dictionary penampung daftar kata. Pada tahap ini digunakan dictionary untuk menampung daftar kata agar kata-kata yang terdapat pada lebih dari satu index dictionary dapat dieliminasi secara otomatis. Setelah daftar kata yang berada pada keempat index dictionary didapatkan maka langkah selanjutnya adalah melakukan penggantian string underscore (_) menjadi string spasi ( ), hal ini dilakukan karena daftar kata ini akan disajikan oleh WordNet Browser pada user sebagai bagian dari fitur auto complete. String yang mengalami proses penggantian tersebut akan disimpan ke dalam sebuah array, karena key dan value pada dictionary penampung memiliki nilai yang sama maka hanya key yang akan diproses dan disimpan ke dalam array. Proses tersebut ditunjukkan pada baris 14-18, dimana pada baris 16 dilakukan proses penggantian string dan pada baris 17 dilakukan penyimpanan ke dalam sebuah array.

Kata yang tersimpan pada array masih belum terurut sesuai dengan abjad karena itu dilakukan proses pengurutan daftar kata. Proses pengurutan ini memanfaatkan sebuah prosedur milik Objective-C dimana kata dalam array akan diurutkan sesuai abjad. Prosedur tersebut ditunjukkan oleh baris 19. Langkah terakhir yang perlu dilakukan adalah menyimpan daftar kata tersebut ke dalam

sebuah plist. Dengan menyimpan daftar kata ke dalam plist maka ketika class WordNet melakukan inisialisasi lagi proses mendapatkan daftar kata tidak perlu dilakukan cukup dengan melakukan pembacaan plist, hal ini akan mempercepat waktu inisialisasi dari class WordNet.

BAB V
EKSTRAKSI INFORMASI OLEH WORDNET BROWSER

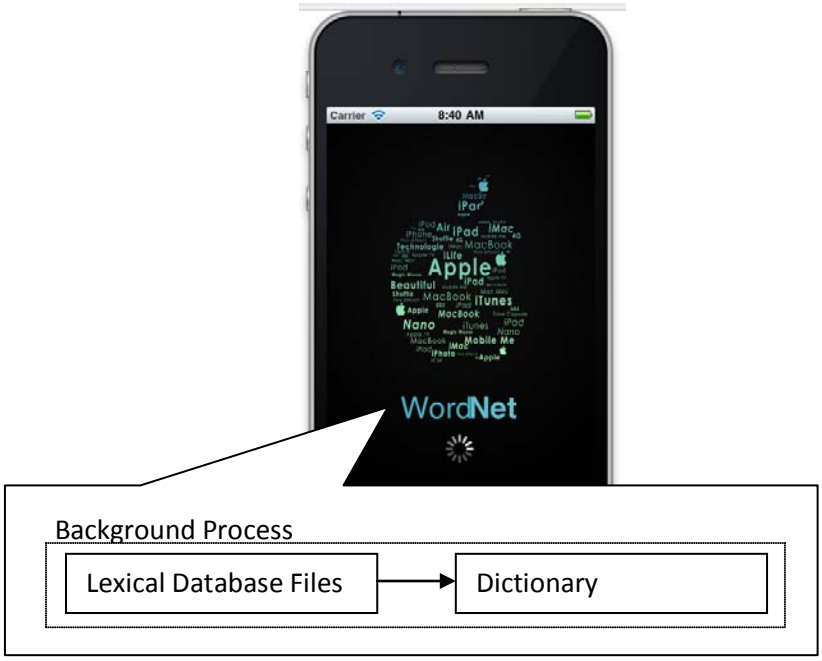
Pada bab ini akan dibahas mengenai ekstraksi informasi yang dilakukan oleh WordNet Browser. Proses ekstraksi informasi akan dilakukan dengan memanfaatkan class WordNet. Proses ekstraksi serta pemersiapan class WordNet akan dibahas secara mendetail pada bab ini. Selain itu akan dijelaskan juga mengenai proses membuat tampilan dengan menggunakan HTML, CSS dan Javascript. Penjelasan tersebut akan disertai dengan contoh-contoh gambar yang dihasilkan dari tahap-tahap pengolahan informasi dan pengaplikasian css. Selain penjelasan mengenai proses ekstraksi informasi, akan dijelaskan juga mengenai fitur-fitur yang membantu user dalam menginputkan kata dan melihat informasi yang dimiliki oleh kata tersebut. Seperti fitur auto complete. Penjelasan-penjelasan yang disediakan akan meliputi arsitektur sistem, segmen program yang melakukan proses tersebut, serta contoh data yang dihasilkan dan ilustrasi proses yang dilakukan.

5.1 Halaman Loading

Halaman Loading merupakan halaman yang muncul pertama kali ketika browser dijalankan. Pada halaman ini class WordNet akan mengalami proses inisialisasi dimana preprocessing dari Lexical Database Files akan dilakukan. Tahap inisialisasi ini dijalankan pada background process.

Proses inisialisasi dari class WordNet diletakkan pada background proses untuk mencegah terjadinya asumsi bahwa program tersebut berhenti bekerja atau mengalami crash. Ketika proses inisialisasi dilakukan, device akan menggunakan memory yang dimiliki untuk melakukan pre-processing Lexical Database. Pada saat proses tersebut berjalan maka device tidak bisa melakukan proses lain, termasuk menampilkan animasi loading yang menunjukkan program berjalan.

Untuk mengatasi hal tersebut maka proses inisialisasi diletakkan pada background process.



Gambar 5.1
Proses Insialisasi pada Halaman Loading

Untuk meletakkan sebuah proses pada background maka terdapat dua buah class yang perlu digunakan. Kedua class tersebut disediakan oleh Objective-C pada library UIKit yang merupakan library standar. Tabel 5.1 menunjukkan deskripsi dari kedua class tersebut.

Tabel 5.1
Class Untuk Meletakkan Proses pada Background

Variabel	Fungsi
NSOperationQueue	Meletakkan operation pada background process.
NSInvocationOperation	Menampung prosedur yang akan dijalankan oleh operation.

Kedua class tersebut akan digunakan untuk meletakkan sebuah prosedur yang berisi proses inisialisasi class WordNet pada background process. Penggunaan dari kedua class tersebut ditunjukkan pada segmen program 5.1.

Segmen Program 5.1 Meletakkan Pada Bakcground Process

```
1:  queue = [[NSOperationQueue alloc]init];
2:  NSInvocationOperation *op = [[NSInvocationOperation
    alloc]initWithTarget:self selector:@selector(processing)
    object:nil ];
3:  [queue addOperation:op];
4:  [op release];
```

Baris 2 segmen program 5.1 menunjukkan deklarasi sebuah variabel bernama op dengan tipe NSInvocationOperation. Variabel ini menampung sebuah prosedur bernama processing, prosedur tersebut berisi inisialisasi dari class WordNet dan beberapa variabel lain yang diperlukan. Pada baris 4 operation tersebut dimasukan ke dalam variabel queue yang bertipe NSOperationQueue. Ketika operation tersebut masuk ke dalam queue maka prosedur yang ditampung oleh operation akan berjalan pada background.

Prosedur processing sendiri berperan sebagai prosedur tempat dilakukannya inisialisasi dan sebagai pemberhenti proses yang berjalan pada background. Ketika semua proses pada background telah selesai dijalankan maka perlu dilakukan spesifikasi proses yang akan dilakukan selanjutnya. Detail prosedur processing ditunjukkan pada segmen program 5.2.

Segmen Program 5.2 Prosedur Processing

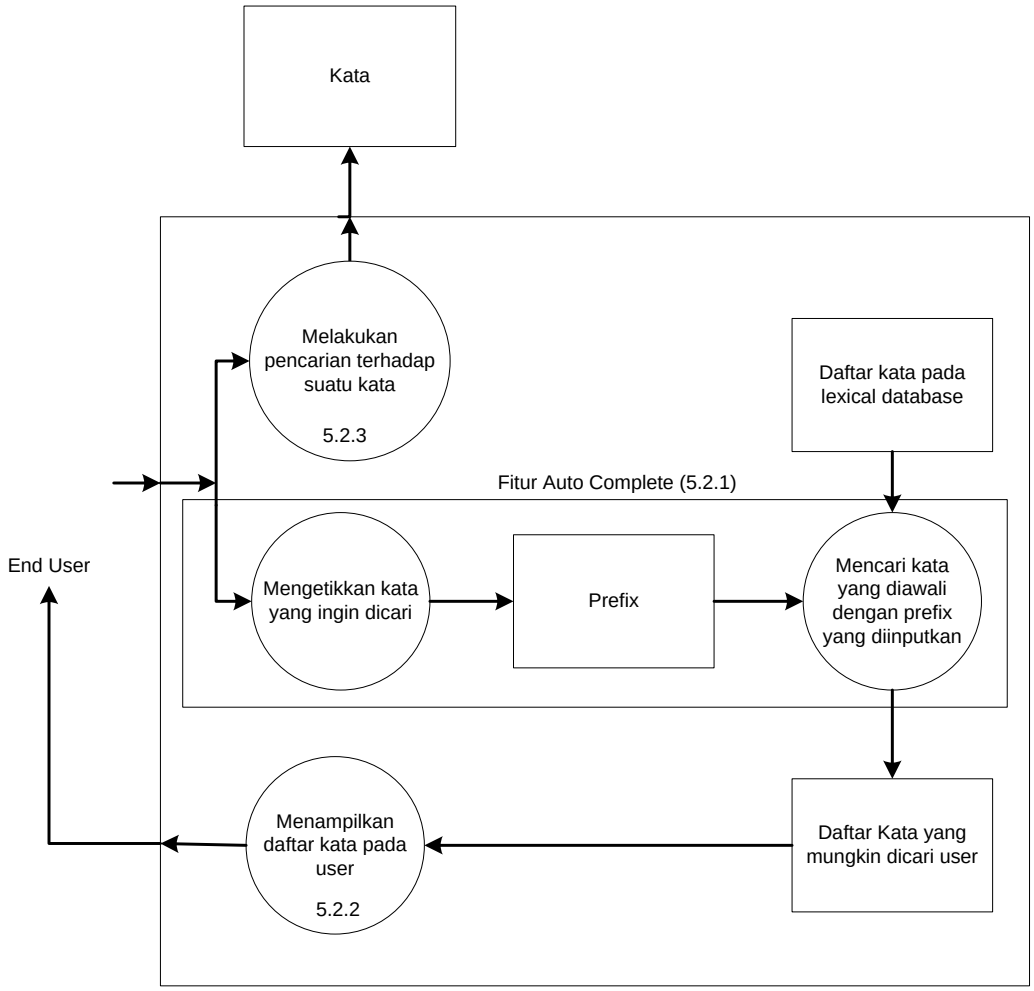
```
1:  WN = [[WordNet alloc]init];

2:  listOfBookmarkedWords = [self readFromFile:[[self
    documentPath]
    stringByAppendingPathComponent:@"/bookmark.plist"]];

3:  historyList = [self readFromFile:[[self documentPath]
    stringByAppendingPathComponent:@"/history.plist"]];

4:  [self performSelectorOnMainThread:@selector(endPreProcessing)
    withObject:nil waitUntilDone:YES];
```

Baris 1 segmen program 5.2 menunjukkan proses inisialisasi class WordNet. Pada tahap ini Lexical Database akan dibentuk menjadi dictionary. Setelah proses inisialisasi class WordNet selesai maka pada baris 2 dan 3 dilakukan juga inisialisasi untuk variabel yang lain. Pada baris 4 dilakukan penghentian proses yang berada pada background. Ketika semua proses inisialisasi telah selesai dilakukan maka prosedur endPreProcessing akan dipanggil, agar aplikasi menunggu semua proses inisialisasi selesai dilakukan maka parameter waitUntilDone harus bernilai YES. Pada aplikasi ini prosedur endPreProcessing akan berisi perpindahan menuju halaman search. Ketika perpindahan ini dilakukan class WordNet telah selesai melakukan pre processing Lexical Database dan siap untuk melakukan ekstraksi informasi milik kata yang diinputkan.

5.2 Halaman Search

Halaman Search merupakan halaman dimana user dapat melakukan pencarian terhadap suatu kata. Pada halaman ini user dapat melakukan pencarian dengan mengetikkan kata yang ingin dicari ke dalam textbox atau memilih daftar kata yang ditampilkan oleh browser.

Pada gambar 5.2 ditunjukkan bahwa browser akan menampilkan daftar kata kepada user. Daftar kata ini didapatkan dari ekstraksi Lexical Database, apabila fitur auto complete diaktifkan maka daftar kata tersebut akan mengalami penyaringan. Hanya kata yang memiliki prefix yang sesuai dengan yang diketikkan oleh user lah yang akan ditampilkan. Tetapi ketika user belum mengetikkan apapun maka daftar kata secara lengkap akan ditampilkan. Input dari sistem berupa kata yang diketikkan oleh user. Kata tersebut akan mengalami pemeriksaan dan menghasilkan output berupa daftar kata yang mungkin ingin diketikkan oleh user. Tetapi jika user menekan tombol search atau memilih salah satu kata dari daftar kata yang ditampilkan maka output yang dihasilkan akan berupa kata yang dipilih oleh user. Kata tersebut nantinya akan diproses menggunakan class WordNet untuk didapatkan synsetnya dan dikirim kepada halaman result.

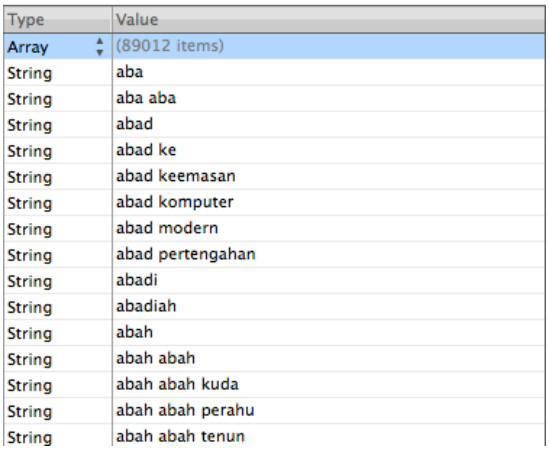


Gambar 5.2
Arsitektur Sistem Halaman Search

Pada gambar 5.2 ditunjukkan bahwa terdapat beberapa proses yang dilakukan pada halaman search. Setiap proses tersebut akan dibahas secara mendetail pada subbab berikutnya.

5.2.1 Fitur Auto Complete

Fitur auto complete merupakan sebuah fitur dimana browser akan menampilkan daftar kata yang mungkin ingin dicari oleh user. Fitur ini akan memanfaatkan daftar kata Lexical Database yang didapatkan dari class WordNet. Daftar kata yang diterima pada tahap ini sudah berupa array yang mengandung seluruh lemma pada Lexical Database seperti yang ditunjukkan pada gambar 5.3.



Type	Value
Array	(89012 items)
String	aba
String	aba aba
String	abad
String	abad ke
String	abad keemasan
String	abad komputer
String	abad modern
String	abad pertengahan
String	abadi
String	abadiah
String	abah
String	abah abah
String	abah abah kuda
String	abah abah perahu
String	abah abah tenun

Gambar 5.3
Daftar Kata

Daftar kata tersebut kemudian akan mengalami pemeriksaan, kata-kata yang tidak memiliki prefix yang sesuai akan dibuang. Pemeriksaan ini ditunjukkan oleh segmen program 5.3.

Segmen Program 5.3 Pemeriksaan Prefix

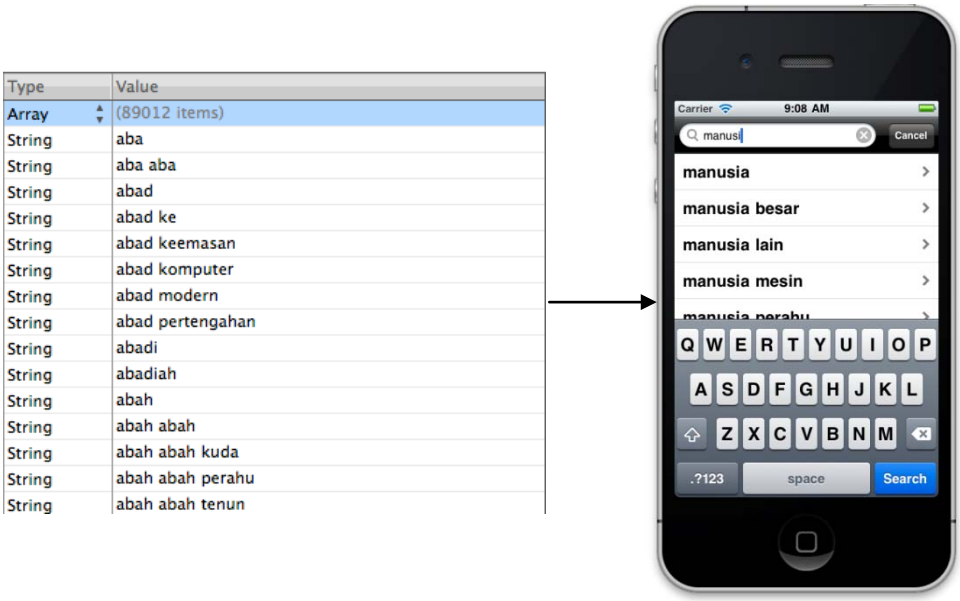
```
1:   [searchResult removeAllObjects];
2:   NSMutableArray *tempList = [[NSMutableArray alloc]init];
3:
4:   NSString *tempWord = [localSearchBar text];
5:   tempWord = [tempWord lowercaseString];

6:   tempList = [WN getListOfAvailableWords];
7:   if ([tempWord length]>0 && [setting autoComplete])
8:   {
9:    for(NSString *str in tempList)
10:   {
11:    if ([str hasPrefix:tempWord])
12:    {
13:     [searchResult addObject:str];
14:    }
15:   }
16:  }
17:  else {
18:   searchResult = [tempList mutableCopy];
19:  }

20:  [xtableView reloadData];
```

Baris 2 segmen program 5.3 menunjukkan sebuah variabel bernama searchResult melakukan pengosongan isi variabel. Variabel searchResult ini bertipe NSMutableArray dan digunakan untuk menampung kata-kata yang akan

disajikan kepada user. Selain variable searchResult proses ini juga menggunakan dua buah variabel lain yaitu variabel tempList, dan tempWord. Variabel tempWord berperan untuk menampung kata yang diketikkan user pada textbox dimana kata tersebut akan mengalami proses lowercaseString, proses pengambilan kata tersebut ditunjukkan pada baris 4 dan 5. Variable tempList berperan untuk menampung daftar kata yang terdapat pada Lexical Database Files, disinilah class WordNet akan berperan. Class WordNet akan memberikan daftar kata yang terdapat pada Lexical Database kepada variabel tempList, proses ini ditunjukkan pada baris 6.

Pada baris 7-19 ditunjukkan proses yang dilakukan untuk memilah kata yang sesuai dengan yang diketikkan oleh user. Pertama akan dilakukan pemeriksaan apakah user sudah melakukan pengetikkan pada textbox dan apakah user menginginkan fitur auto complete untuk dijalankan, apa bila salah satu kondisi tidak terpenuhi maka program akan berlanjut ke baris 18 dimana daftar kata yang didapat dari class WordNet akan langsung diambil untuk ditampilkan tanpa melalui proses pemeriksaan prefix. Tetapi apabila kedua kondisi terpenuhi maka program akan melanjutkan ke baris 7 dimana program akan melakukan iterasi pada semua kata pada variable tempList. Setiap kata yang mengalami iterasi akan mengalami pemeriksaan apakah prefix dari kata tersebut sesuai dengan yang diketikkan oleh user jika ya maka kata tersebut akan dimasukkan ke dalam variabel searchResult. Setelah iterasi selesai dilakukan maka isi dari table view akan diperbaharui dengan menuliskan perintah pada baris 20. Variabel searchResult yang menampung daftar kata yang didapatkan akan digunakan untuk memproses cell pada table view untuk ditampilkan kepada user. Selain itu array tersebut juga akan digunakan untuk memperbaharui tampilan sesuai dengan daftar kata yang didapatkan.

Proses memperbaharui tampilan akan melibatkan beberapa prosedur pada Objective-C. Prosedur-prosedur tersebut memiliki fungsi yang khusus dan saling berhubungan satu sama lain. Detail dari prosedur-prosedur yang digunakan serta proses yang dilakukan ketika pembaharuan tampilan dilakukan akan dijelaskan pada subbab berikutnya.



Gambar 5.4
Daftar Kata Hasil Filter

Hasil dari pencarian ini ditunjukkan oleh gambar 5.4 dimana daftar kata yang dihasilkan tidak semua nya ditampilkan, melainkan hanya yang memiliki prefix yang sesuai. Pada gambar 5.4 ditunjukkan bahwa hanya kata dengan prefix manusi sajalah yang akan ditampilkan.

5.2.2 Menampilkan Daftar Kata kepada User

Untuk menampilkan daftar kata menggunakan sebuah table view maka terdapat dua buah prosedur yang harus dioverride. Kedua prosedur tersebut merupakan prosedur yang disediakan oleh Objective-C. Kedua prosedur tersebut ditunjukkan oleh tabel 5.2.

Tabel 5.2
Prosedur Pengisian Data pada Table View

Nama Prosedur	Fungsi
cellForRowAtIndexPath	Membangung sebuah cell pada table view
numberOfRowsInSection	Menentukan jumlah cell yang dimiliki oleh table view

Kedua prosedur tersebut saling terkait satu sama lain. Nilai yang dihasilkan oleh prosedur numberOfRowsInSection akan menentukan jumlah pemanggilan dari prosedur cellForRowAtIndexPath. Karena jumlah cell akan sesuai dengan jumlah kata yang ditampilkan maka nilai yang akan dikembalikan oleh prosedur numberOfRowsInSection adalah ukuran array yang menampung daftar kata, proses ini ditunjukkan oleh segmen program 5.4.

Segmen Program 5.4 Prosedur numberOfRowsInSection

```
1: -(NSInteger)tableView:(UITableView *)tableView
   numberOfRowsInSection:(NSInteger)section {
2:   return [searchResult count];
3: }
```

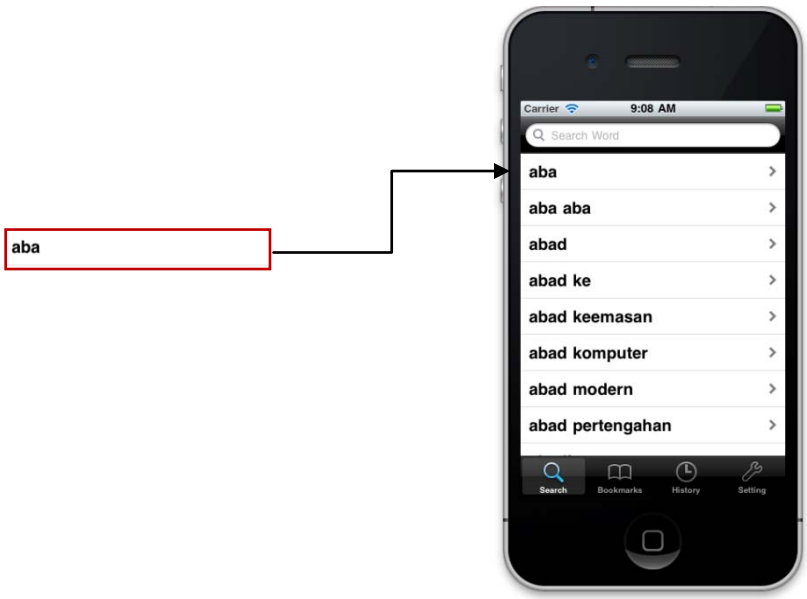
Setelah jumlah cell didapatkan maka langkah selanjutnya adalah membuat cell sesuai dengan jumlahnya. Setiap cell akan mengandung kata yang akan ditampilkan kepada user, pengisian cell ditunjukkan oleh segmen program 5.5.

Segmen program 5.5 Prosedur cellForRowAtIndexPath

```
1:  (UITableViewCell *)tableView:(UITableView *)tableView
2:         cellForRowAtIndexPath:(NSIndexPath *)indexPath{

3:   static NSString *cellIdentifier = @"Cell";
4:   UITableViewCell *cell = [tableView
     dequeueReusableCellWithIdentifier:cellIdentifier];

5:   if (cell == nil)
6:   {
7:    cell = [[[UITableViewCell alloc]
      initWithStyle:UITableViewCellStyleDefault
      reuseIdentifier:cellIdentifier ] autorelease];
8:   }

9:   NSString *cellValue = [searchResult
     objectAtIndex:indexPath.row];
10:  cell.textLabel.text = cellValue;
11:  cell.accessoryType =
     UITableViewCellAccessoryDisclosureIndicator;

12:  return cell;
13: }
```

Pada baris 4 segmen program 5.5 dideklarasikan sebuah UITableViewCell dan dan cell tersebut akan menggunakan memory dari cell yang memiliki

identifier “Cell”. Apabila ternyata belum terdapat cell dengan identifier tersebut maka program akan mengalokasikan memory untuk sebuah cell cell baru dengan identifier “Cell”, proses ini ditunjukkan oleh baris 5-8. Setelah cell telah terbentuk maka langkah selanjutnya adalah mengisi cell tersebut, pada baris 9 program akan mengambil isi dari array searchResult pada index indexPath.row. Variabel indexPath menyimpan nilai berupa posisi baris dan section dimana posisi baris merupakan jumlah iterasi yang telah dijalankan. Pada baris 10 text dari cell akan diisi dengan value dari searchResult dan pada baris 11 accessory dari cell akan diisi dengan tanda panah.



Gambar 5.5
Cell Hasil Pemanggilan Prosedur

Hasil dari prosedur ini adalah sebuah cell dengan sebuah kata didalamnya, setiap pemanggilan prosedur akan menghasilkan sebuah cell. Seperti yang ditunjukkan oleh gambar 5.5 dimana iterasi pertama dari prosedur akan menghasilkan cell dengan kata aba. Setelah mencapai kata abad pertengahan maka kata selanjutnya tidak akan ditampilkan oleh table view. Kata-kata tersebut baru akan diproses dan ditampilkan ketika user melakukan scroll terhadap table view tersebut.

5.4.3 Melakukan Pencarian Terhadap Suatu Kata

Untuk melakukan pencarian pada suatu kata maka user dapat menekan tombol search pada keyboard atau dengan memilih kata yang ditampilkan oleh browser. Kata yang dipilih tersebut akan diproses oleh class WordNet dan dikirimkan kepada halaman result. Pemrosesan data tersebut akan memanfaatkan class Word untuk menampung hasil ekstraksi. Proses ekstraksi dan pengiriman data pada halaman result ditunjukkan oleh segmen program 5.6.

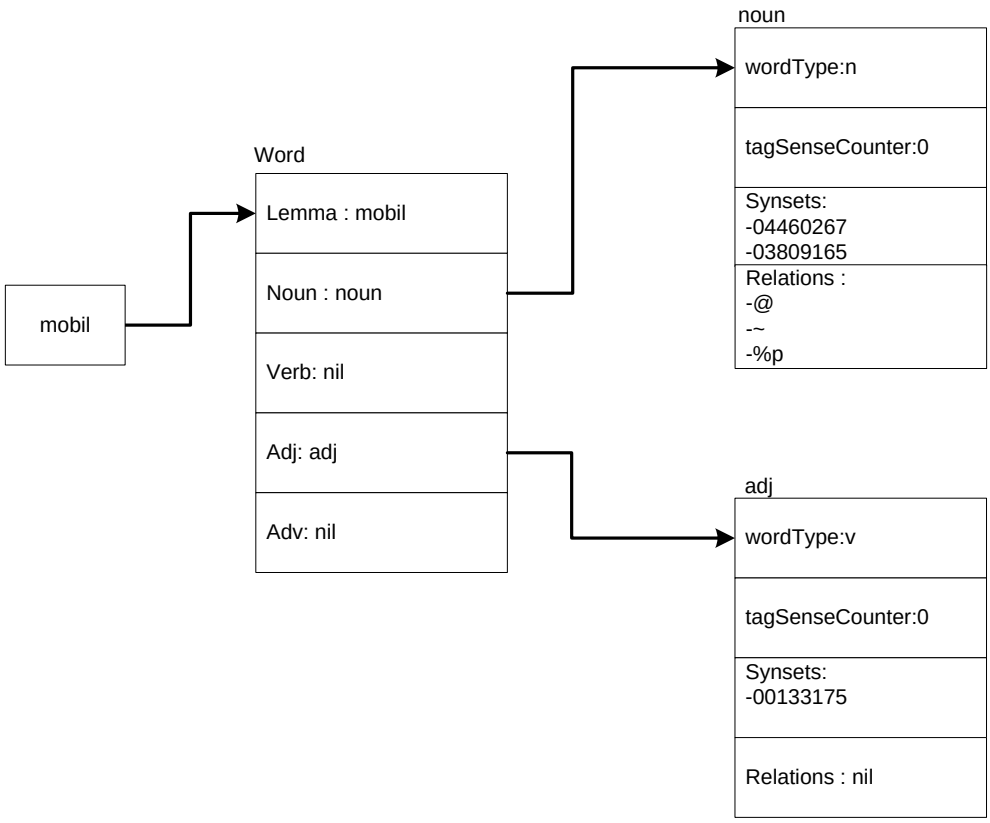
Segmen program 5.6 Pemrosesan Kata yang Diinputkan User

```
1:  self.word = [WN getWordData:[searchBar.text
    stringByReplacingOccurrencesOfString:@" " withString:@"_" ]];

2:  [[NSNotificationCenter defaultCenter]
    postNotificationName:@"openResult" object: self.word];
```

Baris 1 segmen program 5.6 menunjukkan pemanggilan prosedur WordData milik class WordNet. Parameter dari prosedur ini adalah text dari text box pada search bar. Text tersebut terlebih dahulu mengalami penggantian karakter, dimana karakter spasi akan berubah menjadi karakter garis bawah (_). Karena karakter spasi tidak terdapat pada Lexical Database, apa bila karakter spasi diperlukan maka digunakan karakter garis bawah. Hasil dari prosedur akan tersimpan pada variabel dengan class Word.

Pada tahap ini informasi yang didapatkan adalah lemma kata, serta synset offset yang dimiliki oleh kata tersebut pada setiap class kata, selain itu juga terdapat relasi yang dimiliki oleh kata tersebut. Synset offset yang didapatkan akan diproses pada halaman result bersama dengan ekstraksi relasi yang dimiliki. Hasil ekstraksi ini akan ditampung dengan memanfaatkan struktur class Word. Ilustrasi dari hasil ekstraksi ini ditunjukkan oleh gambar 5.6 dimana kata mobil akan dicari pada setiap class kata. Kata mobil tersebut akan dicari pada semua index dictionary, dan ternyata kata mobil terdapat pada dua index dictionary yaitu noun dan adj. Hal tersebut menandakan bahwa kata mobil terdapat pada dua buah class kata. Pada setiap class kata tersebut didapatkan synset yang dimiliki oleh

kata mobil, selain itu relasi yang dimiliki juga didapatkan walaupun belum diketahui synset pemilik relasi tersebut.

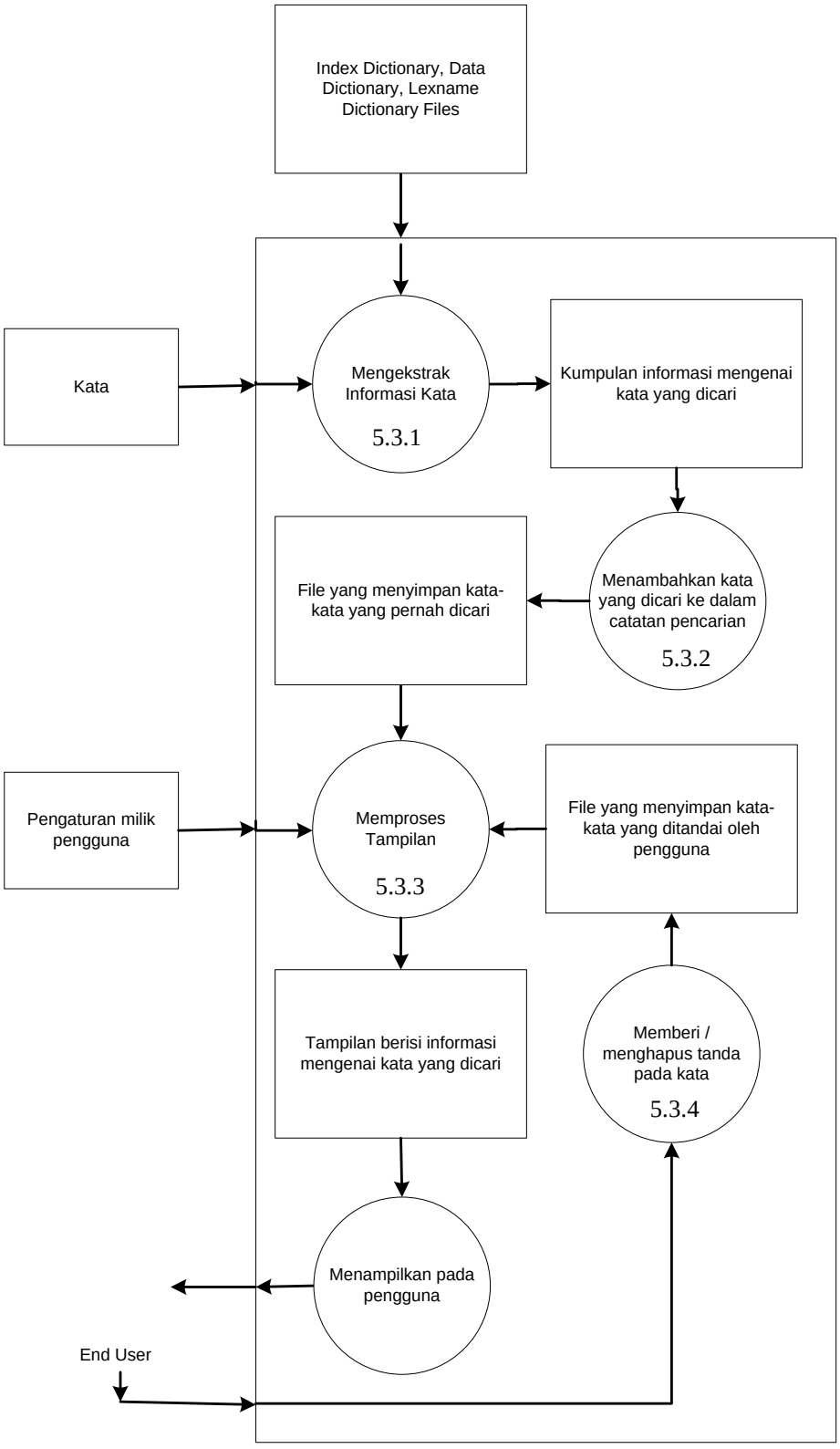


Gambar 5.6
Hasil Ekstraksi Kata Mobil

Baris 2 menunjukkan bahwa hasil ekstraksi dikirimkan ke halaman result melalui NSNotification openResult. NSNotification digunakan ketika suatu prosedur dari view yang lain ingin dipanggil.

5.3 Halaman Result

Halaman result merupakan halaman yang berperan untuk menampilkan informasi mengenai kata yang dicari oleh user. Pada halaman ini user juga bisa menambahkan / menghilangkan bookmark pada kata. Semua kata yang pernah dicari akan dicatat ke dalam history. Gambar 5.7 berikut menunjukkan arsitektur sistem dari halaman result.



Gambar 5.7
Arsitektur Sistem Halaman Result

Pada gambar 5.7 ditunjukkan bahwa proses pertama yang dilakukan oleh halaman result adalah mengekstrak informasi kata, ekstraksi ini dilakukan pada synset offset yang diterima. Setelah semua informasi tersebut telah terekstrak maka kata yang dicari akan dicatat ke dalam catatan pencarian browser. Langkah terakhir adalah memproses tampilan sebelum ditampilkan kepada user, apabila user menambahkan / menghilangkan bookmark pada kata tersebut maka proses akan diperbaharui dan ditampilkan ulang kepada user. Setiap proses yang terjadi akan dijelaskan pada sebuah subbab tersendiri.

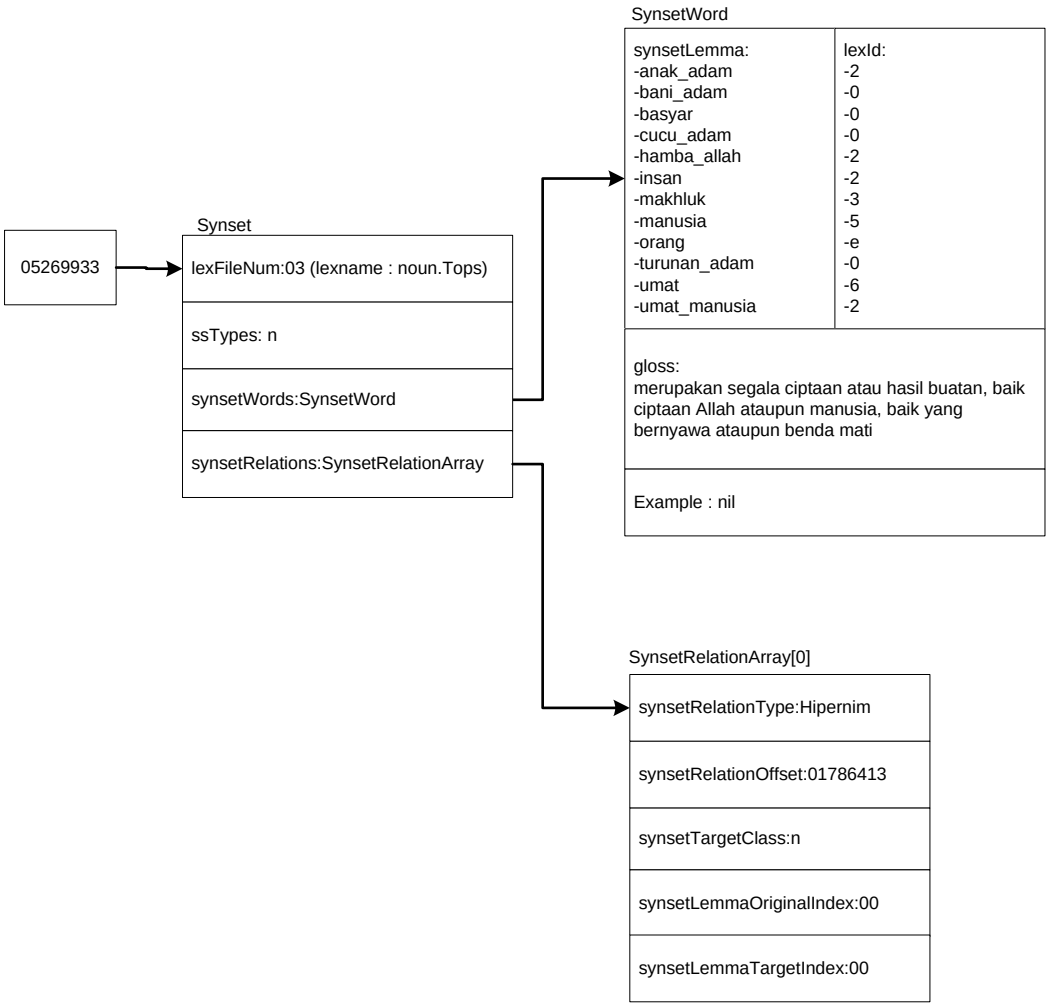
5.3.1 Mengekstrak Informasi Kata

Proses ekstraksi informasi pada halaman result melibatkan prosedur getSynsetData milik class WordNet. Halaman result akan menerima sebuah variabel dengan tipe class Word dari halaman search, pada class tersebut sudah terkandung synset offset yang dimiliki oleh kata pada masing-masing class kata. Yang harus dilakukan oleh halaman result adalah mengekstrak informasi pada setiap synset offset yang didapatkan. Segmen program 5.7 Menunjukkan proses ekstraksi informasi synset.

Segmen Program 5.7 Ekstraksi Informasi Synset

```
1:  NSMutableArray *listOfSynsets;
2:  Synsets *synset;

3:  if([wordClass isEqualToString:@"noun"])
4:     listOfSynsets = [[word noun]synsets];
5:  if([wordClass isEqualToString:@"verb"])
6:     listOfSynsets = [[word verb]synsets];
7:  if([wordClass isEqualToString:@"adj"])
8:     listOfSynsets = [[word adj]synsets];
9:  if([wordClass isEqualToString:@"adv"])
10:    listOfSynsets = [[word adv]synsets];
11:
12: for(int i =0 ; i < [listOfSynsets count];i++)
13: {
14:    synset = [WN getSynsetData:
                 [listOfSynsets objectAtIndex:i]forDict : wordClass];

15:    SynsetWords *wordsInSynset = [[SynsetWords alloc]init];
16:    wordsInSynset = [synset synsetWords];

17:    //Memproses Tampilan
18: }
```

Pada baris 3-10 segmen program 5.7 dilakukan pemeriksaan terhadap variabel wordClass. Variabel tersebut bertipe NSString dan mengandung tipe class kata yang akan diekstrak. Synset offset yang akan ditampung oleh variabel listOfSynsets bergantung pada class kata yang disimpan oleh variabel wordClass. Pada baris 12 dilakukan iterasi sejumlah synset offset yang dimiliki pada class kata yang sedang diproses. Pada baris 14 digunakan prosedur getSynsetData milik class WordNet, dimana parameternya adalah synset offset yang dimiliki dan class kata yang sedang diproses. Hasil ekstraksi ini akan digunakan untuk memproses tampilan pada browser. Gambar 5.8 Menunjukkan hasil ekstraksi synset offset.



Gambar 5.8
Hasil Ekstraksi Synset Offset

Gambar 5.8 Menunjukkan kondisi informasi yang didapatkan dari hasil ekstraksi. Ditunjukkan bahwa synset offset 05269933 memiliki informasi pada class kata noun dan verb. Informasi hasil ekstraksi ini telah siap diproses untuk ditampilkan oleh browser.

Tetapi apakah yang terjadi apabila sebuah kata tidak memiliki synset offset? Misalnya kondisi dimana sebuah kata tidak terdapat pada Lexical Database, maka halaman search tidak akan memberikan synset offset kepada halaman result. Ketika kondisi ini terjadi maka browser akan menggunakan algoritma Levenshtein Distance untuk mencari kata yang memiliki kemiripan penulisan.

5.3.1.1 Levenshtein Distance

Levenshtein Distance merupakan sebuah algoritma yang digunakan untuk mencari jarak antara dua buah kata. Jarak yang dimaksud adalah biaya yang diperlukan untuk merubah sebuah kata menjadi kata yang lain, perubahan tersebut dapat dilakukan dengan melakukan penambahan, pengurangan dan pertukaran posisi huruf. Segmen program 5.8 menunjukkan prosedur levenshtein program yang digunakan pada Tugas Akhir ini.

Segmen Program 5.8 Levenshtein Distance

```
1:  int levenshtein_distance(const char *s,const char *t)
2:  {
3:    int k,i,j,n,m,cost,*d,distance;
4:    n=strlen(s);
5:    m=strlen(t);
6:    if(n!=0&&m!=0)
7:    {
8:      d=malloc((sizeof(int))*(m+1)*(n+1));
9:      m++;
10:     n++;

11:     for(k=0;k<n;k++)
12:       d[k]=k;
13:     for(k=0;k<m;k++)
14:       d[k*n]=k;

15:     for(i=1;i<n;i++)
16:        for(j=1;j<m;j++)
17:        {
18:          if(s[i-1]==t[j-1])
```

Segmen Program 5.8 (Lanjutan)

```
19:             cost=0;
20:           else
21:             cost=1;
22:
23:             d[j*n+i]=minimum(d[(j-1)*n+i]+1,d[j*n+i-1]+1,d[(j-
                1)*n+i-1]+cost);
24:         }
25:     distance=d[n*m-1];
26:     free(d);
27:     return distance;
28:   }
29:   else
30:     return -1;
31: }
```

Segmen program 5.8 menunjukkan sebuah prosedur untuk menghitung Levenshtein Distance yang dibuat dengan menggunakan C++. Hal ini dilakukan karena penghitungan akan melibatkan keseluruhan kata yang berada pada Lexical Database, agar proses perhitungan tidak memakan waktu lama maka digunakan bahasa pemograman C++. Pertama-tama akan dilakukan pemeriksaan pada baris 6 apakah kedua kata yang diberikan sebagai parameter bukanlah kata kosong atau spasi, jika ternyata salah satu kata adalah kosong maka prosedur akan menjalankan baris 30 dimana dikembalikan nilai -1. Tetapi jika tidak maka pada baris 8 disiapkan sebuah array dua dimensi dengan tipe int, array ini akan digunakan untuk menampung biaya yang dihasilkan. Pada baris 11-14 dilakukan insialisasi terhadap array. Pada baris 18 dilakukan pemeriksaan apakah kedua huruf yang sedang diperiksa sama, jika tidak maka akan dihitung biaya yang diperlukan untuk menyamakan kedua huruf tersebut. Proses perhitungan tersebut dilakukan pada baris 23, dimana terdapat dicari nilai minimum dari penambahan huruf, pengurangan huruf dan pertukaran posisi huruf. Pada baris ke 27 biaya yang didapatkan akan dikembalikan.

Tabel 5.3 menunjukkan contoh perhitungan Levenshtein Distance antara kata “mania” dengan kata “manusia”. Perhitungan akan menghasilkan nilai 2, dimana terjadi proses penghapusan terhadap dua buah huruf, yaitu “u” dan “s”. Ketiga kata pertama yaitu “man” terdapat pada kedua kata sehingga biaya yang dibutuhkan masih bernilai 0, tetapi ketika huruf “i” dan “u” dibandingkan maka terjadi proses penghapusan. Begitu juga dengan huruf “i” dan “s”, huruf “s” akan

mengalami penghapusan. Sisa huruf terakhir “ia” terdapat pada kedua kata sehingga tidak memakan biaya apapun. Alur pemeriksaan ditandai dengan tanda garis bawah.

Tabel 5.3
Perhitungan Levenshtein Distance

		M	A	N	I	A
	0	1	2	3	4	5
M	1	0	1	2	3	4
A	2	1	0	1	2	3
N	3	2	1	0	1	2
U	4	3	2	1	1	2
S	5	4	3	2	2	2
I	6	5	4	3	2	3
A	7	6	5	4	3	2

Karena Levenshtein Distance akan melibatkan seluruh kata pada Lexical Database maka waktu yang diperlukan akan cukup lama. Agar user tidak mengira bahwa browser berhenti bekerja maka Levenshtein Distance ini akan dijalankan pada background process. Selain itu threshold yang digunakan pada Tugas Akhir ini adalah dua. Apabila biaya yang diperlukan untuk mengubah menjadi sebuah kata lain lebih dari dua maka kata lain tersebut tidak akan ditampikan. Segmen program 5.9 menunjukkan aplikasi dari penggunaan prosedur Levenshtein Distance.

Segmen Program 5.9 Pemanfaatan Levenshtein Distance

```
1:  int distance = 2;
2:  NSMutableArray *nearest =[[ NSMutableArray alloc]init];
3:  for (NSString *str in [WN getListOfAvailableWords]) {
4:      int temp = levenshtein_distance([str UTF8String], [[word
        lemma]UTF8String]);
5:      if (distance >= temp)
6:      {
7:          distance  = temp;
```

Segmen Program 5.9 (Lanjutan)

```
8:          NSMutableDictionary *near = [NSDictionary
            dictionaryWithObject:[NSString
            stringWithFormat:@"%i",distance] forKey:str];
9:          [nearest addObject:near];
10:     }
11: }
```

Baris 1 segmen program 5.9 menunjukkan nilai threshold yang akan digunakan. Pada baris 3 dilakukan iterasi terhadap semua kata pada Lexical Database, semua kata tersebut akan dihitung terhadap kata yang dicari oleh user. Apabila jarak yang didapatkan lebih kecil dari pada threshold maka kata tersebut beserta dengan jaraknya akan disimpan, selain itu jarak tersebut akan menjadi nilai threshold yang baru. Proses ini ditunjukkan oleh baris 4-10. Tabel 5.4 menunjukkan contoh penerapan Levenshtein Distance.

Tabel 5.4
Penerapan Levenshtein Distance

	Maunsia
Malasia	2
Mania	2
Manusia	2

Pada tabel 5.4 ditunjukkan perhitungan Leveshtein Distance kata maunsia terhadap kata malasia, mania dan manusia. Karena ketiga kata tersebut memenuhi threshold yang telah ditentukan maka ketiga kata tersebut akan ditampilkan sebagai kata yang memiliki kemiripan dengan kata yang diinputkan oleh user.

5.3.2 Menambahkan Kata yang Dicari Ke dalam Catatan Pencarian.

Ketika user mencari sebuah kata, maka kata tersebut akan dicatat ke dalam sebuah file plist bernama history. Apabila ternyata kata tersebut sudah pernah tercatat maka posisi kata tersebut akan berubah menjadi posisi pertama y pada halaman history. Pencatatan ini juga dilakukan apabila user melakukan pencarian

dengan memilih kata yang ditampilkan oleh browser pada halaman result. Misalnya kata yang merupakan hiponim dari kata manusia atau kata yang merupakan sinonim dari kata manusia. Proses pencatatan tersebut dilakukan oleh sebuah prosedur bernama prosedur addToHistory.

Segmen Program 5.10 Prosedur addToHistory

```
1:  -(void) addToHistory : (id) words
2:  {
3:    BOOL check = NO;
4:    int temp=0,tcounter=0;

5:    for(NSString *w in historyList)
6:    {
7:      if ([[w lowercaseString] isEqualToString:[words
        lowercaseString] ])
8:      {
9:        check = YES;
10:       temp = tcounter;
11:       break;
12:     }
13:     tcounter++;
14:   }
15:   if (check == NO && [setting recordHistory])
16:   {
17:     if (jump)
18:     {
19:         if(![lastVisited isEqualToString:@"bookmark"])
20:         {
21:           int prevIndex = indexHistory;
22:           NSString *prevWord = [[historyList
                objectAtIndex:prevIndex]retain];
23:           [historyList removeObjectAtIndex:prevIndex];
24:           [historyList addObject:prevWord];
25:           [prevWord release];
26:         }
27:     }
28:     if ([historyList count]==maxHistory && [historyList
        count] >= [[setting size]intValue])
29:     {
30:         [historyList removeObjectAtIndex:0];
31:     }

32:     [historyList addObject:words];
33:     if(![lastVisited isEqualToString:@"bookmark"])
34:         indexHistory = [historyList count]-1;
35:   }
36:   else {
37:     if(jump)
38:     {
39:         if(![lastVisited isEqualToString:@"bookmark"])
40:         {
41:           int prevIndex = indexHistory;
```

Segmen Program 5.10 (Lanjutan)

```
42:            NSString *prevWord = [[historyList
               objectAtIndex:prevIndex]retain];
43:            [historyList removeObjectAtIndex:prevIndex];
44:            [historyList addObject:prevWord];

45:            [historyList removeObjectAtIndex:temp];
46:            [historyList addObject:words];
47:            indexHistory = [historyList count]-1;

48:            [prevWord release];
49:         }
50:      }
51:      else {
52:         if(![lastVisited isEqualToString:@"bookmark"])
53:         {
54:            if (![lastVisited isEqualToString:@"history"])
55:            {
56:                if (!back && !next)
57:                {
58:                    if (check == YES)
59:                    {
60:                       [historyList removeObjectAtIndex:temp];
61:                       [historyList addObject:words];
62:                       indexHistory = [historyList count]-1;
63:                    }
64:                }
65:                else {
66:                    indexHistory = temp;
67:                }
68:            }
69:            else {
70:                indexHistory = temp;
71:            }
72:         }
73:      }
74:   }
75:   jump= NO;
76:   back= NO;
77:   next= NO;
78:   [historyList writeToFile:[[self documentPath]
      stringByAppendingPathComponent:@"/history.plist"]
      atomically:YES];
79: }
```

Sebelum sebuah kata disimpan ke dalam history, terlebih dahulu dilakukan pemeriksaan apakah kata tersebut sudah berada pada history. Pada baris 5 segmen program 5.10 dilakukan iterasi terhadap variabel historyList, variabel tersebut menampung semua kata pada history.plist. Setiap kata tersebut akan diperiksa dengan kata yang menjadi parameter prosedur, apabila kedua kata tersebut sama maka variabel check akan bernilai YES dan index kata pada historyList akan

dicatat. Setelah itu iterasi akan dihentikan dengan menggunakan perintah break, proses ini ditunjukkan oleh baris 7-12.

Apabila sebuah kata tidak pernah dicari sebelumnya, dengan kata lain tidak berada pada array historyList maka tersebut akan dimasukkan pada index terakhir history. Tetapi perlu dilakukan pemeriksaan apakah pencarian tersebut berasal dari pemilihan kata yang merupakan bagian dari relasi atau sinonim, jika ya maka kata pemilik relasi ataupun sinonim harus mengalami perubahan index. Pada baris 17 dilakukan pemeriksaan terhadap variabel jump, variabel ini akan bernilai YES jika pencarian berasal dari relasi atau sinonim sebuah kata. Pada baris 19 jika dilakukan pemeriksaan apakah kata pemilik relasi atau sinonim berada pada halaman bookmark, jika tidak maka perubahan index kata tersebut akan dilakukan, proses perpindahan ini ditunjukkan pada baris 21-25.

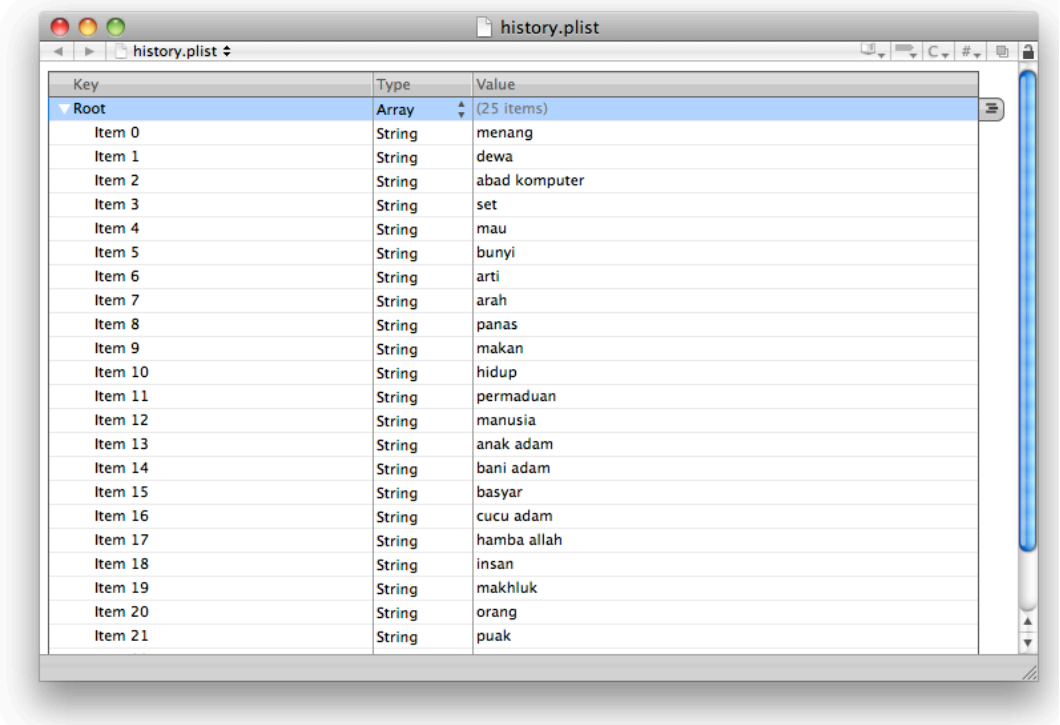
Setelah perpindahan index dilakukan maka langkah selanjutnya adalah memeriksa batasan penyimpanan kata, apabila ternyata kata yang disimpan sudah melebihi batas maka kata yang berada pada index pertama akan dihapus. Pemeriksaan dan penghapusan tersebut ditunjukkan oleh baris 28-31. Pada baris 32 kata yang dicari akan dimasukkan ke dalam array historyList. Setelah kata disimpan maka diperiksa kembali apakah kata yang dicari berada pada halaman bookmark, jika tidak maka index kata yang aktif pada halaman history akan diset pada kata yang terakhir kali dicari. Tetapi apabila kata berada pada halaman history maka index kata pada halaman history tidak akan berubah, hal ini ditujukan agar user dapat mengetahui kata yang terakhir kali sedang dilihat oleh user pada halaman history. Proses pemindahan index tersebut ditunjukkan pada baris 33 dan 34 dimana index history akan diisi dengan ukuran dari array historyList dan dikurangi dengan 1 karena index array dimulai dari angka 0.

Apabila sebuah kata sudah pernah dicari oleh user maka kata tersebut tidak perlu dicatat kembali melainkan hanya mengalami perpindahan index saja. Pada baris 38 dan 40 dilakukan pemeriksaan yang sama dengan kondisi dimana kata belum pernah dicari oleh user, tetapi pada tahap ini selain merubah index kata pemilik relasi atau sinonim, kata yang mengalami pencarian juga akan mengalami perubahan index. Kata yang dicari akan berada pada index terakhir sedangkan

kata pemilik relasi atau sinonim akan berada pada satu index sebelum terakhir. Proses perpindahan index kata pemilik relasi atau sinonim ditunjukkan oleh baris 42-45, sedangkan proses perpindahan kata yang dicari ditunjukkan oleh baris 46-48.Tetapi apabila kata yang dicari bukan berasal dari sinonim atau relasi kata lain maka index yang berubah hanyalah index kata tersebut. Hal ini akan menjadi masalah ketika fitur Word Navigation dilakukan. Karena index dari kata yang dilihat dari fitur navigation tidak boleh berubah, jika index nya berubah maka word navigation akan selalu menampilkan kata yang sama. Untuk mengatasi hal ini maka pada baris 53 dan 55 dilakukan pemeriksaan apakah kata tersebut berasal dari halaman bookmark atau history.

Jika kata tersebut berasal dari halaman history maka index halaman history akan berubah sesuai dengan index kata pada hapa array historyList. Hal ini dilakukan untuk mengatasi kondisi ketika user ingin melihat informasi sebuah kata pada halaman history, index dari kata yang dilihat tidak akan berubah, proses pemindahan index ditunjukkan oleh baris 64. Tetapi jika kata dicari user melalui halaman seach dan user sedang tidak menggunakan fitur word navigarion maka index dari kata akan berubah, proses pemeriksaan dan perpindahan ini ditunjukkan oleh baris 57-68. Tetapi jika user menggunakan fitur word navigation maka index halaman history akan berubah seperti yang ditunjukkan pada baris 71. Nilai dari variabel back, jump, dan next aka direset agar variabel tersebut tidak mempengaruhi pemanggilan prosedur addToHistory yang berikutnya. Proses tersebut ditunjukkan oleh baris 76-78. Langkah yang terakhir adalah menuliskan array historyList ke dalam sebuah file plist bernama history.plist. File tersebut akan disimpan pada folder document milik device, untuk melakukan pembuatan file dilakukan proses seperti pada baris 79.

File history.plist yang dihasilkan akan menjadi sumber pembacaan untuk halaman history. Kata-kata yang tersimpan dalam plist tersebut akan mengalami penggantian karakter garis bawah (_) menjadi karakter spasi. Kemudian urutan kata tersebut akan ditukar, kata pada index pertama plist akan berada pada posisi paling bawah table view dan sebaliknya.



Gambar 5.9
File History.plist

Gambar 5.9 menunjukkan file history.plist yang menyimpan empat buah kata yang pernah dicari oleh user, yaitu kata manusia, aba-aba, set dan tata. Kata tersebut tersimpan sesuai dengan urutan pencarian, kata pertama yang dicari oleh user adalah kata manusia dan kata yang terakhir kali dicari oleh user adalah kata tata.

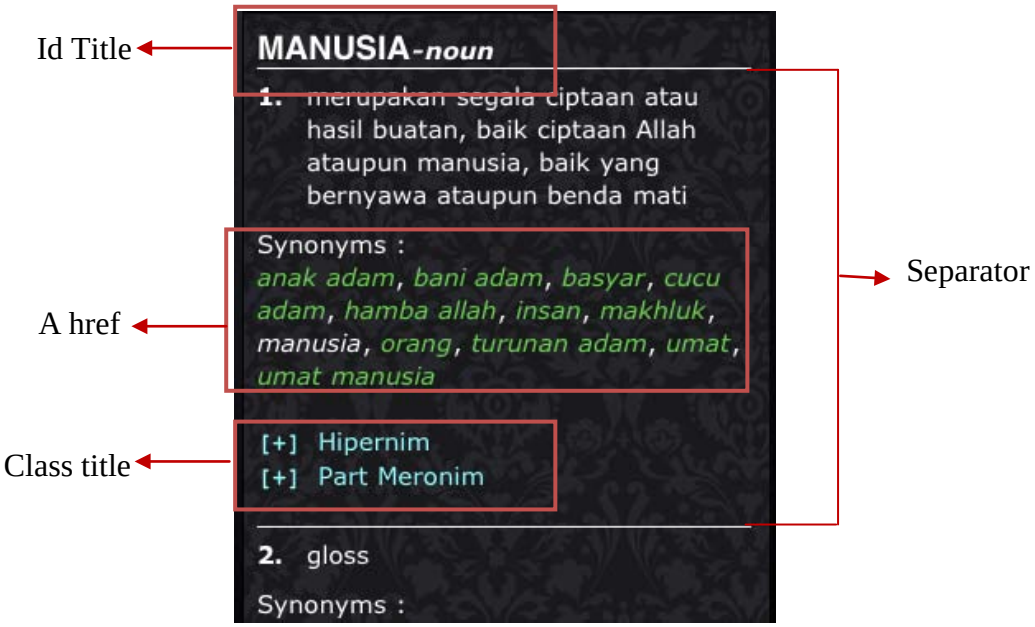
5.3.3 Memproses Tampilan

Setelah informasi kata didapatkan dan kata dicatat ke dalam catatan pencarian kini browser akan mempersiapkan tampilan untuk ditampilkan kepada user. Browser akan memanfaatkan UIWebView untuk menampilkan informasi kepada user. UIWebView digunakan karena sebuah HTML memiliki kebebasan untuk membuat sebuah interface yang menarik dengan memanfaatkan CSS dan javascript.Terdapat beberapa tahap pemrosesan tampilan yang dilakukan oleh browser. Tahap pertama adalah mempersiapkan tag-tag CSS dan java script yang akan digunakan.

Segmen Program 5.11 CSS dan Javascript

```
1:  <style type='text/css'>
2:    a{color:#6ad456;text-decoration:none;}
3:    #data{width:300px;0}
4:    #Title{font-size:20;font-family:helvetica;}
5:    .title{
6:       font-weight:bold;
7:       float:left;width:30px;margin-left:0px;
8:    }
9:    .separator{
10:      width:100%;height:6px;
11:      border-top:1px solid #FFF;
12:   }
13:   .dataCentral{
14:      float:left;width:270px;
15:      text-align:left;margin-bottom:10px;
16:   }
17:   .lexType{font-size:5;}
18:   .relation{color:#99FFFF;}
19:   body{
20:      background-image:url('bg.png');
21:      background-color:black;color:white;
22:      font-family:'verdana';
23:      font-size:15px;
24:      -webkit-touch-callout: none;
25:   }
26: </style>
27: <script language='javascript'>
28:   function xRelation(id){
29:     if (document.getElementById('RelationData'+id)
           .style.display=='block')
30:     {
31:      document.getElementById('RelationData'+id).style.display
        ='none';
32:      document.getElementById('xsign'+id).innerHTML='[+]  ';
33:     }
34:     else{
35:      document.getElementById('RelationData'+id).style.display
        ='block';
36:      document.getElementById('xsign'+id).innerHTML='[-]  ';
37:     }
38:   }
39:   function xSynset(id){
40:     if (document.getElementById('SynsetData'+id)
           .style.display=='none')
41:     {
42:      document.getElementById('SynsetData'+id).style.display='
        block';
43:      document.getElementById('xsign'+id).innerHTML='';
44:     }
45:   }
46: </script>
```

Segmen program 5.11 menunjukkan CSS dan javascript yang digunakan oleh browser. Pada baris 2 ditunjukkan pengaturan warnat untuk ahref yang ada pada browser, sinonim dan relasi-relasi milik kata akan ditampilkan dalam bentuk a href. Pada gambar 5.10 ditunjukkan kata-kata yang merupakan sinonim dari manusia merupakan ahref dengan warna *#6ad456*. Pada baris 4 dideklarasikan id Title yang akan digunakan pada kata manusia pada gambar 5.10. Sedangkan class title pada baris 5 akan digunakan untuk relasi-relasi yang dimiliki seperti Hipernim dan hipernim pada gambar 5.10. Sedangkan separator pada baris 9 berfungsi sebagai pemabatas antar sense kata. Class data, dataCentral dan relation pada baris 3, 13, 18 akan digunakan dalam penyajian gloss, sinonim, dan relasi.



Gambar 5.10
Penerapan CSS dan Javascript

Javascript akan digunakan untuk menampilkan dan menghilangkan relasi kata dan sinonim yang dimiliki. Fungsi xRelation berfungsi untuk menghilangkan dan menampilkan kata-kata pada suatu relasi, user dapat melakukannya dengan melakukan tap pada relasi yang diinginkan. Pada baris 29 dilakukan pemeriksaan apakah kata-kata sedang ditampilkan atau tidak jika tidak maka kata-kata yang ada akan ditampilkan begitu juga sebaliknya. Sedangkan fungsi xSynset berfungsi

untuk menampilkan sinonim yang dimiliki, apabila jumlah sinonim sebuah kata melebihi batas yang ditentukan oleh browser maka sisa sinonim akan disembunyikan. Jika user ingin melihat sinonim yang disembunyikan maka user dapat melakukan tap pada simbol "[…]". Pada baris 40 dilakukan pemeriksaan apakah kata-kata yang disembunyikan telah muncul, jika belum maka kata-kata tersebut akan ditampilkan tetapi jika sudah tampil maka browser tidak akan melakukan apapun.

Setelah CSS dan javascript telah dideklarasikan maka langkah selanjutnya adalah meletakkan informasi-informasi yang diperlukan ke dalam tag HTML dan memberikan class dan id yang sesuai. Segmen program 5.12 menunjukkan proses peletakkan informasi serta pemberian class dan id.

Segmen Program 5.12 Mempersiapkan CSS dan Tampilan Gloss

```
1: for(int i =0 ; i < [listOfSynsets count];i++)
2: {

3:  synset = [WN getSynsetData:[listOfSynsets
    objectAtIndex:i]forDict : wordClass];
4:  SynsetWords *wordsInSynset = [[SynsetWords alloc]init];
5:  wordsInSynset = [synset synsetWords];

6:  if(i==0)
7:  {
8:    //memproses css dan javascript
9:    message = [message stringByAppendingString:@"
       <body >"];

10:   message = [message stringByAppendingFormat:@"
      <span id='Title'><b>%@</b></span>",[[word
      lemma]uppercaseString]];

11:   message = [message stringByAppendingFormat:@"
      <span clas='lexType'><i><b>-%@</b></i></span>",wordClass];

12:   message = [message stringByAppendingString:@"
      <div id='data'>"];
13: }

14: message = [message stringByAppendingString:@"
    <div class='separator'></div>"];
15: message = [message stringByAppendingFormat:@"
      <div class='title'>%i. </div>",i+1];
16: message = [message stringByAppendingString:@"
            <div class = 'dataCentral'>"];
17: if(![setting pwnLayout])
18:   {
```

Segmen Program 5.12 (Lanjutan)

```
19:     if ([[wordsInSynset gloss] length] > 0 &&
            [setting showGloss])
20:         {
21:          message = [message
             stringByAppendingFormat:@"%@",[wordsInSynset gloss]];
22:         }

23:     message = [message stringByAppendingString:@"<i>"];
24:
25:     if ([[wordsInSynset example] length] > 0)
26:         {
27:           message = [message
              stringByAppendingFormat:@": %@ <br>",[wordsInSynset
              example]];
28:         }
29:      message = [message stringByAppendingString:@"</i>"];
30:      message = [message stringByAppendingString:@"</div>"];
31:   }
```

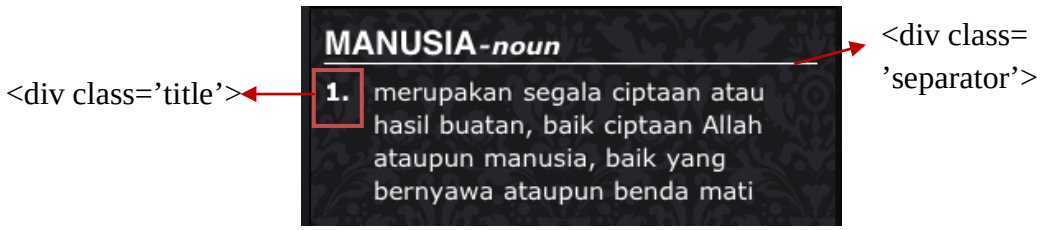
Segmen program 5.12 menunjukkan bahwa prosesi tampilan dilakukan setelah ekstraksi informasi selesai dilakukan. Pada baris 6 dilakukan pemeriksaan apakah synset yang sedang diproses adalah synset pertama, jika ya maka CSS dan java script akan dipersiapkan. Setiap baris CSS dan javascript pada segmen program 5.9 akan ditampung oleh perintah message, seperti pada contoh berikut.

```
message = [message stringByAppendingString:@"a{color:#6ad456;text-
decoration:none;}"];
```

Karena setiap baris CSS memiliki pola yang sama maka pada segmen program 5.9 ditunjukkan hanya CSS dan javascript yang digunakan. Setelah CSS dan javascript telah siap maka langkah berikutnya adalah mempersiapkan tag body dan lemma kata yang sedang dicari, pada baris 9 dipersiapkan tag body yang diperlukan dan pada baris 10 dan 11 dipersiapkan span untuk menampung lemma kata dan tipe class kata. Gambar 5.11 menunjukkan proses yang dilakukan.

Gambar 5.11
Tahap Pertama Pemrosesan Tampilan

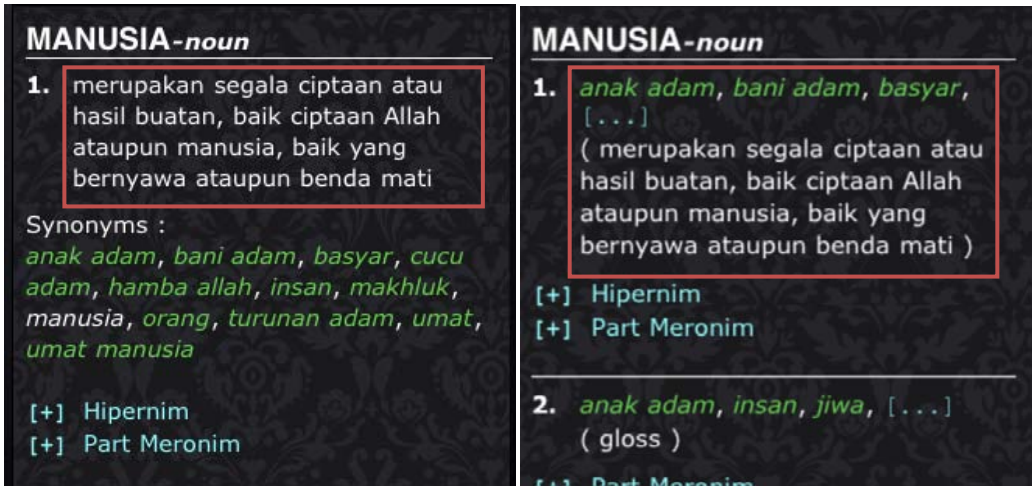
Setelah lemma kata siap maka div penampung semua synset akan disiapkan, pada baris 12 ditunjukkan sebuah div dengan id data akan digunakan untuk menampung semua informasi kata. Setiap sense atau setiap synset akan dipisahkan oleh sebuah garis, garis tersebut dibuat dengan menggunakan div dengan class separator pada baris 14. Separator tersebut akan diikuti dengan no sense yang ditunjukkan pada baris 15. Kemudian dipersiapkan sebuah div yang ditujukkan untuk menampung gloss kata, div ini ditunjukkan oleh baris 16.



Gambar 5.12
Tahap Kedua Pemrosesan Tampilan

Gambar 5.12 menunjukkan aplikasi dari div dengan class title dan div dengan class separator. Pada baris 17 dilakukan pemeriksaan apakah user sedang mengaktifkan fitur pwnLayout atau tidak. Fitur pwnLayout akan mempengaruhi informasi yang tersimpan pada div dataCentral. Pada layout normal, informasi pada dataCentral hanyalah berupa gloss dari synset, sedangkan pada pwnLayout div dataCentral akan mengandung sinonim milik synset beserta gloss yang dimiliki. Perbedaan yang lain adalah pada jumlah sinonim yang ditampilkan. Pada pwnLayout jumlah sinonim yang ditampilkan hanyalah tiga buah, hal ini dilakukan karena jumlah dari sinonim akan mempengaruhi peletakkan gloss. Semakin banyak jumlah sinonim maka gloss akan terletak semakin dibawah. Berbeda dengan layout normal dimana sinonim terletak dibawah gloss. Pada layout normal jumlah sinonim yang ditampilkan mencapai lima belas buah, jumlah tersebut tidak akan mempengaruhi peletakkan gloss.

Pada gambar 5.13 ditunjukkan perbedaan informasi yang disimpan oleh div dataCentral. Gambar 5.13(a) menunjukkan layout normal dan gambar 5.13(b) menunjukkan pwnLayout. Kotak berwarna merah menunjukkan div dataCentral.



(a) **(b)**

Gambar 5.13
Layout Tampilan
(a). Layout Normal
(b). PWN Layout

Pada baris 17-31 dilakukan penambahan gloss dan example pada div dataCentral. Sebelum gloss ditambahkan maka dilakukan pemeriksaan terlebih dahulu apakah user menginginkan gloss ditampilkan, jika ya maka barulah gloss ditambahkan, proses pemeriksaan ini ditunjukkan oleh baris 17.

Segmen Program 5.13 Memproses Tampilan Sinonim

```
1:  NSArray *sorted = [[wordsInSynset synsetLemma]
    sortedArrayUsingSelector:@selector(compare:)];
2:  if (![setting pwnLayout])
3:  {
4:     message = [message stringByAppendingString:@"Synonyms :
       <br>"];
5:  }
6:  for(int z=0;z<[[wordsInSynset synsetLemma]count];z++)
7:  {
8:     if (z<maxSynonym)
9:     {
10:     if (![[sorted objectAtIndex:z] isEqualToString:[[word
           lemma] stringByReplacingOccurrencesOfString:@" "
           withString:@"_"]])
11:     {
12:        message = [message stringByAppendingFormat:@"<i><a
           href=\"tacb://%@\">%@</a></i>",[sorted
           objectAtIndex:z],[sorted objectAtIndex:z]];
13:     }
14:     else {
```

Segmen Program 5.13 (Lanjutan)

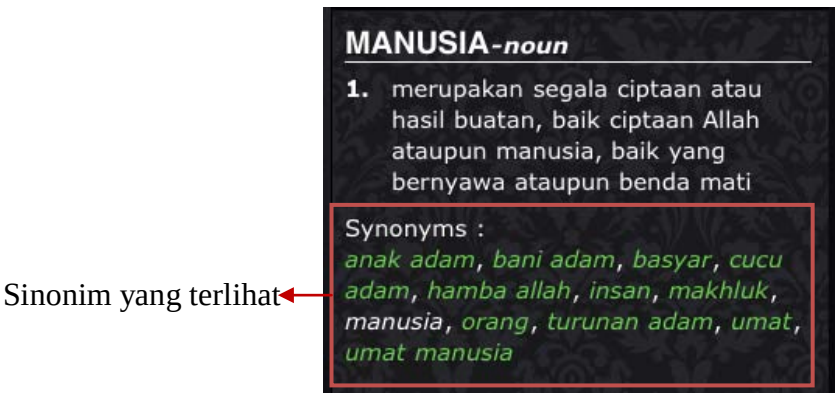
```
15:       message = [message
          stringByAppendingFormat:@"<i>%@</i>",[sorted
          objectAtIndex:z]];
16:     }
17:     if (z!=[[wordsInSynset synsetLemma]count]-1)
18:     {
19:       message = [message stringByAppendingString:@", "];
20:     }
21:   }
22:   else {
23:     if (z==maxSynonym)
24:     {
25:       counter+=1;

26:       message = [message stringByAppendingFormat:@"
          <span id='%i'class='Relation'onClick='xSynset(this.id)'>
          <span id='xsign%i' style='font-family:courier new;'>
          [...]
          </span>
          </span>",counter,counter];

27:       message = [message stringByAppendingFormat:@"
          <span id='SynsetData%i'style='display:none;'>",counter];
28:     }
29:     if (![[sorted objectAtIndex:z] isEqualToString:[[word
        lemma] stringByReplacingOccurrencesOfString:@" "
        withString:@"_"]])
30:     {
31:       message = [message stringByAppendingFormat:@"
          <a href=\"tacb://%@\">%@</a>",[sorted
          objectAtIndex:z],[sorted objectAtIndex:z]];
32:     }
33:     else {
34:       message = [message
          stringByAppendingFormat:@"<i>%@</i>",[sorted
          objectAtIndex:z]];
35:     }
36:     if (z!=[[wordsInSynset synsetLemma]count]-1)
37:       message = [message stringByAppendingString:@", "];
38:     if (z==[[wordsInSynset synsetLemma]count]-1)
39:       message = [message stringByAppendingString:@"</span>"];
40:   }
41: }
42: message = [message stringByAppendingString:@"<br>"];
43: if (![setting pwnLayout])
44:   message = [message stringByAppendingString:@"<br>"];
```

Setelah informasi gloss disimpan maka langkah selanjutnya adalah memproses informasi sinonim. Apabila fitur pwnLayout diaktifkan berarti pada tahap ini informasi gloss belum diproses dan informasi sinonim merupakan informasi yang pertama kali diproses. Baris 1 segmen program 5.13 menunjukan

proses pengurutan sinonim, sinonim-sinonim yang ada akan diurutkan sesuai dengan abjad. Pada baris 2 dilakukan pemeriksaan apakah fitur pwnLayout diaktifkan, jika ternyata tidak aktif berarti sinonim akan dipisahkan dengan gloss karena itu pada 4 ditambahkan string “Synonym : “. Baris 6-41 merupakan proses memasukkan sinonim ke dalam div dataCentral. Pada baris 6 dilakukan iterasi sejumlah sinonim yang dimiliki oleh synset. Pada setiap iterasi diperiksa apakah jumlah sinonim yang ditampilkan telah melebihi batas, apa bila belum maka sinonim tersebut akan langsung ditampilkan. Pemeriksaan jumlah sinonim dilakukan oleh baris 8. Pada baris 10 sinonim tersebut akan mengalami pemeriksaan kembali apakah sinonim tersebut sama dengan kata yang sedang ditampilkan. Jika tidak maka pada baris 12 browser akan membuat sebuah ahref untuk link tersebut. Pada baris 17 dilakukan pemeriksaan apakah masih terdapat sinonim untuk ditampilkan, jika ya maka pada baris 19 ditambahkan karakter koma (,). Berikut ini adalah contoh dari ahref yang dihasilkan.

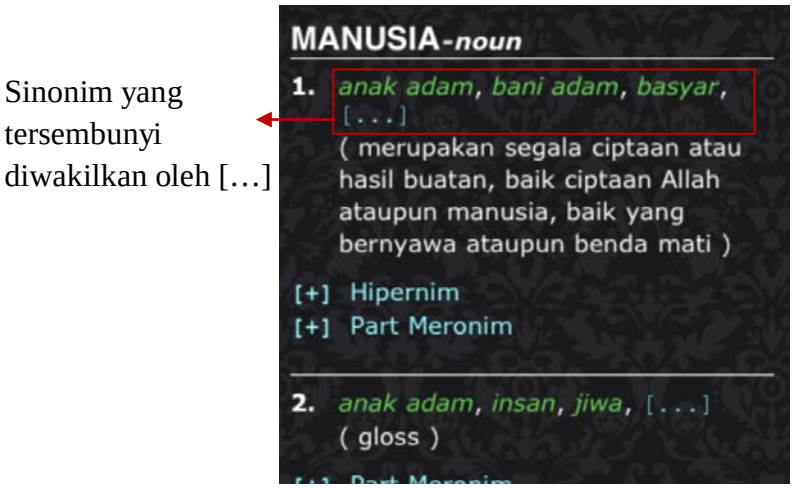
```
<a href=\"tacb://anak_adam\">anak_adam</a>
```

Link yang dihasilkan pada tahap ini masih belum mengganti karakter garis bawah menjadi karakter spasi. Perubahan tersebut akan dilakukan ketika tag HTML sudah tersusun secara lengkap. Gambar 5.14 menunjukkan link-link yang terbentuk untuk setiap sinonim pada layout normal.



Gambar 5.14
Memproses Sinonim

Ketika batas sinonim yang sudah ditampilkan sudah tercapai maka program akan menjalankan baris 23. Pada baris tersebut dilakukan pemeriksaan apakah sinonim tersebut adalah sinonim pertama setelah batas sinonim yang ditampilkan tercapat. Jika ya maka pada baris 26 dan 27 dibuatlah sebuah span untuk menampung sinonim yang tidak ditampilkan. Baris 29-37 memiliki fungsi yang sama dengan kondisi dimana sinonim masih belum melebihi batas sinonim yang ditampilkan. Gambar 5.15 menunjukkan hasil dari sinonim yang disembunyikan pada pwnLayout.



Gambar 5.15
Sinonim yang disembunyikan

Setelah sinonim diproses maka perlu dilakukan pemeriksaan apakah pwnLayout diaktifkan, jika ya maka informasi tentang gloss perlu diproses sebelum relasi synset diproses.

Segmen Program 5.14 Memproses Tampilan Gloss PWN

```
1:  if(![setting pwnLayout])
2:     {
3:       if ([[wordsInSynset gloss] length] > 0 &&
             [setting showGloss])
4:           {
5:            message = [message
              stringByAppendingFormat:@"%@",[wordsInSynset gloss]];
6:           }
7:
```

Segmen Program 5.14 (Lanjutan)

```
8:       message = [message stringByAppendingString:@"<i>"];
9:
10:     if ([[wordsInSynset example] length] > 0)
11:         {
12:           message = [message
              stringByAppendingFormat:@": %@ <br>",[wordsInSynset
              example]];
13:         }
14:      message = [message stringByAppendingString:@"</i>"];
15:      message = [message stringByAppendingString:@"</div>"];
16:     if ([[synset synsetRelations]count]==0)
17:       message = [message stringByAppendingString:@"
          No Relationship Found <br>"];
18:   }
```

Pemrosesan gloss yang dilakukan pada baris 1-18 segmen program 5.14 sama dengan pemrosesan gloss pada segmen program 5.12 hanya saja terdapat sebuah tambahan pemeriksaan. Pada baris 16 dilakukan pemeriksaan apakah synset tersebut memiliki relasi. Jika ternyata synset tersebut tidak memiliki relasi sama sekali maka pada baris 17 akan ditambahkan sebuah keterangan bahwa tidak terdapat relasi pada synset tersebut. Relasi-relasi yang dimiliki oleh synset pertama-tama akan dikelompokkan terlebih dahulu, kata-kata yang memiliki relasi hipernim akan disimpan ke dalam sebuah dictionary, begitu juga dengan jenis relasi lain. Proses pengelompokkan ini ditunjukkan segmen program 5.15.

Segmen Program 5.15 Pengelompokkan relasi

```
1:   NSMutableDictionary *listOfRelation = [[NSMutableDictionary
     alloc]init];
2:   for(int z=0;z<[[synset synsetRelations]count];z++)
3:   {
4:     NSString *relType,*relOffset,*ori,*tar,*wordClass;

5:     relType=[[[synset synsetRelations]
       objectAtIndex:z]synsetRelationType];
6:     relOffset=[[[synset synsetRelations]
       objectAtIndex:z]synsetRelationOffset];
7:     wordClass = [[[synset synsetRelations]
       objectAtIndex:z]synsetTargetClass];
8:     ori=[[[synset
       synsetRelations]objectAtIndex:z]synsetLemmaOriginalIndex];
9:     tar=[[[synset
       synsetRelations]objectAtIndex:z]synsetLemmaTargetIndex];

10:    NSString *type = [[NSString alloc]init];
11:    if ([wordClass isEqualToString:@"n"])
12:       type = @"noun";
```

Segmen Program 5.15 (Lanjutan)

```
13:    if ([wordClass isEqualToString:@"v"])
14:        type = @"verb";
15:    if ([wordClass isEqualToString:@"a"])
16:        type = @"adj";
17:    if ([wordClass isEqualToString:@"r"])
18:        type = @"adv";
19:    if([listOfRelation objectForKey:relType])
20:    {
21:        NSString *opp = [[NSString alloc]init];
22:        if(![ori isEqualToString:@"00"] &&
              ![tar isEqualToString:@"00"])
23:             opp=[opp stringByAppendingFormat:@"
                 %@ [Opposed by : %@]",
                 [WN getSynsetLemma:relOffset forDict:type
                 forIndex:tar],
                 [WN getSynsetLemma:[listOfSynsets objectAtIndex:i]
                 forDict:type forIndex:ori]];
24:        else
25:             opp=[opp stringByAppendingFormat:@"%@",
                 [WN getSynsetLemma:relOffset forDict:type
                 forIndex:tar]];

26:        NSString *temp = [[NSString alloc]init];
27:        temp = [temp stringByAppendingFormat:@"%@;%@"
           ,[listOfRelation objectForKey:relType],[opp retain]];

28:        [listOfRelation setObject:[temp retain]
           forKey:relType];
29:        [temp release];
30:        [opp release];
31:    }
32:  else {
33:        NSString *opp = [[NSString alloc]init];
34:        if(![ori isEqualToString:@"00"] && ![tar
           isEqualToString:@"00"])
35:             opp=[opp stringByAppendingFormat:@"
                 %@ [Opposed by : %@]",
                 [WN getSynsetLemma:relOffset forDict:type
                 forIndex:tar],
                 [WN getSynsetLemma:[listOfSynsets objectAtIndex:i]
                 forDict:type forIndex:ori]];
36:        else
37:             opp=[opp stringByAppendingFormat:@"%@",
                 [WN getSynsetLemma:relOffset forDict:type
                 forIndex:tar]];

38:        [listOfRelation setObject:[opp retain] forKey:relType];
39:        [opp release];
40:    }
41:  }
```

Pada baris 2 segmen program 5.15 dilakukan iterasi terhadap semua relasi yang dimiliki oleh synset. Untuk setiap relasi akan diambil informasi mengenai

jenis relasi, synset offset pemilik relasi, index lemma pemilik relasi, serta class kata dari synset offset pemilik relasi, informasi tersebut akan digunakan untuk mengelompokkan relasi-relasi yang ada. Proses pengambilan informasi ini dilakukan oleh baris 4-18. Setiap relasi akan menjadi sebuah key value pada dictionary. Sedangkan kata pemilik relasi akan disimpan pada valuenya. Pada baris 19 dilakukan pemeriksaan apakah dictionary telah memiliki relasi yang sedang diperiksa sebagai sebuah key. Apabila ternyata belum dimiliki maka dictionary akan membuatkan sebuah key untuk relasi tersebut.

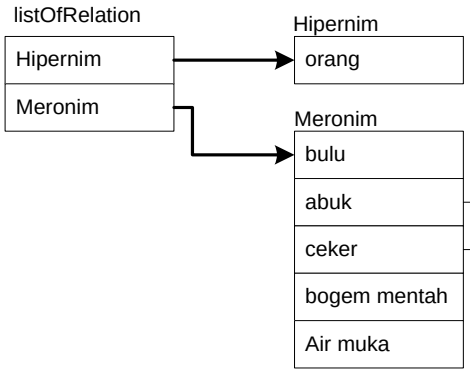
Baris 21-25 dan baris 33-37 merupakan proses yang sama. Proses yang dilakukan adalah melakukan pemeriksaan apakah relasi yang sedang diperiksa merupakan relasi semantik. Relasi semantik ditandakan oleh nilai 00 pada index lemma pemilik relasi pada kedua synset yang berhubungan, proses pemeriksaan ini dilakukan oleh 22 dan baris 34. Apabila keduanya adalah relasi semantik maka lemma dari pemilik relasi akan didapatkan dan disimpan secara langsung ke dalam dictionary. Proses mendapatkan lemma dilakukan dengan memanfaatkan prosedur milik class WordNet berikut.

```
[WN getSynsetLemma:[listOfSynsets objectAtIndex:i] forDict:type
                        forIndex:ori]];
```

Prosedur getSynsetLemma akan menerima parameter berupa synset offset ,class kata tempat synset berada dan index yang ingin diambil. Apabila index yang dikirimkan bernilai 00 maka lemma sinonim pertama akan diambil. Proses ini ditunjukkan oleh baris 25 dan baris 37. Tetapi apabila relasi adalah relasi leksikal maka yang disimpan bukan hanya lemma pemilik relasi tetapi juga lemma dari sinonim kata yang memiliki relasi tersebut. Pengambilan lemma tersebut ditunjukkan oleh baris 23 dan 35.

Apabila relasi yang diperiksa sudah berada pada key dictionary maka penyimpanan akan dilakukan dengan menggunakan baris 27 dan 28. Dimana pada baris 27, kata-kata lain yang memiliki relasi yang sama akan disimpan kembali dan dipisahkan dengan menggunakan tanda titik koma (;). Sedangkan bila relasi

belum berada pada key dictionary maka penyimpanan dilakukan oleh baris 38. Dimana kata dan jenis relasi akan disimpan ke dalam dictionary.



Gambar 5.16
Hasil Pengelompokkan Relasi

Gambar 5.16 menunjukkan dictionary hasil dari pengelompokkan relasi milik kata manusia. Terdapat dua jenis relasi yang dimiliki yaitu hipernim dan meronim. Setiap relasi tersebut memiliki kata yang berhubungan dengan relasi tersebut. Hipernim berhubungan dengan kata orang. Meronim berhubungan dengan kata bulu, abuk, ceker, bogem mentah, dan air muka. Setelah pengelompokkan ini selesai dilakukan maka yang harus dilakukan adalah memprosesnya menjadi tag-tag HTML.

Segmen Program 5. 16 Memproses Tampilan Relasi

```
1:   for(NSString *keys in [listOfRelation allKeys])
2:   {
3:     counter+=1;
4:     message = [message stringByAppendingFormat:@"
       <div id='%i' class='Relation' onClick='xRelation(this.id)'>
          <span id='xsign%i'
           style='font-family:courier new;font-weight:bold;'>
          [+]
           </span>%@
       </div>",counter,counter,keys];

5:     message = [message stringByAppendingFormat:@"
       <div id='RelationData%i' style='display:none;'>",counter];
6:     NSArray *temp = [[listOfRelation objectForKey:keys]
       componentsSeparatedByString:@";"];
7:     for(NSString *value in temp)
```

Segmen Program 5.16 (Lanjutan)

```
8:     {
9:       if ([value rangeOfString:@"["].location ==NSNotFound)
10:      {
11:        message = [message stringByAppendingFormat:@"
           <a href='tacb://%@' style='padding-left:50px;'>%@</a>
           <br>",value,value];
12:      }
13:      else {
14:        NSString *search = [[NSString alloc]init];
15:        NSString *search2 = [[NSString alloc]init];
16:        NSRange pos = [value rangeOfString:@"["];
17:        NSRange pos2 = [value rangeOfString:@"]"];

18:        search= [value substringWithRange:NSMakeRange(0,
           pos.location)];
19:        search2= [value
           substringWithRange:NSMakeRange(pos.location+1,
           (pos2.location-pos.location)-1)];

20:        message = [message stringByAppendingFormat:@"
           <a href='tacb://%@' style='padding-left:50px;'>%@</a>
           <br>
           <span style='padding-left:52px;'>
             [Opposed by : <a href='tacb://%@'>%@</a>]
           </span>
           <br>",[search retain],[search retain],[search2
           retain],[search2 retain]];

21:        [search release];
22:        [search2 release];
23:      }
24:    }
25:    message = [message stringByAppendingString:@"</div>"];
26:  }
```

Pada baris 1 dilakukan iterasi untuk setiap relasi yang tersimpan pada dictionary. Untuk setiap relasi tersebut dibuat sebuah div yang akan menampung kata-kata yang memiliki relasi tersebut, pembuatan div ditunjukkan oleh baris 4 dan 5. Pada baris 6 setiap kata yang memiliki relasi akan diambil dengan memotong string berdasarkan karakter titik koma (;) dan pada baris 7 setiap kata tersebut akan mengalami iterasi. Untuk setiap kata tersebut akan diperiksa apakah relasi yang dimiliki adalah relasi leksikal, pemeriksaan dilakukan dengan memeriksa apakah terdapakat karakter “[“ pada string. Jika relasi adalah relasi semantik maka kata tersebut akan langsung dimasukkan ke dalam div sebagai link. Pemeriksaan dan penyimpanan kata dilakukan oleh baris 9 dan 11. Sedangkan apabila relasi bersifat semantik maka akan terdapat dua link kata, link

pertama adalah kata pemilik relasi dan link kedua lemma sinonim kata yang memiliki relasi tersebut.Pada baris 14-19 dilakukan pemotongan string untuk mendapatkan kata yand diapit oleh simbol “[ ]”. Setelah kata tersebut didapatkan maka pada baris 20 dilakukan pembuatan kedua link kata tersebut. Pada baris 25 diberikan tag penutup div. Hasil dari proses ini ditunjukkan oleh gambar 5.17 dimana ditunjukkan contoh dari relasi semantik dan relasi leksikal.

(a) **(b)**

Gambar 5.17
Relasi Kata
(a). Relasi Semantik
(b). Relasi Leksikal

Gambar 5.17(a) menunjukkan relasi-relasi semantik dimana relasi berlaku untuk semua sinonim pada synset. Sedangkan gambar 5.17(b) menunjukkan relasi leksikal dan semantik. Antonim merupakan sebuah relasi leksikal dimana relasi ini dimiliki oleh kata monogami dan kata poligami yang merupakan lemma sinonim dari kata permaduan. Hiponim merupakan sebuah relasi semantik dimana relasi ini berlaku untuk semua lemma sinonim. Langkah terakhir dari pemrosesan tampilan adalah pemberian tag penutup dan mengganti karakter garis bawah (_) dengan karakter spasi. Pemrosesan tag penutup ini ditunjukkan oleh segmen program 5.17.

Segmen Program 5.17 Memproses Tag Penutup

```
1:    if(i==[listOfSynsets count]-1)
2:    {
3:     if (![setting groupByClass])
4:     {
5:      message = [message stringByAppendingString:@" </div>"];
6:      message = [message stringByAppendingString:@" </body>"];
7:      message = [message stringByAppendingString:@"</html>"];
8:     }
9:     else {
10:     message = [message stringByAppendingString:@"</div>"];
11:    }
12:   }
13:  }
14:  message = [message stringByReplacingOccurrencesOfString:@"_"
     withString:@" "];
```

Pada baris 1 dilakukan pemeriksaan apabila sudah mencapai synset terakhir maka akan diberikan tag penutup. Tetapi pada baris 3 diperiksa apakah user menginginkan fitur groupByClass dijalankan, bila tidak maka akan diberikan tag penutup untuk body dan html. Bila fitur diaktifkan maka hanya diberikan penutup div agar class kata berikutnya dapat diletakkan pada satu HTML yang sama.

5.3.3.1 Memberikan Tanda pada Bookmark Button

Setelah tampilan selesai diproses maka terdapat satu hal lagi yang perlu dilakukan yaitu memberikan tanda pada bookmark button. Apabila sebuah kata sudah dibookmark maka tombol tersebut akan menyala, sedangkan bila kata tersebut belum dibookmark maka tombol tidak akan menyala. Segmen program 5.18 memberikan efek menyala pada bookmark button.

Segmen Program 5.18 Memberikan Style pada Button

```
1: if ([isBookmarked isEqual:@"YES"])
2:   [bookmarkButton setStyle:UIBarButtonItemStyleDone];
3: else
4:   [bookmarkButton setStyle:UIBarButtonItemStyleBordered];
```

Pada baris 1 segmen program 5.18 dilakukan pemeriksaan apakah kata yang sedang diperiksa merupakan kata yang berada pada bookmark, jika ya maka baris 2 akan dijalankan dimana style dari button akan diubah menjadi menyala.

Jika tidak maka baris 4 akan dijalankan dimana button akan diubah menjadi tidak menyala. Pemeriksaan kata yang berada pada bookmark dilakukan oleh prosedur isInBookmark yang ditunjukkan oleh segmen program 5.19.

Segmen Program 5.19 Prosedur isInBookmark

```
1:  -(BOOL) isInBookmark : (id) words
2:  {
3:   for(NSString *w in listOfBookmarkedWords)
4:   {
5:     if ([[w lowercaseString] isEqualToString:[words
       lowercaseString] ])
6:       return YES;
7:   }
8:   return NO;
9:  }
```

Baris 3 segmen program 5.19 menunjukkan iterasi terhadap semua kata yang telah dibookmark oleh user. Setiap kata tersebut akan mengalami pemeriksaan apabila kata yang menjadi parameter bernilai sama dengan kata yang dibookmark oleh user maka akan dikembalikan nilai YES.

(a) **(b)**

Gambar 5.18
Tampilan Bookmark
(a). Bookmark Button Tidak Menyala
(b). Bookmark Button Menyala

Gambar 5.18 menunjukkan kondisi dimana bookmark button menyala dan tidak menyala. Gambar 5.18(a) menunjukkan bookmark button yang tidak menyala, menandakan bahwa kata yang sedang dilihat belum dibookmark. Sedangkan gambar 5.18(b) menunjukkan bookmark button yang menyala, menandakan bahwa kata yang dilihat sudah dibookmark.

5.3.3.2 Memberikan Tampilan Fitur Suggestion.

Fitur suggestion memiliki tampilan yang berbeda dengan tampilan informasi utama. Terdapat dua jenis tampilan yang dimiliki oleh fitur suggestion, tampilan pertama adalah tampilan loading yang ditampilkan ketika penghitungan Levenshtein Distance sedang dilakukan dan tampilan kedua adalah tampilan yang mengandung daftar kata yang memiliki kemiripan dengan kata yang dicari. Gambar 5.19 menunjukkan tampilan ketika loading.

Gambar 5.19
Tampilan Loading Levenshtein Distance

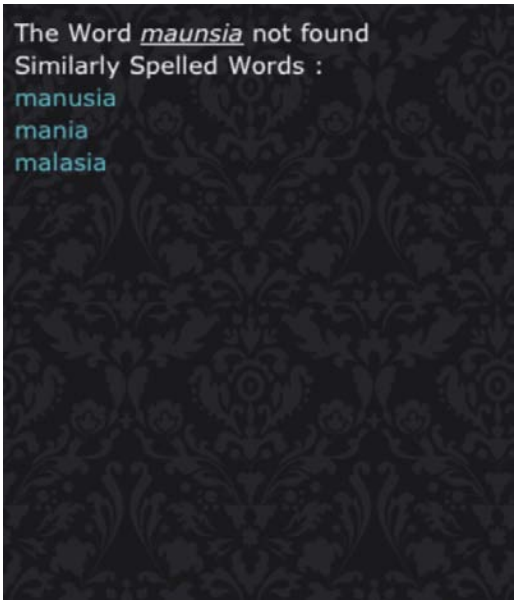
Gambar 5.19 menunjukkan tampilan ketika Levenshetin sedang dijalankan pada background process. Tampilan yang dihasilkan ini juga memanfaatkan HTML serta CSS.

Segmen Program 5.20 Memproses Tampilan Loading Levenshtein Distance

```
1:  if (![[word noun]synsets] && ![[word verb]synsets] &&  ![[word
    adj]synsets] && ![[word adv]synsets])
2:  {
3:  bookmarkButton.enabled = NO;

4:  NSString *message = [[NSString alloc]init];
5:  message = [message stringByAppendingString:@"<html>"];
6:  message = [message stringByAppendingString:@"<head>"];
7:  message = [message stringByAppendingString:@"
               <style type='text/css'>"];
8:  message = [message stringByAppendingString:@"
            body{
                 background-image:url('bg.png');
                 background-color:black;
                 color:white;
                 font-family:'verdana';
                 font-size:15px;
                 -webkit-touch-callout: none;
                }"];
9:  message = [message stringByAppendingString:@"
           img{position:relative;top:110px;left:130px;}"];
10: message = [message stringByAppendingString:@"
           #Search{position:relative;top:90px;left:120px;}"];
11: message = [message stringByAppendingString:@"
           </style>"];
12: message = [message stringByAppendingString:@"</head>"];
13: message = [message stringByAppendingString:@"<body>"];
14: message = [message stringByAppendingString:@"
          <div id='Search'>Searching</div>"];
15: message = [message stringByAppendingFormat:@"
          <img src='loadingAnimation.gif'></img>"];
16: message = [message stringByAppendingString:@"</body>"];
17: message = [message stringByAppendingString:@"</html>"];

18: [page1 loadHTMLString:message baseURL:files];
19: [self startPreProcessing];
20: }
```

Pada baris 1 segmen program 5.20 dilakukan pemeriksaan apakah kata yang dicari memiliki synset pada class kata manapun. Jika ternyata kata tersebut tidak memiliki synset pada class kata manapun maka Levenshtein Distance akan dijalankan untuk memperoleh kata-kata yang memiliki kemiripan penulisan. Pada baris 7-11 disiapkan sebuah CSS yang akan digunakan oleh tampilan loading dan pada baris 14 dan 15 dideklarasikan sebuah div dengan tulisan “Searching” dan image loading. Pada baris 18 tampilan ini akan ditampilkan pada user. Setelah tampilan telah siap maka Levenshtein Distance akan dijalankan pada background process, proses ini ditunjukkan oleh baris 19.

Ketika Levenshtein Distance selesai dijalankan maka dibuat sebuah HTML baru yang berisi daftar kata yang memiliki kemiripan penulisan dengan kata yang dicari oleh user. Gambar 5.20 menunjukkan daftar kata yang dihasilkan oleh Levenshtein Distance.



Gambar 5.20
Daftar Kata Hasil Levenshtein Distance

Pada gambar 5.20 ditunjukkan bahwa Levenshtein Distance menghasilkan tiga buah kata yang mirip dengan kata manusia. Kata-kata tersebut akan berupa link sehingga user dapat langsung melakukan tap pada kata tersebut untuk mengetahu informasi dari kata tersebut. Segmen program 5.21 menunjukkan proses untuk menghasilkan tampilan pada gambar 5.20.

Segmen Program 5.21 Memproses Tampilan Hasil Levenshtein Distance

```
1:  message = [message stringByAppendingString:@"<html>"];
2:  message = [message stringByAppendingString:@"<head>"];
3:  message = [message stringByAppendingString:@"
              <style type='text/css'>"];
4:  message = [message stringByAppendingString:@"
              a{color:#62BECB;text-decoration:none;}"];
5:  message = [message stringByAppendingString:@"
              #data{width:300px;}"];
```

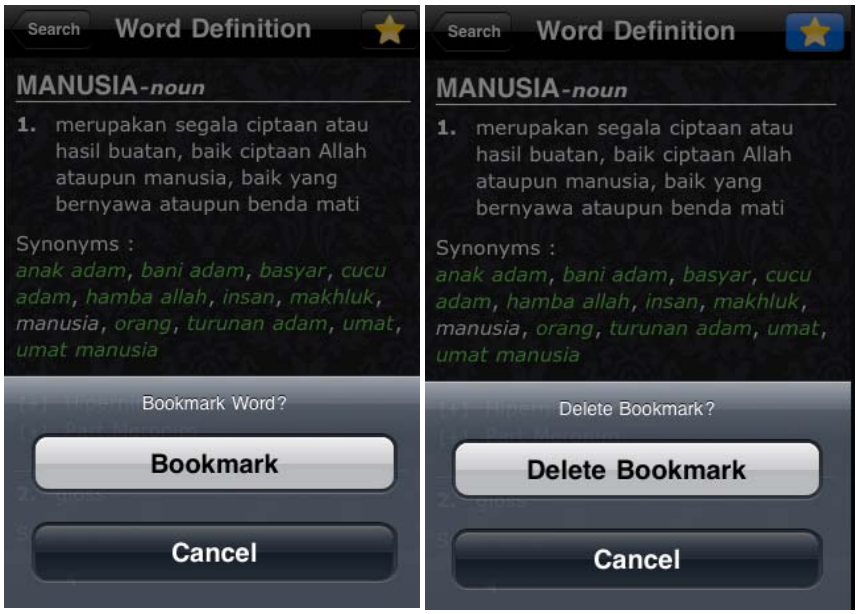
Segmen Program 5.21 (Lanjutan)

```
6:  message = [message stringByAppendingString:@"
            body{
                  background-image:url('bg.png');
                  background-color:black;
                  color:white;
                  font-family:'verdana';
                  font-size:15px;
                  -webkit-touch-callout: none;
            }"];
7:  message = [message stringByAppendingString:@"</style>"];
8:  message = [message stringByAppendingString:@"</head>"];
9:  message = [message stringByAppendingString:@"<body>"];
10: message = [message stringByAppendingString:@"
               <div id='data'>"];

11: message = [message stringByAppendingFormat:@"
    The Word <i><u>%@</u></i> not found <br/>
    Similarly Spelled Words : <br>",[word lemma]];

12: NSSortDescriptor *descriptors=[[[NSSortDescriptor
    alloc]initWithKey:@"a" ascending:YES]autorelease];
13: NSArray *sortdescriptor=[NSArray arrayWithObject:descriptors];
14: NSArray *sortedarray=[nearest
    sortedArrayUsingDescriptors:sortdescriptor];

15: for(NSMutableDictionary *dict in [sortedarray
    reverseObjectEnumerator])
16: {
17:     for (NSString *str in [dict allKeys])
18:         message = [message stringByAppendingFormat:@"
            <a href=\"tacb://%@\">%@</a><br>",str ,str];
19: }

20: message = [message stringByAppendingString:@"</div>"];
21: message = [message stringByAppendingString:@"</body>"];
22: message = [message stringByAppendingString:@"</html>"];
```

Baris 3-7 segmen program 5.21 menunjukkan CSS yang akan digunakan oleh halaman hasil Levenshtein Distance. Pada baris 11 diberikan pesan bahwa kata yang dicari tidak ditemukan. Pada baris 12 dibuat sebuah NSSortDescriptor, variabel tersebut berperan untuk melakukan sorting terhadap dictionary nearest. Dimana sorting akan dilakukan berdasarkan biaya yang diperlukan untuk merubah kata, proses ini ditunjukkan oleh baris 12-14. Pada baris 15 dilakukan iterasi terhadap dictionary yang mengandung hasil sorting. Untuk setiap dictionary dilakukan iterasi lagi terhadap key yang dimiliki dan dibuat sebuah link berupa ahref untuk key tersebut. Proses ini ditunjukkan oleh baris 17 dan 18. Pada baris 20-22 dibuat tag-tag HTML penutup.

5.3.4 Memberi / Menghapus Tanda pada Kata

User dapat member atau menghapus tanda (bookmark) pada kata yang sedang dicari oleh user. Pemberian atau penghilangan tanda tersebut dilakukan dengan menekan bookmark button yang terdapat pada posisi kanan atas. Ketika button ditekan maka akan muncul sebuah action sheet yang meminta konfirmasi user. Gambar 5.21 menunjukkan action sheet tersebut.



(a) (b)

Gambar 5.21
Tampilan Action Sheet
(a). Add Bookmark
(b). Delete Bookmark

Kemunculan action sheet pada gambar 5.21 diatur oleh sebuah prosedur bernama bookmarked. Prosedur tersebut akan mengatur kata-kata yang muncul pada action sheet, sedangkan untuk penyimpanan dan penghapusan kata pada daftar bookmark akan ditangani oleh prosedur lain. Prosedur bookmarked ini akan dipanggil ketika user melakukan penekanan bookmark button, prosedur akan mendeteksi kondisi kata dan memunculkan actionsheet yang sesuai dengan kondisi kata. Segmen program 5.22 menunjukkan detail dari prosedur bookmarked.

Segmen Program 5.22 Prosedur Bookmarked

```
1:  -(IBAction) bookmarked
2:  {
3:   UIActionSheet *bookmarkSheet;
4:   if ([isBookmarked isEqual:@"NO"])
5:      bookmarkSheet = [[UIActionSheet alloc]
       initWithTitle:@"Bookmark Word?"
       delegate:self
       cancelButtonTitle:@"Cancel"
       destructiveButtonTitle:nil
       otherButtonTitles:@"Bookmark",nil];
6:   else
7:      bookmarkSheet = [[UIActionSheet alloc]
       initWithTitle:@"Delete Bookmark?"
       delegate:self
       cancelButtonTitle:@"Cancel"
       destructiveButtonTitle:nil
       otherButtonTitles:@"Delete Bookmark",nil];

8:   [bookmarkSheet showInView:self.view];
9:   [bookmarkSheet release];
10: }
```

Baris 4 segmen program 5.22 akan melakukan pemeriksaan apakah kata sudah dibookmar oleh user. Jika sudah maka baris 5 akan dijalankan dimana action sheet akan mendapat pertanyaan “Bookmark Word?” tetapi jika kata sudah dibookmark maka akan dijalankan baris ke 7 dimana action sheet akan mendapat pertanyaan “Delete Bookmark?”. Prosedur bookmarked ini tidak menangani penekanan tombol yang ada pada action sheet. Untuk menangani penekanan tombol tersebut maka dioverride sebuah prosedur bernama clickedButtonAtIndex. Prosedur tersebut ditunjukkan oleh segmen program 5.23.

Segmen Program 5.23 Prosedur clickedButtonAtIndex

```
1:  (void)actionSheet:(UIActionSheet *)actionSheet
    clickedButtonAtIndex:(NSInteger)buttonIndex{
2:    if (buttonIndex == 0)
3:    {
4:     if([isBookmarked isEqual:@"NO"])
5:     {
6:      [[NSNotificationCenter defaultCenter]
        postNotificationName:@"addToBookmark"
        object: [[word lemma]
        stringByReplacingOccurrencesOfString:@" "
        withString:@"_"]];
7:      [bookmarkButton setStyle:UIBarButtonItemStyleDone];
8:      self.isBookmarked = @"YES";
9:     }
10:    else
```

Segmen Program 5.23 (Lanjutan)

```
11:     {
12:      [[NSNotificationCenter defaultCenter]
         postNotificationName:@"deleteBookmark" object: [[word
         lemma] stringByReplacingOccurrencesOfString:@" "
         withString:@"_"]];

13:     [bookmarkButton setStyle:UIBarButtonItemStyleBordered];
14:     self.isBookmarked = @"NO";
15:     }
16:   }
17: }
```

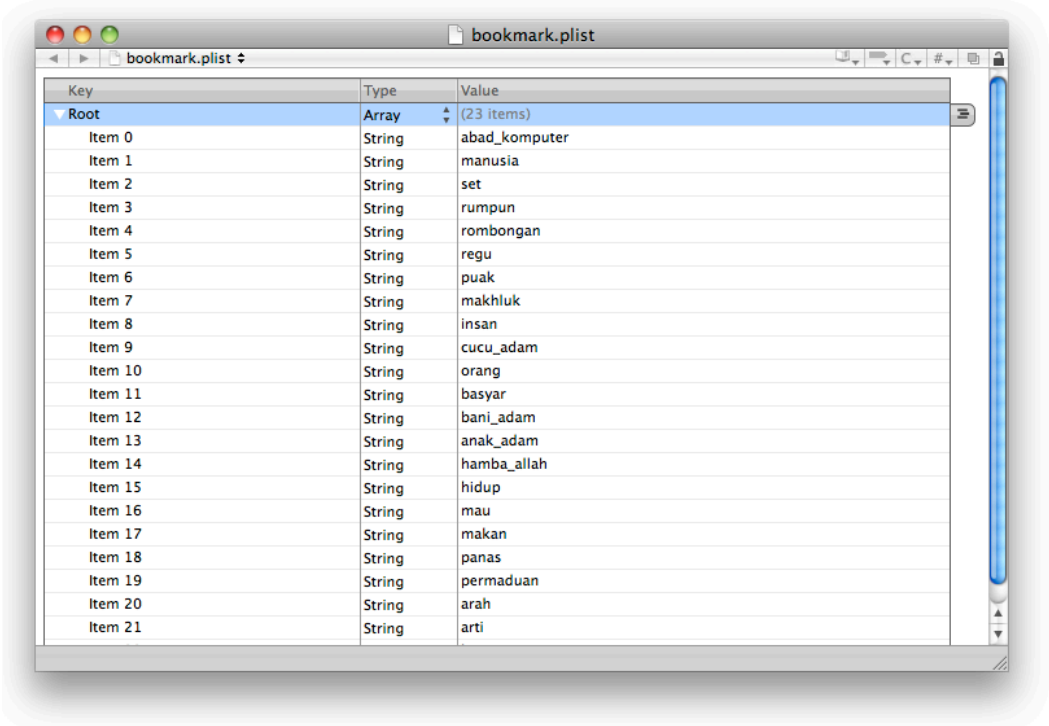
Pada baris 2 segmen program 5.23 dilakukan pemeriksaan apakah tombol yang ditekan adalah tombol yang berada pada posisi pertama, yang berarti tombol tersebut bukan lah tombol cancel. Jika ya maka pada baris 4 dilakukan pemeriksaan lagi apakah kata sudah dibookmark oleh user, jika belum berarti akan dijalankan prosedur addToBookmark pada baris 6. Sedangkan jika sudah berarti akan dijalankan prosedur deleteBookmark pada baris 12. Kedua prosedur tersebut memiliki parameter kata yang sedang dicari user sebagai parameter dimana karakter spasi digantikan oleh karakter garis bawah (_). Pada baris ke 7 dilakukan perubahan style button menjadi menyala, dan kondisi kata akan dirubah menjadi kata yang sudah dibookmark pada baris 8. Sedangkan pada baris 13 style button akan menjadi tidak menyala dan pada baris 14 kondisi kata dirubah menjadi kata yang tidak dibookmark. Prosedur addToBookmark adalah sebuah prosedur yang berfungsi untuk menambahkan sebuah kata ke dalam daftar kata yang dibookmark oleh user. Daftar kata tersebut kemudian akan disimpan ke dalam bookmark.plist dan disimpan pada folder document device.

Segmen Program 5.24 Prosedur addToBookmark

```
1: -(void) addToBookmark : (id) words
2: {
3:   [listOfBookmarkedWords addObject:[words object]];
4:   [listOfBookmarkedWords writeToFile:[[self documentPath]
     stringByAppendingPathComponent:@"/bookmark.plist"]
     atomically:YES];
5: }
```

Baris 3 segmen program 5.24 menunjukkan proses memasukkan kata ke dalam array yang menampung daftar kata yang dibookmark oleh user. Pada baris

4 array tersebut akan disimpan ke dalam bookmark.plist pada folder document milik device. File bookmark.plist ditunjukkan oleh gambar 5.22.



Gambar 5.22
File Bookmark.plist

Prosedur deleteBookmark merupakan sebuah prosedur yang berfungsi untuk menghapus sebuah kata yang ada didalam daftar kata yang dibookmark oleh user. Perubahan tersebut akan disimpan pada file bookmark.plist. Segmen program 5.25 menunjukkan prosedur deleteBookmark.

Segmen Program 5.25 Prosedur deleteBookmark

```
1:  -(void) deleteBookmark : (id) words
2:  {
3:     int x=0;
4:     for(NSString *w in listOfBookmarkedWords)
5:     {
6:      if ([[w lowercaseString] isEqualToString:[[words object]
        lowercaseString] ])
7:          break;
8:      x+=1;
9:     }
```

Segmen Program 5.25 (Lanjutan)

```
10:   [listOfBookmarkedWords removeObjectAtIndex:x];
11:   [listOfBookmarkedWords writeToFile:[[self documentPath]
      stringByAppendingPathComponent:@"/bookmark.plist"]
      atomically:YES];
12: }
```

Pada baris 4 segmen program 5.25 dilakukan iterasi terhadap semua kata yang dibookmark oleh user. Setiap kata akan mengalami pemeriksaan apakah katat tersebut sama dengan yang dicari oleh user, jika ya maka iterasi dihentikan. Tetapi jika tidak maka variabel x akan mengalami penambahan nilai. Proses ini ditunjukkan oleh baris 6-8. Pada baris 10 dilakukan penghapusan kata pada index yang didapatkan, yaitu variabel x. Kemudian pada baris 11 perubahan tersebut akan disimpan ke dalam file plist.

5.3.5 Navigasi Kata

WordNet browser menyediakan fitur navigasi kata dimana user dapat melakukan back dan next pada kata yang pernah dicari. Untuk melakukannya user cukup menekan tombol back dan next yang ada pada bagian bawah. Penekanan tombol back akan menyebabkan prosedur back dipanggil. Prosedur back ditunjukkan oleh segmen program 5.26.

Segmen Program 5.26 Prosedur Back

```
1:  -(void) back
2:  {
3:   back=YES;
4:   if(indexHistory == [activeList count])
5:     indexHistory-=1;
6:   if (indexHistory > 0)
7:     indexHistory-=1;
8:   lastSelectedTab = -1;
9:   Word *w = [WN getWordData:[[activeList
     objectAtIndex:indexHistory]
     stringByReplacingOccurrencesOfString:@" " withString:@"_"]];
10:  [self openResult:w];
11: }
```

Pada baris 4 segmen program 5.26 dilakukan pemeriksaan apakah index kata bernilai sama dengan jumlah kata yang dapat dilihat. Jika ya maka index akan dikurangi satu karena index dimulai dari 0. Pada baris 6 dilakukan

pemeriksaan apakah kata sudah mencapai index pertama, jika belum maka index akan dikurangi satu. Kemudian kata pada index tersebut akan diambil dan dilakukan ekstraksi synset offsetn dengan menggunakan prosedur getWordData milik class WordNet, tetapi sebelumnya karakter spasi akan digantikan dengan karakter garis bawah (_). Proses ekstraksi ini ditunjukkan oleh baris 9. Pada baris 10 hasil ekstraksi kemudian akan dikirimkan pada halaman result.

Prosedur forward memiliki kemiripan alur dengan prosedur back. Prosedur ini dipanggil ketika tombol next ditekan oleh user. Segmen program 5.27 menunjukkan detail dari prosedur next.

Segmen Program 5.27 Prosedur Forward

```
1:  -(void) forward
2:  {
3:   next=YES;
4:   if (indexHistory !=[activeList count]-1)
5:    indexHistory+=1;

6:   lastSelectedTab = -1;

7:   Word *w = [WN getWordData:[[activeList
     objectAtIndex:indexHistory]
     stringByReplacingOccurrencesOfString:@" " withString:@"_"]];
8:   [self openResult:w];
9:  }
```

Pada baris 3 segmen program 5.27 dilakukan pemeriksaan apakah index kata sudah mencapai index terakhir, jika belum maka index akan ditambah satu. Kemudian kata pada index tersebut akan diambil dan dilakukan ekstraksi synset offsetn dengan menggunakan prosedur getWordData milik class WordNet, tetapi sebelumnya karakter spasi akan digantikan dengan karakter garis bawah (_). Proses ekstraksi ini ditunjukkan oleh baris 7. Pada baris 8 hasil ekstraksi kemudian akan dikirimkan pada halaman result. Ketika halaman result terbuka akan dilakukan pengecekan terhadap index kata. Apabila index sudah mencapai batas awal atau akhir maka tombol back atau next akan dinonaktifkan.

ActiveList merupakan sebuah pointer yang dapat mengarah pada array bookmark maupun array history. Perubahan pointer tersebut ditentukan oleh halaman yang dilihat oleh user sebelum mencapai halaman result. Apabila user

berasal dari halaman search atau halaman history maka activeList akan berisi pointer menuju array history, sedangkan jika user berasal dari bookmark maka activeList akan berisi pointer menuju array bookmark.

BAB VI
FITUR-FITUR TAMBAHAN WORDNET BROWSER

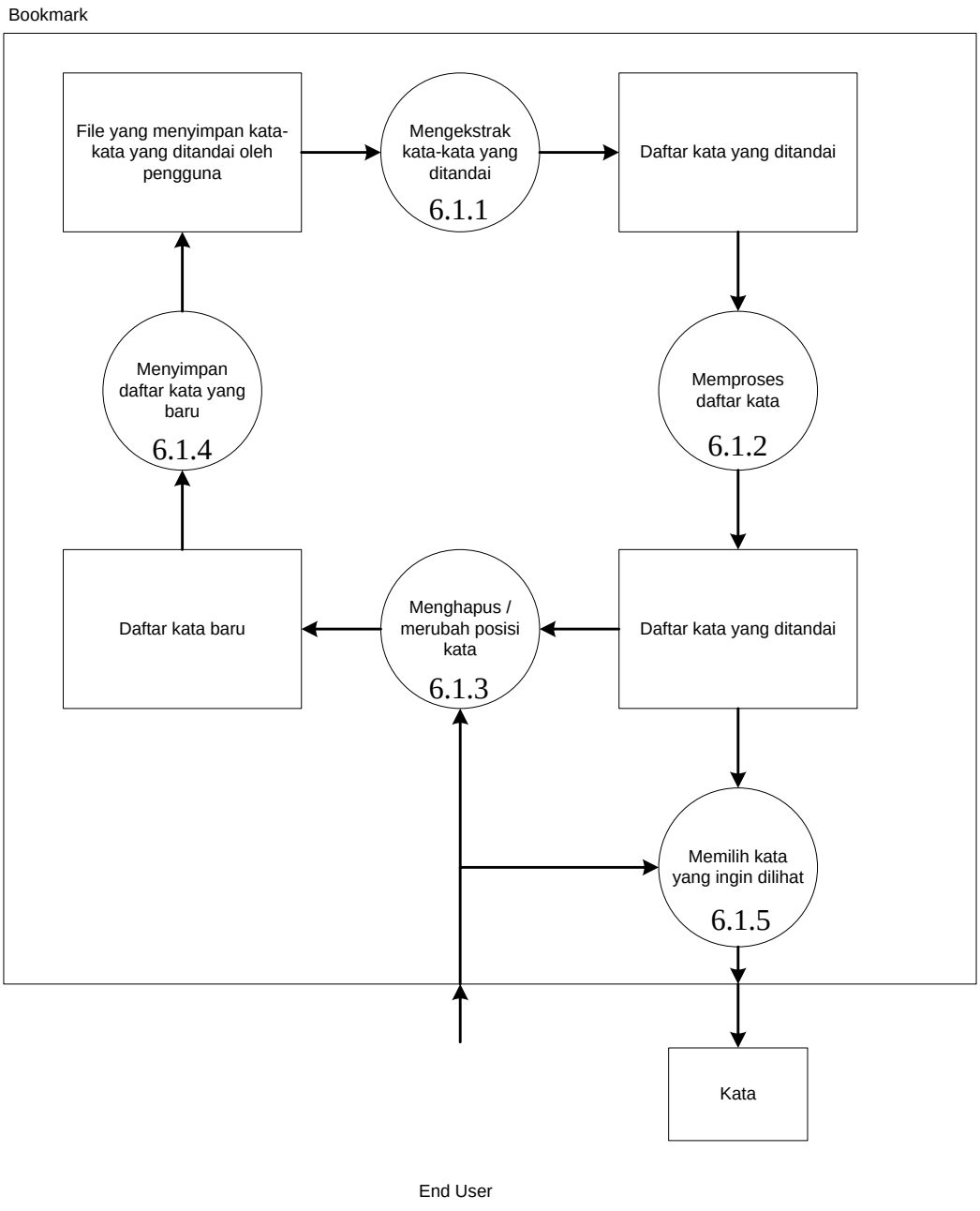
Dalam bab ini akan jelaskan mengenai fitur-fitur tambahan yang dimiliki oleh WordNet Browser. Fitur-fitur tersebut meliputi halaman bookmark, halaman history dan halaman setting. Pada halaman bookmark akan dijelaskan mengenai fitur perpindahan dan penghapusan kata, pada halaman history akan dijelaskan mengenai proses penampilan kata dan pada halaman setting akan dijelaskan mengenai fitur-fitur yang dapat diubah oleh pengguna. Penjelasan meliputi arsitektur sistem, input dan output sistem, proses-proses yang terjadi serta prosedur yang terlibat dalam pelaksanaan proses. Dalam proses penjelasan akan ditunjukkan juga hasil yang didapatkan dari suatu proses.

6.1 Halaman Bookmark

Halaman bookmark merupakan fitur dimana user dapat melihat kata-kata yang telah diberi tanda (dibookmark) serta melakukan perubahan pada kata-kata tersebut. Perubahan yang dapat dilakukan oleh user meliputi perubahan posisi kata serta penghapusan kata yang sudah tidak diperlukan. User juga dapat melihat informasi mengenai sebuah kata yang telah ditandai. Untuk melihat informasi yang dimiliki sebuah kata user cukup melakukan tap pada kata tersebut ketika tidak sedang berada pada mode edit.

Pada gambar 6.1 ditunjukkan arsitektur sistem dari halaman bookmark. Terdapat lima buah proses yang terjadi pada halaman bookmark dan setiap proses tersebut akan dijelaskan pada sebuah subbab tersendiri. Input dari sistem merupakan aksi yang dilakukan oleh user. Terdapat tiga jenis aksi yang dapat dilakukan yaitu mengubah posisi sebuah kata, menghapus kata yang ditandai, dan memilih kata yang ingin dilihat oleh user. Output dari sistem adalah sebuah kata yang ingin dilihat oleh user, kata tersebut akan diproses oleh class WordNet dan dikirimkan kepada halaman result. Sedangkan ketika user melakukan perubahan posisi atau menghapus sebuah kata yang ditandai maka browser akan melakukan

pencatatan terhadap perubahan yang terjadi dan menyimpan perubahan tersebut. Perubahan akan tersimpan ke dalam sebuah file yang mencatat daftar kata. Setelah perubahan telah tersimpan maka daftar kata yang ditampilkan kepada user akan mengalami pembaharuan.



Gambar 6.1
Arsitektur Sistem Halaman Bookmark

Penjelasan mengenai halaman bookmark akan diawali dengan penjelasan mengenai ekstraksi kata-kata yang ditandai dari file yang menyimpan kata-kata yang ditandai tersebut. Hasil ekstraksi ini nantinya akan diproses untuk ditampilkan kepada user dimana user dapat melakukan interaksi dengan daftar kata tersebut. Interaksi yang dimaksud adalah pengubahan posisi, penghapusan kata dan pemilihan kata.

6.1.1 Mengekstrak Kata-Kata yang Ditandai

Kata-Kata yang ditandai oleh user akan disimpan ke dalam sebuah plist bernama bookmark.plist. Kata-kata tersebut akan diambil dari file plist dan disimpan ke dalam sebuah array of string bernama listOfBookmarkedWords. Proses ini dilakukan pada halaman loading bersamaan dengan inisialisasi calss WordNet. Perintah yang digunakan adalah perintah berikut ini

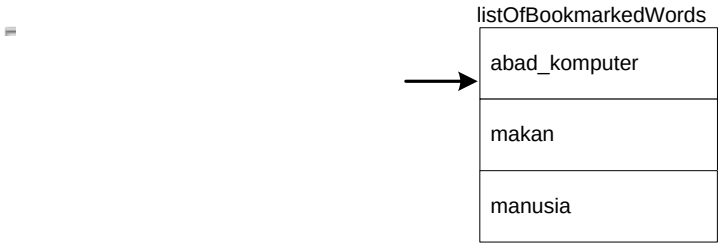
```
listOfBookmarkedWords = [self readFromFile:[[self documentPath]
stringByAppendingPathComponent:@"/bookmark.plist"]];
```

Untuk melakukan pengambilan kata yang telah ditandai digunakan sebuah prosedur bernama readFromFile. Prosedur ini menerima parameter berupa file path yang memiliki daftar kata yang diinginkan dan akan mengembalikan sebuah array yang telah mengandung kata-kata yang terdapata didalam file tersebut. Segmen program 6.1 menunjukkan detail prosedur readFromFile.

Segmen Program 6.1 Prosedur readFromFile Bookmark dan History

```
1:  -(id) readFromFile:(NSString *) filePath{
2:     NSMutableArray *Data;
3:     if ([[NSFileManager defaultManager]
       fileExistsAtPath:filePath]) {
4:        Data = [[NSMutableArray
          alloc]initWithContentsOfFile:filePath];
5:     }
6:     else
7:        Data = [[NSMutableArray alloc]init];
8:     return Data;
9:  }
```

Pada baris 3 segmen program 6.1 dilakukan pemeriksaan apakah file yang ingin diekstrak berada pada file path yang diterima sebagai parameter. Apabila ya maka pada baris 4 dilakukan pembacaan isi file dan disimpan ke dalam sebuah array. Tetapi bila ternyata tidak ada maka dijalankan baris 7 dimana disiapkan sebuah array kosong yang siap digunakan. Kondisi file bookmark tidak berada pada file path yang dikirimkan adalah pada saat browser pertama kali dijalankan dan user belum memberikan tanda pada kata apapun.



Gambar 6.2
Pemrosesan File Plist

Gambar 6.2 menunjukkan hasil ekstraksi file bookmark.plist menjadi sebuah array yang mengandung kata-kata yang ditandai oleh user. Pada tahap ini kata-kata yang didapatkan belum mengalami pemrosesan apapun untuk ditampilkan kepada user.

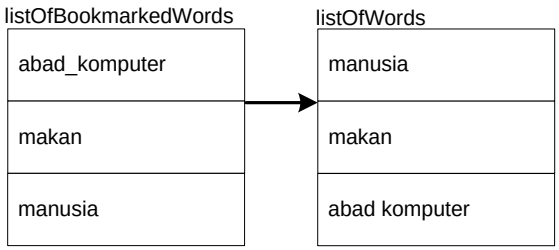
6.1.2 Memproses Daftar Kata

Setelah array berisi daftar kata yang ditandai didapatkan maka langkah selanjutnya adalah menampilkan daftar kata tersebut kepada user dengan menggunakan sebuah table view. Tetapi sebelum daftar kata tersebut ditampilkan maka terlebih dahulu harus dilakukan penggantian karakter garis bawah menjadi karakter spasi. Daftar kata yang tersimpan pada file bookmark.plist akan dibaca dan mengalami penggantian karakter, selain itu urutan dari kata tersebut akan dibalik. Pada file plist, kata yang terbaru akan berada pada posisi paling bawah dan ketika ditampilkan kata tersebut yang terbaru harus berada pada posisi pertama table view. Proses ini ditunjukkan oleh segmen program 6.2.

Segmen Program 6.2 Memproses Daftar Kata

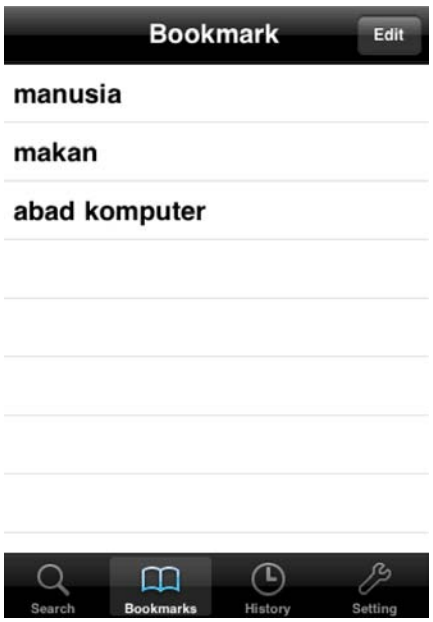
```
1: NSMutableArray *temp = [[NSMutableArray alloc]init];
2: for(int i=0;i<[listOfBookmarkedWords count];i++)
3: {
4:    [temp addObject:[[listOfBookmarkedWords objectAtIndex:i]
     stringByReplacingOccurrencesOfString:@"_" withString:@" "]];
5: }
6: temp = [[[temp reverseObjectEnumerator]
   allObjects]mutableCopy];
7: listOfWords = [temp retain];
8: [temp release];
9: [bookmarkTableView reloadData];
```

Baris 2 segmen program 6.2 menunjukkan iterasi terhadap semua kata yang ditandai oleh user. Setiap kata tersebut akan mengalami penggantian karakter garis bawah menjadi karakter spasi, kemudian kata yang mengalami penggantian tersebut akan disimpan ke dalam sebuah array bernama temp. Proses tersebut dilakukan pada baris 4. Array temp tersebut berfungsi untuk membalik urutan kata-kata yang dimiliki, kata yang terakhir kali ditandai oleh user akan ditampilkan pada baris pertama dari table view dan kata yang pertama kali ditandai akan diletakkan pada paris paling akhir dari table view. Setelah urutan kata dibalik maka kata-kata tersebut akan disimpan pada array listOfWords yang akan digunakan untuk memasukkan kata ke dalam cell milik table view. Proses pembalikan urutan kata dan penyimpanan pada array listOfWords ditunjukkan oleh baris 6 dan 7. Pada baris 8 isi dari table view akan diproses untuk ditampilkan. Gambar 6.3 menunjukkan pemrosesan kata yang didapat dari bookmark.plist menjadi aray listOfWords.



Gambar 6.3
Pemrosesan Daftar Kata

Setelah kata siap untuk ditampilkan kepada user maka langkah selanjutnya adalah membuat cell-cell berisi kata untuk dimasukkan ke dalam table view. Proses pembuatan cell tersebut sama dengan yang dijelaskan pada bab V subbab 5.2.2 yaitu menggunakan prosedur cellForRowAtIndexPath dan prosedur numberOfRowsInSection. Tetapi perbedaannya adalah array yang mengandung kata-kata yang akan ditampilkan bukanlah array searchResult melainkan array listOfWords. Gambar 6.4 menunjukkan hasil pembuatan cell dengan menggunakan array listOfWords sebagai sumber kata-kata yang akan ditampilkan kepada user.

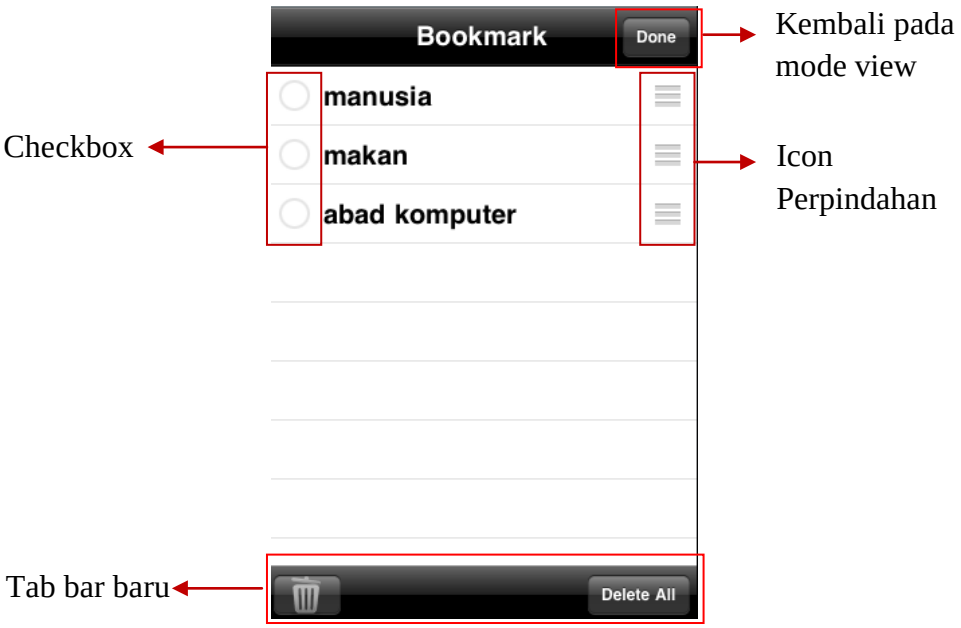


Gambar 6.4
Hasil Pemrosesan Array Kata

Setelah tampilan sudah siap dan telah ditampilkan kepada user, kini user dapat berinteraksi dengan halaman bookmark. Apabila user menekan tombol edit maka user dapat melakukan perubahan pada daftar kata pada halaman bookmark berupa pemindahan posisi kata dan penghapusan kata. Tetapi apabila user langsung memilih kata maka informasi dari kata tersebut akan ditampilkan pada user melalui halaman result. Proses perubahan posisi kata serta penghapusan posisi kata akan dijelaskan secara detail pada subbab 6.1.3.

6.1.3 Menghapus / Merubah Posisi Kata

Ketika tombol edit ditekan maka tampilan pada halaman bookmark akan berubah. Setiap cell pada table view akan memiliki sebuah checkbox pada sisi kiri dan sebuah icon perpindahan pada sisi kanan, selain itu tabbar akan berubah menjadi hanya dua buah tombol yaitu delete dan delete all. Bila user ingin mengaktifkan checkbox pada suatu cell maka user dapat melakukan tap pada checkbox tersebut. Icon yang berada pada sebelah kanan cell berguna untuk melakukan pemindahan posisi kata, dengan melakukan drag and drop pada icon perpindahan tersebut. Pada mode ini informasi dari kata tidak akan ditampilkan, untuk mengakhiri pengeditan maka user dapat menekan tombol done. Tampilan mode edit ini ditunjukkan oleh gambar 6.5.



Gambar 6.5
Mode Edit Halaman Bookmark

Perubahan tampilan tersebut ditangani oleh sebuah prosedur bernama editMode yang dijalankan ketika tombol edit ditekan pada mode view dan ketika tombol done ditekan pada mode edit. Ketika tombol edit ditekan maka prosedur akan memunculkan tampilan milik mode edit dan sebaliknya. Segmen program 6.3 menunjukkan detail dari prosedur editMode.

Segmen Program 6.3 Prosedur editMode

```
1:  -(IBAction) editMode
2:  {
3:    if (![bookmarkTableView isEditing])
4:    {
5:      [bookmarkTableView setEditing:YES animated:YES];
6:      [editButton setTitle:@"Done"];
7:      [[NSNotificationCenter defaultCenter]
        postNotificationName:@"hideTabBar" object: nil];
8:      [deleteBar setHidden:NO];
9:    }
10:   else {
11:     for(int i=[listOfBookmarkedWords count]-1;i>=0;i--)
12:     {
13:       NSIndexPath *path = [NSIndexPath indexPathForRow:i
          inSection:0];
14:       UITableViewCell *cell = [bookmarkTableView
          cellForRowAtIndexPath:path];
15:       cell.accessoryType = UITableViewCellAccessoryNone;
16:     }
17:     [bookmarkTableView setEditing:NO animated:YES];
18:     [editButton setTitle:@"Edit"];
19:     [[NSNotificationCenter defaultCenter]
        postNotificationName:@"showTabBar" object: nil];
20:     [deleteBar setHidden:YES];
21:   }
22: }
```

Pada baris 3 segmen program 6.3 dilakukan pemeriksaan apakah prosedur ini dijalankan ketika tombol edit ditekan atau ketika tombol done ditekan. Jika ternyata prosedur dipanggil ketika tombol edit ditekan maka baris 5 akan dijalankan dimana table view akan diubah menjadi mode edit, tombol edit menjadi tombol done, serta menghilangkan tab bar yang aktif. Proses tersebut dilakukan pada baris 5-7.Tetapi jika ternyata prosedur dijalankan ketika tombol done ditekan maka baris 11 akan dijalankan. Pada baris 11 dilakukan iterasi terhadap semua kata yang ditandai, dan untuk setiap cell yang mengandung kata-kata yang ditandai akan dihilangkan tanda check yang aktif. Hal ini untuk mengatasi kondisi dimana user mencentang checkbox pada cell dan menekan tombol done, apa bila checkbox tidak dinonaktifkan maka pada mode view tanda centang akan tetap muncul. Pada baris 13 section dari index path akan diisi dengan nilai nol karena tidak terdapat section pada halaman bookmark, kemudian pada baris 14 index path tersebut akan digunakan untuk mengambil cell dari table

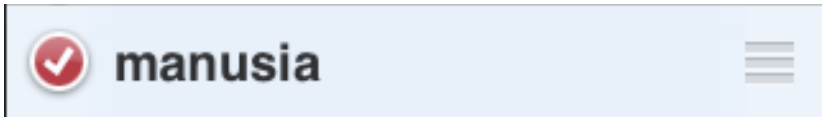
view. Setelah cell didapatkan pada baris 15 aksesoris dari cell akan dihilangkan. Pada baris 17-20 tampilan akan dikembalikan pada mode view.

Untuk bisa menampilkan sebuah checkbox dan icon perpindahan pada cell milik table view maka terdapat dua buah prosedur yang harus dioverride. Kedua prosedur tersebut ditunjukkan oleh tabel 6.1

Tabel 6.1
Prosedur Pembuatan Checkbox dan Icon Perpindahan

Prosedur	Return Value
canMoveRowAtIndexPath	YES
editingStyleForRowAtIndexPath	UITableViewCellAccessoryCheckmark

Prosedur canMoveRowAtIndexPath akan memunculkan icon perpindahan ketika return value dari prosedur bernilai YES. Sedangkan untuk mengubah icon delete menjadi checkbox maka prosedur editingStyleForRowAtIndexPath harus memiliki UITableViewCellAccessoryCheckmark sebagai return value.



Gambar 6.6
Cell dengan Checkbox dan Icon Perpindahan

Pada gambar 6.6 ditunjukkan hasil override dari kedua prosedur pada tabel 6.1. Aksesoris dari cell akan menjadi sebuah checkbox dan pada bagian kanan akan muncul sebuah icon perpindahan.

6.1.3.1 Merubah Posisi Kata

User dapat mengubah posisi kata yang terdapat pada halaman bookmark dengan menekan icon perpindahan pada suatu cell dan memindahkannya ke posisi yang diinginkan. Untuk mendeteksi perpindahan ini maka digunakan prosedur moveRowAtIndexPath, segmen program 6.4 menunjukkan detail dari prosedur tersebut.

Segmen Program 6.4 Prosedur moveRowAtIndexPath

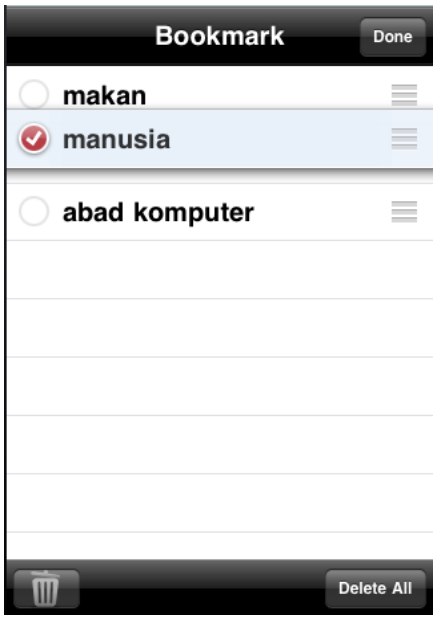
```
1:  -(void) tableView:(UITableView *)tableView
    moveRowAtIndexPath:(NSIndexPath *)sourceIndexPath
    toIndexPath:(NSIndexPath *)destinationIndexPath
2:  {
3:     NSMutableDictionary *data =[[NSMutableDictionary
       alloc]init];
4:     NSString *key=
       [[NSString alloc]initWithFormat:@"%i",[listOfWords count]-1-
       sourceIndexPath.row];
5:     NSString *value=
       [[NSString alloc]initWithFormat:@"%i",[listOfWords count]-1-
       destinationIndexPath.row];
6:     [data setObject:[value retain] forKey:[key retain]];
7:     [[NSNotificationCenter defaultCenter]
       postNotificationName:@"moveBookmark" object: [data retain]];
8:     NSMutableArray *temp = [[NSMutableArray alloc]init];
9:     for(int i=0;i<[listOfBookmarkedWords count];i++)
10:    {
11:       [temp addObject:[[listOfBookmarkedWords objectAtIndex:i]
          stringByReplacingOccurrencesOfString:@"_" withString:@"
          "]];
12:    }
13:    temp = [[[temp reverseObjectEnumerator]
       allObjects]mutableCopy];

14:    listOfWords = [temp retain];
15:    [temp release];
16:    [data release];
17:    [key release];
18:    [value release];
19: }
```

Pada baris 3 disiapkan sebuah dictionary yang akan digunakan untuk menampung index kata yang akan dipindahkan. Key dari dictionary akan berisi index yang akan dipindahkan dan value nya akan berisi index tujuan. Proses pembuatan dictionary ini ditunjukkan oleh baris 4-6. Perlu diingat bahwa posisi kata yang ditampilkan kepada user telah dibalik urutannya karena itu untuk mengambil index yang benar dari kata digunakan rumus:

```
[listOfWords count]-1-sourceIndexPath.row
```

Pada baris 7 dictionary tersebut akan dikirimkan kepada prosedur moveBookmark. Pada baris 9 dilakukan iterasi untuk setiap kata pada bookmark dan setiap kata akan mengalami penggantian karakter garis bawah dengan

karakter spasi, proses ini ditunjukkan oleh baris 10. Setelah semua kata mengalami penggantian maka pada baris 13 urutan kata-kata tersebut akan diubah. Pada baris 14 kata-kata tersebut akan disimpan pada array listOfWords agar kata-kata tersebut disusun kembali untuk ditampilkan kepada user. Prosedur moveBookmark akan dibahas lebih lanjut pada subbab 6.1.4. Dengan adanya prosedur moveRowAtIndexPath ini maka user telah dapat melakukan perpindahan kata dengan melakukan drag and drop pada posisi kata yang diinginkan. Gambar 6.7 menunjukkan contoh perpindahan kata yang dilakukan oleh user.



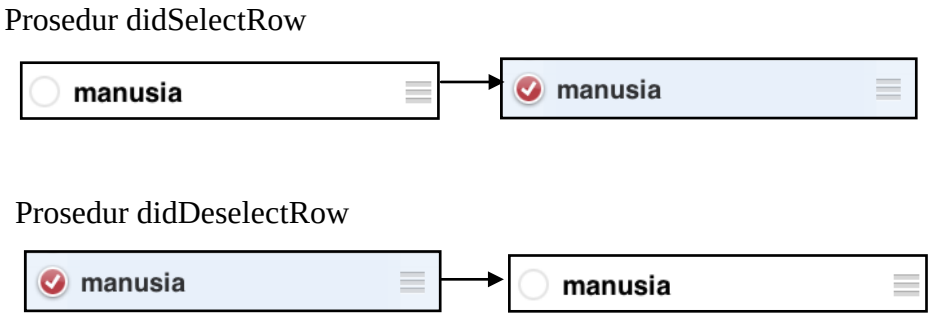
Gambar 6.7
Perpindahan Kata

Pada gambar 6.7 ditunjukkan pemindahan kata manusia yang berasal dari posisi pertama menuju posisi tengah diantara kata makan dan kata abad komputer. Perpindahan ini akan tetap tersimpan pada file bookmark.plist sehingga ketika user berpindah halaman dan kembali ke halaman bookmark posisi kata telah sesuai dengan posisi yang telah ditetapkan oleh user. Walaupun pada gambar 6.7 ditunjukkan kondisi checkbox yang aktif tetapi bukan berarti perpindahan posisi kata akan mengakibatkan checkbox menjadi aktif. Perpindahan posisi kata dapat dilakukan tanpa mengaktifkan checkbox.

6.1.3.2 Menghapus Kata

Untuk melakukan penghapusan pada satu atau lebih kata maka terlebih dahulu user harus mencentang kata yang ingin dihapus. Ketika semua kata yang ingin dihapus sudah dicentang maka user dapat menekan tombol delete yang berada pada posisi kiri bawah atau melakukan swipe ke kanan.

Untuk melakukan proses penghapusan tersebut maka disediakan sebuah array bernama toBeDeleted yang menampung kata-kata yang ingin dihapus oleh user. Pencatatan tidak dilakukan terhadap index cell karena index tersebut dapat berubah-ubah apabila user melakukan perpindahan posisi cell. Tetapi walaupun index berubah tetapi kata yang ingin dihapus tidak akan berubah dan tidak mungkin terdapat kata yang sama pada halaman bookmark. Karena itu pecatatan dilakukan terhadap kata didalam cell. Untuk dapat mencatat kata yang akan dihapus maka browser harus mengenali event ketika suatu cell dipilih. Terdapat dua jenis event yang terjadi yaitu select dan deselect. Gambar 6.8 menunjukkan pemanggilan kedua event tersebut serta prosedur yang terkait.



Gambar 6.8
Ilustrasi Pemanggilan Prosedur

Pada gambar 6.8 ditunjukkan bahwa terdapat dua jenis prosedur yang dipanggil ketika sebuah cell dipilih. Apabila cell berasal dari kondisi tidak aktif menjadi menjadi kondisi aktif maka prosedur didSelectRow akan dipanggil dan jika sebaliknya maka prosedur didDeselectRow akan dipanggil. Kedua prosedur tersebut akan dimanfaatkan untuk melakukan penambahan index ke dalam array toBeDeleted. Prosedur didSelectRow yang ditunjukkan oleh segmen program 6.5.

Segmen Program 6.5 Prosedur didSelectRow

```
1:  -(void) tableView:(UITableView *)tableView
    didSelectRowAtIndexPath:(NSIndexPath *)indexPath
2:  {
3:   if (![bookmarkTableView isEditing])
4:   {
5:     //Pemanggilan halaman result
6:   }
7:   else {
8:     UITableViewCell *cell =
           [bookmarkTableView cellForRowAtIndexPath:indexPath];
9:     [toBeDeleted addObject:cell.textLabel.text];}
10: }
```

Pada baris 3 segmen program 6.5 dilakukan pemeriksaan apakah halaman bookmark sedang berada pada mode edit, jika ya maka program akan menjalankan baris 8 dimana cell yang sedang dipilih user akan disimpan. Text dari cell tersebut kemudian disimpan ke dalam array yang menampung kata yang akan dihapus, proses ini dilakukan pada baris 10. Apabila halaman bookmark tidak sedang berada pada mode edit maka kata yang dipilih akan diproses untuk dikirim pada halaman result.

Segmen Program 6.6 Prosedur didDeselectRow

```
1:  -(void) tableView:(UITableView *)tableView
    didDeselectRowAtIndexPath:(NSIndexPath *)indexPath
2:  {
3:    UITableViewCell *cell =
          [bookmarkTableView cellForRowAtIndexPath:indexPath];
4:    [cell setAccessoryType:UITableViewCellAccessoryNone];

5:    NSMutableArray *cancel = [[NSMutableArray alloc]init];
6:    for(NSString *str in toBeDeleted)
7:    {
8:          if ([str isEqualToString:cell.textLabel.text])
9:                [cancel addObject:str];
10:   }

11:   [toBeDeleted removeObjectsInArray:cancel];
12:   [cancel release];
13: }
```

Ketika sebuah cell dinonaktifkan oleh user maka tanda centang pada cell tersebut harus dihilangkan, oleh karena itu pada baris 3 dan 4 segmen program 6.6 aksesori dari cell yang sedang dipilih user akan dihilangkan. Langkah selanjutnya adalah menghilangkan kata tersebut dari array toBeDeleted. Pada baris 5

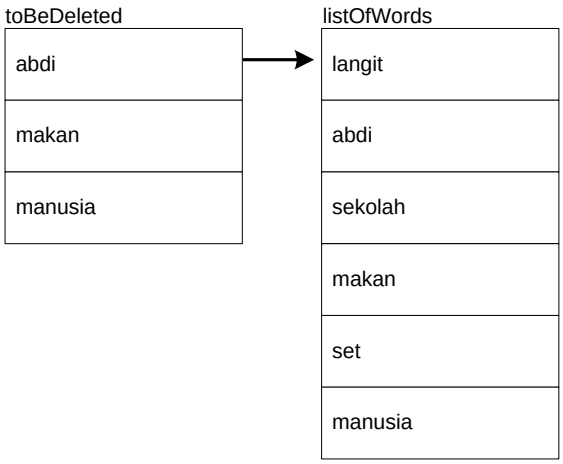
disiapkan sebuah array yang akan menampung kata-kata yang akan dibatalkan oleh user. Pada baris 6 dilakukan iterasi terhadap semua kata yang berada pada array toBeDeleted. Setiap kata tersebut akan mengalami proses pemeriksaan apakah kata tersebut sama dengan kata yang dibatalkan oleh user, jika ya maka kata tersebut akan disimpan ke dalam array cancel. Proses pemeriksaan dan penambahan ke dalam array cancel ditunjukkan oleh baris 8 dan 9. Setelah semua kata yang dibatalkan didapatkan maka langkah selanjutnya adalah menghapus kata-kata tersebut dari array toBeDeleted. Pada baris 11 ditunjukkan penghapusan kata dari array toBeDeleted.

Segmen Program 6.7 Prosedur deleteMode

```
1:  -(IBAction) deleteMode{
2:     NSMutableArray *erase = [[NSMutableArray alloc]init];
3:     for(NSString *str in toBeDeleted)
4:     {
5:       NSIndexPath *path = [NSIndexPath
         indexPathForRow:[listOfWords indexOfObject:str]
         inSection:0];

6:       UITableViewCell *cell = [bookmarkTableView
         cellForRowAtIndexPath:path];

7:       [erase addObject:path];
8:       cell.accessoryType = UITableViewCellAccessoryNone;

9:     }
10:    [listOfWords removeObjectsInArray:toBeDeleted];
11:    [toBeDeleted removeAllObjects];

12:    NSMutableArray *temp = [[NSMutableArray alloc]init];
13:    for(int i=0;i<[listOfWords count];i++)
14:    {
15:      [temp addObject:[[listOfWords objectAtIndex:i]
         stringByReplacingOccurrencesOfString:@"_" withString:@"
         "]];
16:    }
17:    temp =
         [[[temp reverseObjectEnumerator] allObjects]mutableCopy];
18:    listOfWords = [temp retain];
19:    [temp release];
20:    [bookmarkTableView deleteRowsAtIndexPaths:
           erase withRowAnimation:UITableViewRowAnimationRight];
21:    [[NSNotificationCenter defaultCenter]
           postNotificationName:@"updateBookmark" object:
           listOfWords];
22:    [erase release];
23:    [self refresh];
24: }
```

Ketika user menekan tombol delete maka penghapusan kata yang dipilih oleh user akan dilakukan. Proses penghapusan tersebut ditangani oleh prosedur deleteMode yang dipanggil ketika tombol delete ditekan. Segmen program 6.7 menunjukkan detail dari prosedur tersebut.

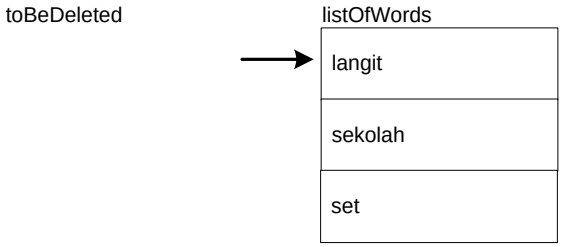
Pada baris 2 segmen program 6.7 ditunjukkan bahwa disiapkan sebuah array bernama erase. Array ini akan menampung cell dari table view yang akan dihapus. Pada baris 3 dilakukan iterasi pada semua kata pada array toBeDeleted. Setiap kata akan digunakan untuk mendapatkan index cell pada table view. Pada baris 5 ditunjukkan pembuatan indexPath menggunakan index dari kata dan pada baris 6 cell pada indexpath tersebut akan didapatkan. Cell tersebut kemudian akan disimpan ke dalam array erase dan aksesori cell akan dihilangkan, proses tersebut ditunjukkan oleh baris 7 dan 8. Setelah semua cell telah didapatkan maka kata-kata tersebut dapat dihapus dari daftar kata bookmark. Proses penghapusan ditunjukkan pada baris 10.Untuk menghapus sebuah cell dari table view, maka ukuran array yang digunakan untuk membentuk cell-cell pada table view harus berjumlah sama dengan jumlah cell baru pada table view.



Gambar 6.9
Kondisi Awal Array

Pada gambar 6.9 ditunjukkan ilustrasi kondisi awal dari aray tobeDeleted yang berisi kata-kata yang akan dihapus dari array listOfWords. Terdapat tiga buah kata yang akan dihapus yaitu abdi, makan, manusia. Ketiga kata tersebut

berada pada posisi yang berbeda-beda pada array listOfWords. Kata abdi berada pada posisi awal, kata makan pada posisi tengah dan kata manusia pada posisi akhir. Proses penghapusan akan menghasilkan kondisi seperti pada gambar 6.10.



Gambar 6.10
Hasil Penghapusan

Walaupun posisi kata yang akan dihapus berada di tengah array tetapi hal tersebut tidak akan mempengaruhi urutan kata yang berada pada array. Setelah semua kata yang akan dihapus sudah dihilangkan dari array listOfWords maka kata-kata yang tercatat pada array toBeDeleted juga harus dihapus karena itu pada gambar 6.10 ditunjukkan array toBeDeleted yang kosong.

Setelah proses penghapusan pada array selesai dilakukan maka pada baris 13-19 array tersebut diproses dan disimpan ke dalam array listOfWords untuk ditampilkan kepada user. Tetapi sebelum ditampillam, cell-cell yang mengandung kata yang dihapus oleh user perlu dihilangkan dari table view terlebih dahulu. Pada baris 20 dilakukan penghapusan pada table view dengan menggunakan prosedur deleteRowsAtIndexPaths. Parameter dari prosedur tersebut adalah sebuah array yang mengandung index path dari cell yang akan dihapus. Pada baris 21 dilakukan penyimpanan perubahan pada file bookmark.plist.

Prosedur untuk menangani penghapusan semua kata pada halaman bookmark memiliki kemiripan dengan prosedur deleteMode perbedaannya terdapat pada iterasi penghapusan kata pada array. Pada proses penghapusan semua kata yang dihapus bukan hanya kata-kata yang ditandai oleh user tetapi semua yang terdapat pada halaman bookmark. Proses penghapusan semua kata tersebut hanya mengubah tahap pertama dari prosedur deleteMode. Sedangkan proses penghapusan cell dari table view yang dilakukan sama. Penghapusan kata

dari array listOfWords untuk semua kata pada bookmark ditunjukkan oleh segmen program 6.8.

Segmen Program 6.8 Penghapusan Semua Kata

```
1:  NSMutableArray *erase = [[NSMutableArray alloc]init];
2:  for(NSString *str in listOfWords)
3:  {
4:     NSIndexPath *path = [NSIndexPath
       indexPathForRow:[listOfWords indexOfObject:str]
       inSection:0];
5:     UITableViewCell *cell = [bookmarkTableView
       cellForRowAtIndexPath:path];
6:     [erase addObject:path];
7:     cell.accessoryType = UITableViewCellAccessoryNone;
8:  }
9:  [listOfWords removeAllObjects];
```

Seperti yang ditunjukkan oleh segmen program 6.8 bahwa iterasi dilakukan sejumlah ukuran dari array listOfWords dan penghapusan kata dilakukan pada setiap iterasi, bukan hanya pada index yang tercatat pada array toBeDeleted.

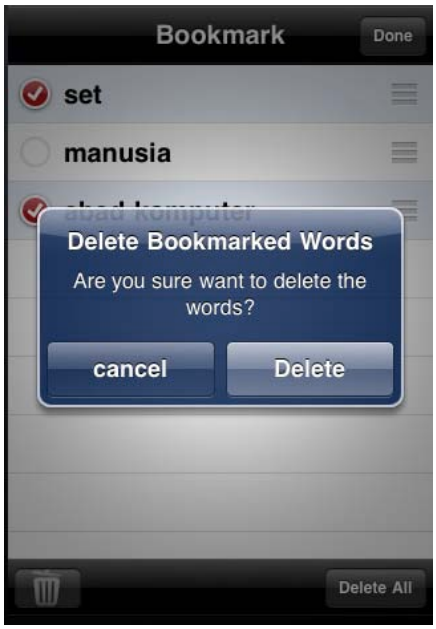
Selain penghapusan dengan menekan tombol delete dan delete all, user juga dapat melakukan penghapusan dengan melakukan swipe ke kanan. Table view akan mengenali swipe tersebut dan menjalankan prosedur delete. Untuk mengenali swipe tersebut maka perlu diberikan sebuah gesture recognizer. Segmen program 6.9 menunjukkan pengenalan gesture tersebut.

Segmen Program 6.9 Pengenalan Gesture

```
1: UISwipeGestureRecognizer *gesture =
   [[[UISwipeGestureRecognizer alloc]initWithTarget:self
   action:@selector(deleteGesture)]autorelease];
2: gesture.direction = UISwipeGestureRecognizerDirectionRight;
3: [bookmarkTableView addGestureRecognizer:gesture];
```

Pada baris 1 segmen program 6.9 dibuat sebuah gesture recognizer. Variabel ini akan menangani swipe yang dilakukan oleh user, ketika swipe dilakukan maka prosedur deleteGesture dilakukan. Prosedur tersebut bertugas untuk menampilkan sebuah alert kepada user untuk melakukan konfirmasi. Apabila user menjawab ya maka prosedur deleteMode akan dipanggil. Pada tahap

ini masih belum ditentukan arah swipe yang ditangani karena itu pada baris 2 dispesifikan bahwa swipe yang ditangani adalah swipe ke arah kanan. Setelah recognizer siap maka langkah terakhir adalah meletakkannya pada table view. Sebuah gesture recognizer tidak hanya dapat diletakkan pada suatu view, dapat juga diletakkan pada table view.



Gambar 6.11
Penghapusan Menggunakan Gesture

Gambar 6.11 menunjukkan tampilan konfirmasi yang muncul ketika penghapusan kata dengan menggunakan gesture dilakukan. Untuk penghapusan dengan menggunakan gesture perlu diberikan sebuah konfirmasi kepada user. Hal tersebut untuk mengatasi kondisi dimana apabila user secara tidak sengaja melakukan gesture penghapusan. Apabila konfirmasi dimunculkan maka penghapusan secara tidak sengaja dapat dicegah.Setelah proses pemindahan posisi kata atau penghapusan kata selesai dilakukan oleh user maka browser akan menyimpan daftar kata yang baru kedalam file bookmark.plist. Proses penyimpanan daftar kata yang baru tersebut dilaksanakan oleh dua buah prosedur. Detail dari proses penyimpanan daftar kata tersebut akan dibahas pada subbab 6.1.4.

6.1.4 Menyimpan Daftar Kata yang Baru

Pada proses perubahan posisi kata dan penghapusan posisi kata terdapat dua buah prosedur yang digunakan untuk menyimpan perubahan yang terjadi ke dalam file bookmark.plist. Ketika user melakukan perubahan posisi kata maka digunakan prosedur moveBookmark sedangkan ketika user melakukan penghapusan kata maka digunakan prosedur updateBookmark. Kedua prosedur tersebut melakukan proses yang berbeda tetapi keduanya menyimpan perubahan yang terjadi ke dalam bookmark.plist.

Prosedur moveBookmark merupakan sebuah prosedur yang digunakan untuk menukar posisi kata dan menyimpan perubahan tersebut pada file bookmark.plist. Segmen program 6.10 menunjukkan detail prosedur tersebut.

Segmen Program 6.10 Prosedur moveBookmark

```
1:  -(void) moveBookmark : (id) data
2:  {
3:   int original,target;
4:   for(NSString *old in [[data object] allKeys])
5:   {
6:    original = [old intValue];
7:    target = [[[data object] valueForKey:old]intValue];
8:   }
9:   NSString *moved = [[listOfBookmarkedWords
     objectAtIndex:original]retain];

10:  [listOfBookmarkedWords removeObjectAtIndex:original];
11:  [listOfBookmarkedWords insertObject:moved atIndex:target];
12:  [listOfBookmarkedWords writeToFile:[[self documentPath]
     stringByAppendingPathComponent:@"/bookmark.plist"]
     atomically:YES];

13:  bookmark.listOfBookmarkedWords = listOfBookmarkedWords;
14: }
```

Pada baris 4 segmen program 6.10 dilakukan iterasi terhadap dictionary yang menampung index kata yang akan dipindah. Karena dipastikan bahwa dictionary hanya berisi sebuah key dan sebuah value maka pada baris 6 key dari dictionary akan diambil sebagai index awal dan pada paris 7 value akan diambil sebagai index yang baru. Setelah index didapatkan maka langkah selanjutnya adalah menyimpan kata pada index awal, hal ini dlakukan pada baris 9. Kemudian kata pada index awal tersebut akan dihapus dan kata yang telah disimpan akan

diletakkan pada index yang baru. Proses perpindahan dan penghapusan ini ditunjukkan oleh baris 10 dan 11. Pada baris 12 perubahan tersebut disimpan ke dalam file bookmark.plist. Langkah terakhir adalah mengirimkan daftar kata yang telah berubah ke halaman bookmark seperti yang dilakukan pada baris 13.

Berbeda dengan prosedur moveBookmark yang berperan melakukan pertukaran index pada bookmark, prosedur updateBookmark hanya bertugas untuk mencatat daftar kata yang didapatkan ke dalam file bookmark.plist. Prosedur ini dipanggil ketika user melakukan penghapusan kata dengan menggunakan tombol delete, tombol delete all, dan gesture dimana penghapusan kata telah dilakukan oleh prosedur deleteMode. Detail dari prosedur ini ditunjukkan oleh segmen program 6.11.

Segmen Program 6.11 Prosedur updateBookmark

```
1:  -(void) updateBookmark : (id) words
2:  {
3:   [listOfBookmarkedWords removeAllObjects];
4:
5:   for(NSString *content in [words object])
6:   {
7:    [listOfBookmarkedWords addObject:[content
      stringByReplacingOccurrencesOfString:@" " withString:@"_"]];
8:   }
9:   [listOfBookmarkedWords writeToFile:[[self documentPath]
     stringByAppendingPathComponent:@"/bookmark.plist"]
     atomically:YES];

10:  bookmark.listOfBookmarkedWords = listOfBookmarkedWords;
11:  activeList = listOfBookmarkedWords;
12:  [bookmark refresh];
13: }
```

Pada baris 5 segmen program 6.11 dilakukan iterasi terhadap semua kata yang telah ditandai user. Setiap kata tersebut akan mengalami pergantian karakter spasi menjadi karakter garis bawah (_), proses ini ditunjukkan oleh baris 7. Setelah semua kata mengalami pergantian karakter maka langkah terakhir adalah menuliskan daftar kata tersebut ke dalam file bookmark.plist yang ditunjukkan oleh baris 9. Pada baris 10 daftar kata yang baru dikirimkan pada halaman bookmark dan pada baris 12 daftar kata yang ditampilkan pada halaman bookmark akan diperbaharui.

6.1.5 Memilih Kata yang Ingin Dilihat

Ketika user memilih kata pada saat mode edit tidak aktif maka browser akan memproses kata yang dipilih dan mengirimkannya kepada halaman result. Pemrosesan kata tersebut ditunjukkan oleh segmen program 6.12.

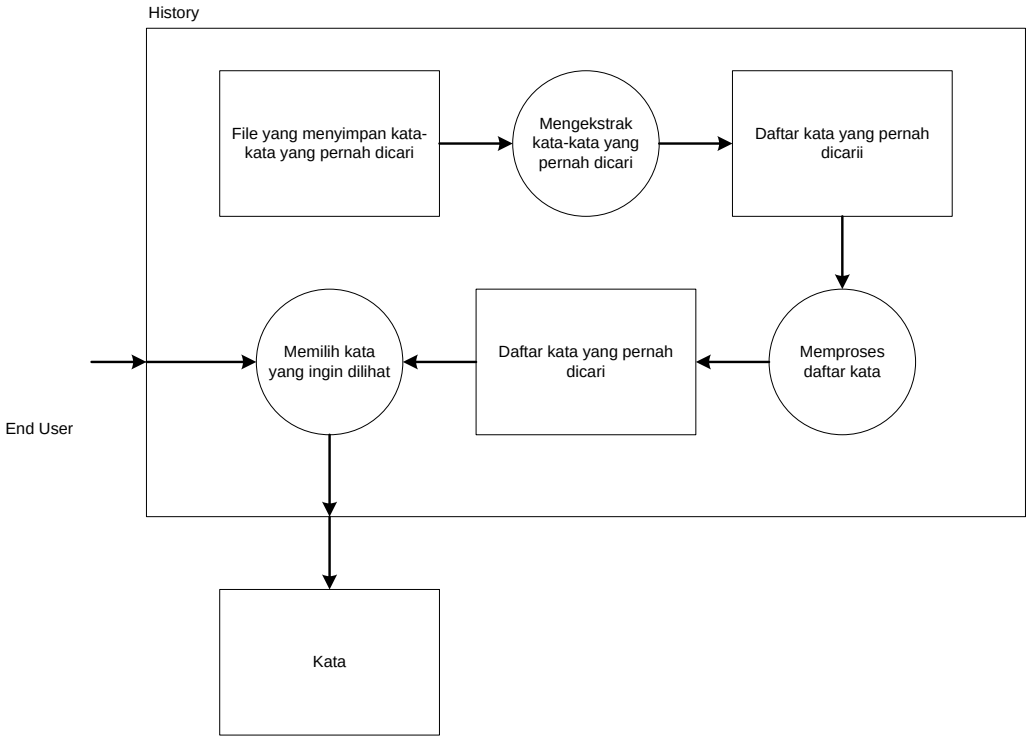
Segmen Program 6.12 Pemrosesan Kata yang Ingin Dilihat

```
1:  Word *w = [WN getWordData:[[listOfWords
    objectAtIndex:indexPath.row]
    stringByReplacingOccurrencesOfString:@" " withString:@"_" ]];

2:  NSString *temp = [[NSString alloc]
    initWithFormat:@"%i",([listOfWords count]-1-indexPath.row) ];

3:  [[NSNotificationCenter defaultCenter]
    postNotificationName:@"setIndexHistory" object: temp];

4:  [[NSNotificationCenter defaultCenter]
    postNotificationName:@"openResult" object: w];
```

Pada baris 1 segmen program 6.12 dilakukan pemrosesan kata dengan memanfaatkan prosedur getWordData milik class WordNet. Hasil dari proses ini dikirimkan kepada halaman result pada baris 4. Selain itu index dari kata yang dipilih juga dikirimkan pada baris 2 dan 3, hal ini dilakukan agar user dapat melakukan navigasi kata (back dan next) pada kata-kata yang sudah dibookmark.

6.2 Halaman History

Halaman History merupakan halaman dimana user dapat melihat daftar kata yang pernah dicari oleh user. Kata-kata yang ditampilkan akan diurutkan sesuai dengan urutan pencarian, kata yang terakhir kali dicari akan ditampilkan pertama dan kata yang pertama kali dicari akan ditampilkan terakhir. Arsitektur sistem dari halaman History memiliki kemiripan dengan halaman bookmark. Proses yang dilakukan lebih sedikit daripada halaman history karena interaksi yang dapat dilakukan oleh user hanyalah memilih kata yang ingin dilihat informasinya. Kata-kata yang terdapat pada halaman history tidak dapat dihapus atau dipindahkan posisinya oleh user. Arsitektur sistem halaman history ditunjukkan oleh gambar 6.12.



Gambar 6.12
Arsitektur Sistem Halaman History

Alur Sistem serta proses yang terjadi pada halaman history memiliki kemiripan dengan halaman bookmark. Input sistem berupa sebuah aksi dari user, hanya saja pada halaman history aksi yang dapat dilakukan oleh user hanyalah memilih kata yang ingin dicari. Output dari sistem berupa sebuah kata yang akan diproses oleh class WordNet dan dikirimkan kepada halaman result untuk ditampilkan infomasi dari kata tersebut.

Proses ekstraksi yang dialami oleh halaman history sama dengan halaman bookmark, yang membedakannya hanyalah file plist yang menjadi sumber data. Pada halaman history yang menjadi sumbernya adalah file history.plist. Sehingga perintah untuk melakukan ekstraksi kata-kata adalah :

```
historyList = [self readFromFile:[[self documentPath]
 stringByAppendingPathComponent:@"/history.plist"]];
```

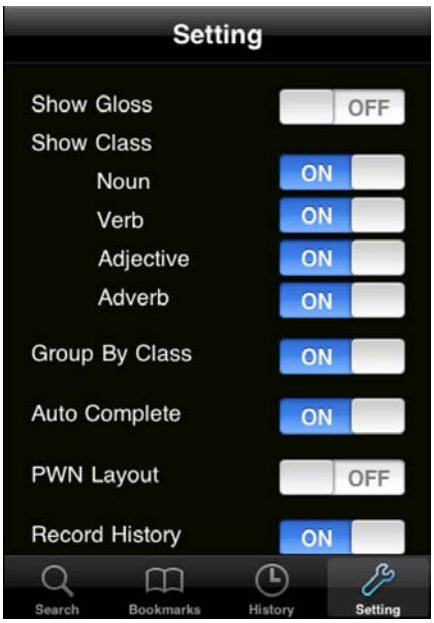
Hasil ekstraksi ini disimpan kedalam sebuah array bernama historyList. Kata-kata yang terdapat pada array historyList akan mengalami proses pembalikan urutan, penggantian karakter garis bawah (_) dengan karakter spasi dan pembentukan cell berisi kata untuk table view, sama dengan proses yang dialami oleh kata-kata yang terdapat pada array listOfBookmarkedWords milik halaman bookmark.

Gambar 6.13
Hasil Pemrosesan Array HistoryList

Gambar 6.13 menunjukkan hasil pemrosesan dari array historyList menjadi cell pada table view milik halaman history. Kata-kata yang terdapat pada array historyList akan mengalami pembalikan urutan dan juga penggantian karakter garis bawah (_) menjadi karakter spasi. Setiap kata tersebut kemudian akan dibentuk menjadi cell-cell pada table view. Pada halaman history tidak terdapat tombol edit yang berfungsi mengubah mode halaman menjadi mode edit. Halaman history hanya memiliki sebuah mode yaitu mode view dan mode tersebut tidak dapat diubah oleh user. Sama dengan halaman bookmark dan halaman search. Setiap cell pada table view memiliki sebuah aksesori disebelah kanan berupa sebuah tanda panah berwarna abu-abu.

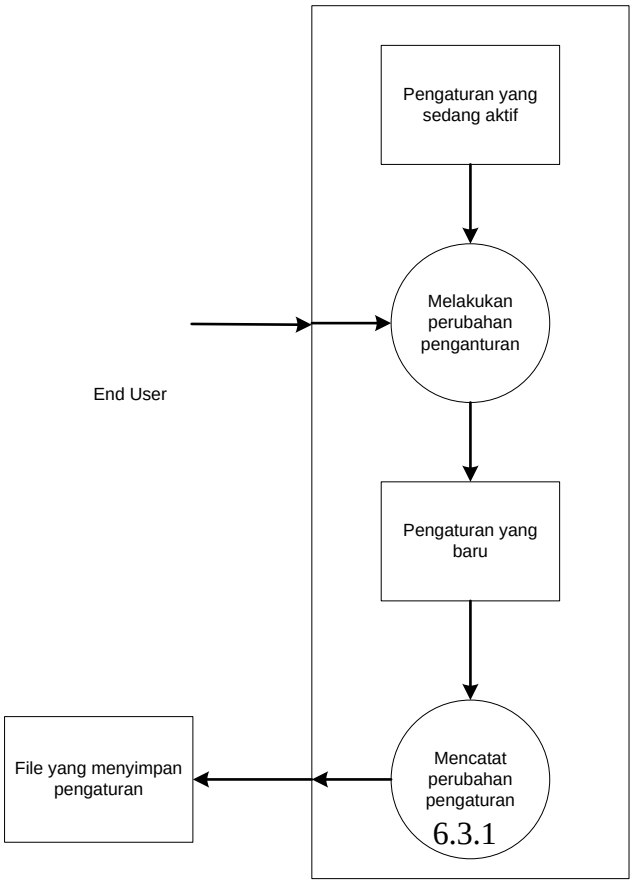
6.3 Halaman Setting

Halaman Setting merupakan halaman yang mengijinkan user untuk melakukan perubahan pengaturan yang dimiliki oleh browser. Setiap fitur yang dimiliki oleh browser diwakilkan oleh sebuah on / off switch, nilai yang didapatkan dari switch tersebut menandakan aktif tidaknya fitur tersebut. Gambar 6.14 menunjukkan tampilan dari halaman setting. Fitur-fitur yang dimiliki oleh aplikasi diwakilkan oleh sebuah on / off switch. Pada gambar 6.14 ditunjukkan bahwa terdapat dua buah fitur yang dinonaktifkan oleh user.



Gambar 6.14
Halaman Setting

Sebelum sebuah fitur dilaksanakan, pemeriksaan terhadap pengaturan ini akan dilakukan. Apabila ternyata user menonaktifkan fitur tersebut maka fitur tidak akan dijalankan. Untuk pengaturan jumlah history yang dapat dicatat oleh user tidak menggunakan on / off switch melainkan menggunakan sebuah slider. Posisi slider tersebut menunjukkan jumlah kata yang akan disimpan pada history browser. Slider tersebut memiliki nilai minimum sepuluh dan nilai maximum seratus.



Gambar 6.15
Arsitektur Sistem Halaman Setting

Gambar 6.15 menunjukkan arsitektur sistem dari halaman setting. Input sistem berupa sebuah aksi dari user dimana user akan melakukan pengubahan pengaturan yang dimiliki oleh browser. Perubahan tersebut kemudian akan dicatat dan diproses oleh browser. Output dari sistem akan berupa sebuah file plist yang mencatat pengaturan baru yang dimiliki oleh user. File tersebut adalah file setting.plist. Nilai yang disimpan pada file setting.plist memiliki tipe boolean dan string dimana tipe string digunakan untuk menyimpan jumlah maksimum kata yang dicatat oleh history. Nilai yang tersimpan pada file setting.plist inilah yang akan digunakan oleh browser untuk menentukan apakah sebuah fitur akan dijalankan oleh browser. Jika nilai yang didapatkan adalah nilai boolean NO maka fitur tersebut tidak akan dijalankan. Khusus untuk pencatatan pada history, browser juga akan melihat jumlah maksimum yang ditetapkan oleh user.

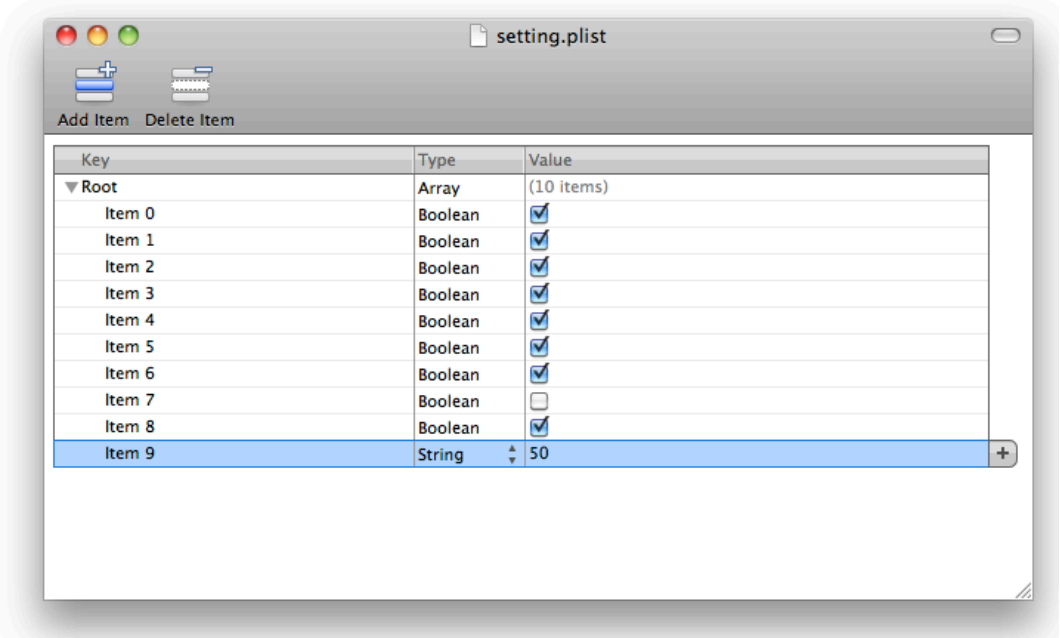
6.3.1 Mencatat Perubahan Pengaturan

Pencatatan perubahan dilakukan oleh prosedur bernama saveSetting. Prosedur ini dipanggil ketika terjadi perubahan state pada on / off switch atau perubahan value pada slider. Prosedur save setting akan melakukan perubahan terhadap file yang menyimpan pengaturan, yaitu file setting.plist. Detail dari prosedur ini ditunjukkan oleh segmen program 6.13.

Segmen Program 6.13 Prosedur saveSetting

```
1:  -(IBAction) saveSetting
2:  {
3:   if ([recordHistory isOn])
4:   {
5:     [label setHidden:NO];
6:     [size  setHidden:NO];
7:   }
8:   else {
9:     [label setHidden:YES];
10:    [size  setHidden:YES];
11:  }
12:  if (![noun isOn] && ![verb isOn] && ![adj isOn] && ![adv
     isOn]) {
13:    [showGloss setOn:NO animated:YES];
14:  }
15:  NSMutableArray *set = [[NSMutableArray alloc]init];
16:  [set addObject: [[NSNumber alloc] initWithBool:[showGloss
     isOn]]];
17:  [set addObject: [[NSNumber alloc] initWithBool:[noun isOn]]];
18:  [set addObject: [[NSNumber alloc] initWithBool:[verb isOn]]];
19:  [set addObject: [[NSNumber alloc] initWithBool:[adj isOn]]];
20:  [set addObject: [[NSNumber alloc] initWithBool:[adv isOn]]];
21:  [set addObject: [[NSNumber alloc] initWithBool:[groupByClass
     isOn]]];
22:  [set addObject: [[NSNumber alloc] initWithBool:[autoComplete
     isOn]]];
23:  [set addObject: [[NSNumber alloc] initWithBool:[pwnLayout
     isOn]]];
24:  [set addObject: [[NSNumber alloc] initWithBool:[recordHistory
     isOn]] ];
25:  NSString *temp = [[NSString
     alloc]initWithFormat:@"%.0f",[size value]*100];
26:  [label setText:temp];
27:  [set addObject:[temp retain]];
28:  [temp release];
29:  [set writeToFile:[[self documentPath]
     stringByAppendingPathComponent:@"/setting.plist"]
     atomically:YES];
30:  [[NSNotificationCenter defaultCenter]
     postNotificationName:@"getSetting" object: nil];
31:  [set release];
32: }
```

Pada baris 3 segmen program 6.13 dilakukan pemeriksaan apakah user menginginkan kata yang dicari untuk dicatat jika ya maka baris 5 dan 6 akan dijalankan dimana slider akan ditampilkan. Tetapi jika user tidak menginginkan pencatatan dilakukan maka akan dijalankan baris 9 dan 10 dimana slider tidak ditampilkan. Pada baris 12 dilakukan pemeriksaan apakah semua class kata tidak akan ditampilkan, jika tidak ada satu pun class kata yang ditampilkan maka secara otomatis gloss tidak akan ditampilkan, proses ini ditunjukkan pada baris 13.Pada baris 15 disiapkan sebuah array yang akan menampung nilai pengaturan dari user. Array ini nantinya akan disimpan pada setting.plist. Baris 16-24 menunjukkan penyimpanan nilai pengaturan ke dalam array. Pada Objective-C nilai bool tidak dapat disimpan ke dalam plist karena itu nilai bool yang didapatkan akan disimbolkan dengan menggunakan angka. Dimana nilai YES akan menjadi 1 dan nilai NO akan menjadi 0. Karena batas pencatatan history menggunakan slider maka pada 25 didapatkan nilai dari slider tanpa digit dibelakang koma. Secara default nilai dari slider bertipe float. Setelah semua nilai didapatkan maka pada baris 29 pengaturan akan disimpan ke dalam file setting.plist.



Gambar 6.16
File Setting.plist

Gambar 6.16 menunjukkan file plist yang dimiliki oleh browser. Setiap nilai bool tersebut melambangkan pengaturan sebuah fitur milik WordNet Browser. Pada baris 30 segmen program 6.13 ditunjukkan pemanggilan dari prosedur getSetting.

6.3.2 Membaca Pengaturan yang Tersimpan

Pembacaan pengaturan milik user ditangani oleh sebuah prosedur bernama getSetting. Prosedur ini akan dipanggil setiap kali terjadi perubahan pengaturan pada halaman setting dan ketika browser dijalankan. Prosedur getSetting akan membaca pengaturan pada file setting.plist dan menyimpannya. Selain itu jika browser pertama kali dijalankan dan belum memiliki file setting.plist maka prosedur ini akan membuatkan sebuah file setting.plist dengan nilai YES untuk semua fitur kecuali pwnLayout. Untuk fitur pwnLayout diberi nilai NO untuk mengatasi kondisi dimana user tidak terbiasa menggunakan WordNet dan lebih terbiasa dengan tampilan kamus konvensional. Detail dari prosedur getSetting ditunjukkan oleh segmen program 6.14.

Segmen Program 6.14 Prosedur getSetting

```
1:  -(void) getSetting
2:  {
3:  if ([[NSFileManager defaultManager] fileExistsAtPath:[[self
    documentPath]
    stringByAppendingPathComponent:@"/setting.plist"]])
4:   {
5:   NSMutableArray *temp = [[NSMutableArray
     alloc]initWithContentsOfFile:[[self documentPath]
     stringByAppendingPathComponent:@"/setting.plist"]];
6:   [setting setShowGloss:[[temp objectAtIndex:0]boolValue]];
7:   [setting setNoun:[[temp objectAtIndex:1]boolValue]];
8:   [setting setVerb:[[temp objectAtIndex:2]boolValue]];
9:   [setting setAdj:[[temp objectAtIndex:3]boolValue]];
10:  [setting setAdv:[[temp objectAtIndex:4]boolValue]];
11:  [setting setGroupByClass:[[temp objectAtIndex:5]boolValue]];
12:  [setting setAutoComplete:[[temp objectAtIndex:6]boolValue]];
13:  [setting setPwnLayout:[[temp objectAtIndex:7]boolValue]];
14:  [setting setRecordHistory:[[temp objectAtIndex:8]boolValue]];
15:  [setting setSize:[temp objectAtIndex:9]];
16:  maxHistory = [[setting size]intValue];
17:   if ([historyList count] > [[setting size]intValue])
18:   {
19:     for (int i = [historyList count] ; i>[[setting size]
        intValue];i--)
```

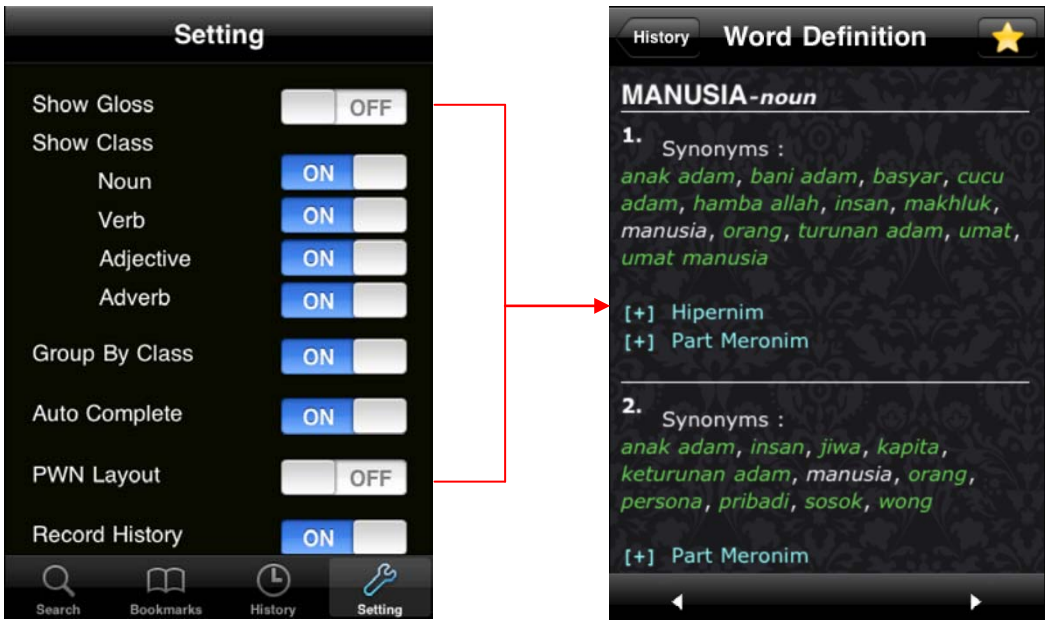
Segmen Program 6.14 (Lanjutan)

```
20:     {
21:       [historyList removeObjectAtIndex:0];
22:     }
23:     [historyList writeToFile:[[self documentPath]
        stringByAppendingPathComponent:@"/history.plist"]
        atomically:YES];
24:   }
25:   indexHistory = [[setting size]intValue]-1;
26:   [temp release];
27:  }
28:  else {
29:  NSMutableArray *temp = [[NSMutableArray alloc]init];
30:  for(int i=0;i<9;i++)
31:  {
32:   if  (i!=7)
33:     [temp addObject:[[NSNumber alloc]initWithBool:YES]];
34:   else
35:     [temp addObject:[[NSNumber alloc]initWithBool:NO]];
36:  }
37:  [temp addObject:@"50"];
38:  [setting setShowGloss:[[temp objectAtIndex:0]boolValue]];
39:  [setting setNoun:[[temp objectAtIndex:1]boolValue]];
40:  [setting setVerb:[[temp objectAtIndex:2]boolValue]];
41:  [setting setAdj:[[temp objectAtIndex:3]boolValue]];
42:  [setting setAdv:[[temp objectAtIndex:4]boolValue]];
43:  [setting setGroupByClass:[[temp objectAtIndex:5]boolValue]];
44:  [setting setAutoComplete:[[temp objectAtIndex:6]boolValue]];
45:  [setting setPwnLayout:[[temp objectAtIndex:7]boolValue]];
46:  [setting setRecordHistory:[[temp objectAtIndex:8]boolValue]];
47:  [setting setSize:[temp objectAtIndex:9]];
48:  maxHistory = [[setting size]intValue];

49:  [temp writeToFile:[[self documentPath]
     stringByAppendingPathComponent:@"/setting.plist"]
     atomically:YES];
50:  [temp release];
51:  }
52: }
```

Terdapat dua jenis proses yang dilakukan oleh prosedur getSetting. Proses pertama adalah pembacaan file setting.plist, proses ini dilakukan ketika pemeriksaan pada baris 3 segmen program 6.14 mengembalikan nilai YES. Pemeriksaan yang dilakukan adalah apakah file setting.plist terdapat pada folder document milik device. Jika file ada maka program akan menjalankan baris 5 dimana dilakukan pembacaan file setting.plist. Pada baris 6-16 dilakukan penyimpanan pengaturan user ke dalam class Setting yang telah disiapkan. Setelah batas pencatatan history didapatkan maka dilakukan pemeriksaan terhadap

array historyList. Apabila jumlah array melebihi batas yang ditentukan oleh user maka array pada index pertama akan selalu dihapus sejumlah selisih batas yang ditentukan dengan ukuran array historyList. Proses pemeriksaan dan penghapusan array ini ditunjukkan oleh baris 17-25.

Proses kedua dilakukan apabila file setting.plist tidak terdapat pada folder document milik device. Bila kondisi ini terjadi berarti browser untuk pertama kalinya dijalankan dan browser harus membuatan sebuah file setting.plist. Pada baris 29-37 disiapkan sebuah array of bool dimana semuanya akan bernilai YES kecuali untuk index 7 dimana index 7 merupakan fitur pwnLayout. Hal ini dilakukan untuk menjaga situasi dimana user tidak terbiasa menggunakan wordnet pada desktop atau web application, karena itu disajikan tampilan yang mengikuti tampilan kamus konvensional. Pada baris 38-48 nilai tersebut akan disimpan ke dalam class setting, untuk batas pencatatan history akan diset dengan nilai 50. Class setting inilah yang akan digunakan untuk melakukan pemeriksaan apakah suatu fitur diaktifkan atau dinonaktifkan oleh user ketika suatu proses dijalankan, misalnya pada fitur auto complete pemeriksaan setting akan dilakukan. Pada baris 49 pengaturan tersebut akan disimpan ke dalam file setting.plist.



Gambar 6.17
Penerapan Pengaturan

Gambar 6.17 menunjukkan penerapan dari pengaturan yang diubah oleh user pada halaman setting. Ditunjukkan bahwa fitur show gloss dan pwn layout dinonaktifkan sehingga pada halaman result gloss dari kata yang dicari tidak akan ditampilkan. Selain itu layout yang digunakan adalah layout yang mengikuti kamus konvensional.

6.3.3 Struktur Penyimpanan Pengaturan.

Untuk menyimpan pengaturan yang dimiliki oleh user maka dibuatlah sebuah struktur yang khusus digunakan untuk menampung pengaturan user yaitu class Setting. Property yang dimiliki oleh class Setting berjumlah sama dengan jumlah pengaturan yang dapat diubah oleh user dan setiap propery tersebut mewakili setiap fitur yang dapat diubah oleh user, selain itu setiap property tersebut memiliki fungsi yang sesuai dengan namanya. Pada tabel 6.2 ditunjukkan struktur dari class setting, struktur tersebut mencakup nama property, tipe property tersebut dan fungsinya.

Tabel 6.2
Struktur Class Setting

Property	Tipe	Fungsi
showGloss	BOOL	Menampilkan gloss
noun	BOOL	Menampilkan informasi class noun
verb	BOOL	Menampilkan informasi class verb
adj	BOOL	Menampilkan informasi class adjective
adv	BOOL	Menampilkan informasi class adverb
groupByClass	BOOL	Apakah setiap class akan memiliki satu layar
autoComplete	BOOL	Menampilkan kata yang mungkin ingin diketikkan
pwnLayout	BOOL	Menggunakan layout PWN
recordHistory	BOOL	Mencatat pencarian kata
size	NSString	Jumlah kata yang akan dicatat

Setiap property yang ada pada tabel 6.2 akan dideklarasikan dengan parameter nonatomic kecuali untuk property size. Property size akan memiliki parameter nonatomic dan retain. Setiap property tersebut akan diakses oleh fitur yang berbeda-beda sesuai dengan fungsi dari property tersebut.

BAB VII
UJI COBA APLIKASI

Pada bab ini akan dijelaskan mengenai uji coba yang dilakukan pada Tugas Akhir. Uji coba akan dilakukan dengan menggunakan beberapa test case. Test case merupakan kumpulan kondisi atau variabel dimana tester akan menentukan apakah kebutuhan atau use case dari suatu aplikasi telah terpenuhi. Test case dapat digunakan untuk mengetahui ketidakstabilan sistem, error yang terjadi, ataupun berisi saran untuk pengembangan lebih lanjut. Selain uji coba aplikasi, juga akan dilaksanan analisa dari Lexical Database yang digunakan, analisa akan meliputi jumlah synset dan gloss yang dimiliki oleh setiap class kata.

7.1 Analisa Kelengkapan Data

Pada subbab ini akan dilakukan analisa kelengkapan data pada Lexical Database. Analisa ini akan dilakukan pada setiap pada setiap class kata, dimana jumlah synset yang dimiliki oleh setiap class kata akan dihitung. Setiap synset akan mengalami pemeriksaan apakah gloss dari synset tersebut telah didapatkan. Hasil analisa ini ditunjukkan oleh tabel 7.1

Tabel 7.1
Analisa Lexical Database Files

Class Kata	Jumlah Synset	Gloss Kosong	Persentasi Gloss Kosong
Noun	41217	4142	10,04%
Verb	20779	7710	37,07%
Adjective	7101	1998	28,14%
Adverb	653	213	32,62%

Pada tabel 7.1 ditunjukkan bahwa class kata noun memiliki jumlah synset terbanyak dan persentasi gloss kosong terkecil. Dari 41217 synset hanya 4142 synset yang tidak memiliki gloss. Hal ini dipengaruhi oleh proses yang dilakukan oleh mahasiswa-mahasiswa yang telah melakukan penelitian untuk melengkapi

class kata noun sebagai Tugas Akhirnya. Sebaliknya class kata verb memiliki persentase gloss kosong yang cukup besar yaitu 37,07%.

7.2 Ujicoba Fungsionalitas Sistem

Ujicoba fungsionalitas sistem akan dikelompokkan berdasarkan halaman yang dimiliki oleh aplikasi. Terdapat lima buah halaman yang akan mengalami uji coba fungsionalitas sistem, yaitu halaman search, halaman result, halaman bookmark, halaman history dan halaman setting. Ujicoba akan dilakukan dengan memanfaatkan test case yang telah disiapkan.

7.2.1 Ujicoba Fungsionalitas Search

Pada subbab ini akan dilakukan ujicoba fungsionalitas pada halama search. Ujicoba akan dilakukan terhadap test case yang telah disiapkan. Pencatatan akan meliputi deskripsi aksei, hasil yang diharapkan dan hasil yang didapatkan.

Tabel 7.2
Uji Coba Halaman Search

No	Deskripsi	Hasil Yang Diharapkan	Hasil
1.	Mengetikkan kata.	Daftar Kata yang mungkin ingin diketikkan user akan muncul.	✓
2.	Melakukan scroll pada daftar kata.	Daftar kata selanjutnya akan ditampilkan.	✓
3.	Melakukan penekanan tombol cancel.	Keyboard akan hilang dari layar.	✓
4.	Mengetikkan dua buah kata yang terpisah dengan spasi.	Browser dapat menangkap dua kata tersebut sebagai satu kesatuan kata.	✓
5.	Menekan tombol search.	Kata yang diketikkan dibawa ke halaman result.	✓
6.	Melakukan tap pada kata yang ada di table view.	Kata yang dipilih dibawa ke halaman result.	✓

Tabel 7.2
(Lanjutan)

No	Deskripsi	Hasil Yang Diharapkan	Hasil
7.	Menekan icon bookmark pada tab bar.	Melakukan perpindahan halaman menuju halaman bookmark.	✓
8.	Menekan icon history pada tab bar.	Melakukan perpindahan halaman menuju halaman history.	✓
9.	Menekan icon setting pada tab bar.	Melakukan perpindahan halaman menuju halaman setting.	✓

Ujicoba pada halaman search berjalan baik untuk keseluruhan test case. Hasil yang didapatkan sesuai dengan hasil yang diharapkan ketika test case dibuat, Selain itu tidak terdapat masalah pada pengujian fitur auto complete. Dapat disimpulkan bahwa halaman search dapat membantu pencarian yang dilakukan oleh user.

7.2.2 Ujicoba Fungsionalitas Result

Untuk melakukan ujicoba fungsionalitas halaman result maka akan dilakukan pencarian terhadap 10 jenis kata, yaitu manusia, set, permaduan, uang, sekolah, organisme, wanita, rumah, langit, baik. Kata-kata tersebut mengandung relasi-relasi yang dimiliki oleh Lexical Database dan beberapa kata memiliki lebih dari satu class kata.

Tabel 7.3
Uji Coba Halaman Result

No	Deskripsi	Hasil Yang Diharapkan	Hasil
1.	Peletakkan lemma kata.	Lemma kata akan terletak pada posisi paling atas.	✓
2.	Peletakkan class kata.	Class kata akan berada di sebelah lemma kata.	✓
3.	Penggunaan separator.	Setiap sense kata akan dipisahkan oleh sebuah garis.	✓

Tabel 7.3
(Lanjutan)

No	Deskripsi	Hasil Yang Diharapkan	Hasil
4.	Pemberian warna yang berbeda untuk sinonim dan relasi.	Sinonim kata akan memiliki wana hijau dan relasi kata akan memiliki warna biru.	✓
5.	Melakukan penekanan simbol […].	Sinonim yang disembunyikan akan ditampilkan.	✓
6.	Menampilkan dan menyembunyikan kata yang memiliki relasi.	Kata yang memiliki relasi dapat ditampilkan dan dihilangkan sesuai keinginan user.	✓
7.	Melakukan pencarian kata dengan menekan kata yang berwarna hijau.	informasi mengenai kata yang diinginkan akan ditampilkan tanpa harus melakukan pencarian pada halaman search.	✓
8.	Menekan tombol back page.	User akan kembali ke halaman yang terakhir dilihat.	✓
9.	Memberi bookmark pada kata.	Konfirmasi penambahan bookmark akan ditampilkan.	✓
10.	Konfirmasi insert bookmark.	Kata akan dicatat pada file bookmark.	✓
11.	Menghilangkan bookmark pada kata.	Konfirmasi penghapusan bookmark akan ditampilkan.	✓
12.	Konfirmasi delete bookmark.	Kata akan dihapus dari file bookmark.	✓
13.	Menekan tombol back pada navigasi kata.	Kata dengan index yang lebih kecil akan ditampilkan.	✓
14.	Menekan tombol next pada navigasi kata.	Kata dengan index yang lebih besar akan ditampilkan.	✓
15.	Menekan tombol back pada index kata terkecil.	Tombol back menjadi non aktif dan tidak ada kata yang ditampilkan.	✓

Tabel 7.3
(Lanjutan)

No	Deskripsi	Hasil Yang Diharapkan	Hasil
16.	Menekan tombol next ketika kata merupakan index terbesar.	Tombol next menjadi non aktif dan tidak ada kata yang ditampilkan.	✓
17.	Melakukan swipe horizontal untuk melihat informasi class kata yang lain.	Informasi kata yang terdapat pada class kata lain akan ditampilkan.	✓
18.	Melakukan swipe vertical untuk melihat informasi berikutnya.	Informasi yang belum terlihat akan ditampilkan.	✓
19.	Melakukan pencarian pada kata yang tidak terdapat pada Lexical Database.	Browser akan menampilkan kata-kata yang memiliki kemiripan penulisan.	✓

Ujicoba pada halaman result berjalan dengan baik untuk keseluruhan test case, hasil yang didapatkan sesuai dengan hasil yang diharapkan. Untuk fitur suggestion waktu yang diperlukan untuk menampilkan kata yang memiliki kemiripan penulisan bervariasi tergantung pada device yang digunakan serta panjang kata yang diinputkan. Semakin panjang kata yang diinputkan maka semakin lama waktu yang diperlukan untuk mendapatkan kata yang memiliki kemiripan penulisan, selain itu kemampuan prosessor dari device juga memberikan pengaruh yang besar. Ketika ujicoba dilakukan pada iTouch dan iPad, iPad yang memiliki kemampan prosessor yang lebih baik dapat menampilkan hasil dengan waktu yang lebih singkat dibandingkan dengan iTouch. Disimpulkan bahwa semua fitur pada halaman result dapat berjalan dengan baik tetapi kecepatan fitur suggestion bergantung pada kemampuan device.

7.2.3 Ujicoba Fungsionalitas Bookmark

Pada subbab ini akan dilakukan uji coba terhadap halaman bookmark. Ujicoba akan meliputi perubahan posisi kata yang dibookmark, penghapusan kata serta melihat informasi dari kata.

Tabel 7.4
Uji Coba Halaman Bookmark

No	Deskripsi	Hasil Yang Diharapkan	Hasil
1.	Memilih kata pada tabel view.	Informasi dari kata yang dipilih akan ditampilkan melalui halaman result.	✓
2.	Menekan tombol edit.	Mode edit akan dijalankan, checkbox dan icon perpindahan akan muncul dan tab bar akan berubah.	✓
3.	Melakukan drag and drop untuk merubah posisi suatu kata.	Kata akan mengalami perubahan posisi dan perubahan akan dicatat.	✓
4.	Memberikan tanda centang pada satu kata atau lebih.	Kata yang dicentang akan dicatat oleh browser.	✓
5.	Menekan tombol delete.	Kata yang dicentang akan dihapus .	✓
6.	Menekan tombol delete all.	Semua kata yang dibookmark akan dihapus.	✓
7.	Menghapus dengan gesture.	Kata yang dicentang akan dihapus.	✓
8.	Menekan tombol done.	Mode edit akan berakhir, aksesori cell akan hilang dan tab bar kembali seperti semula.	✓
9.	Melakukan scroll pada table view.	Kata-kata yang belum muncul akan ditampilkan.	✓

Tabel 7.4
(Lanjutan)

No	Deskripsi	Hasil Yang Diharapkan	Hasil
10.	Menekan icon search pada tab bar.	Melakukan perpindahan halaman menuju search bookmark.	✓
11.	Menekan icon history pada tab bar.	Melakukan perpindahan halaman menuju halaman history.	✓
12.	Menekan icon setting pada tab bar.	Melakukan perpindahan halaman menuju halaman setting.	✓

Pengujian fitur perpindahan posisi kata dan penghapusan kata dilakukan berkali-kali dan dengan berbagai kemungkinan yang ada. Misalnya penandaan kata dan memindahkan kata tersebut sebelum menghapusnya, memindahkan posisi kata dan mengembalikannya lalu menghapusnya. Tidak terdapat masalah pada fitur tersebut, termasuk juga pada penghapusan dengan menggunakan gesture dan tombol delete all. Disimpulkan bahwa fitur-fitur pada halaman bookmark berjalan dengan lancar baik pada mode edit maupun pada mode view.

7.2.4 Ujicoba Halaman History

Pada subbab ini akan dilakukan uji coba terhadap halaman history. Ujicoba akan meliputi perpindahan halaman serta melihat informasi kata dan penyajian daftar kata.

Tabel 7.5
Uji Coba Halaman History

No	Deskripsi	Hasil Yang Diharapkan	Hasil
1.	Menampilkan daftar kata sesuai dengan urutan pencarian.	Kata yang terakhir kali dicari akan berada pada posisi pertama halaman history.	✓
2.	Melakukan tap pada kata.	Informasi pada kata akan ditampilkan melalui halaman result.	✓

Tabel 7.5
(Lanjutan)

No	Deskripsi	Hasil Yang Diharapkan	Hasil
3.	Melakukan scroll pada table view.	Kata-kata yang tidak cukup untuk ditampilkan pada sebuah layar akan ditampilkan.	✓
4.	Menekan icon search pada tab bar.	Melakukan perpindahan halaman menuju search bookmark.	✓
5.	Menekan icon history pada tab bar.	Melakukan perpindahan halaman menuju halaman history.	✓
6.	Menekan icon setting pada tab bar.	Melakukan perpindahan halaman menuju halaman setting.	✓

Ujicoba pada halaman history memiliki hasil yang sesuai dengan hasil yang diharapkan. Penyajian kata sesuai dengan urutan pencarian, kata yang terakhir dicari berada pada posisi pertama halaman history. User dapat melihat semua kata yang pernah dicari user selama kondisi fitur record history diaktifkan ketika pencarian dilakukan. Tidak ditemukan keanehan ataupun error pada saat user memilih kata dari daftar yang ada.

7.2.5 Ujicoba Halaman Setting

Pada subbab ini akan dilakukan uji coba terhadap halaman setting. Ujicoba akan meliputi pengubahan fitur yang dimiliki oleh browser serta efek dari perubahan pengaturan tersebut pada halaman search dan result.

Tabel 7.6
Uji Coba Halaman Setting

No	Deskripsi	Hasil Yang Diharapkan	Hasil
1.	Show Gloss on /off.	User dapat memilih apakah gloss kata akan ditampilkan.	✓
2.	Show Noun on /off.	User dapat memilih apakah informasi kata pada class noun akan ditampilkan.	✓

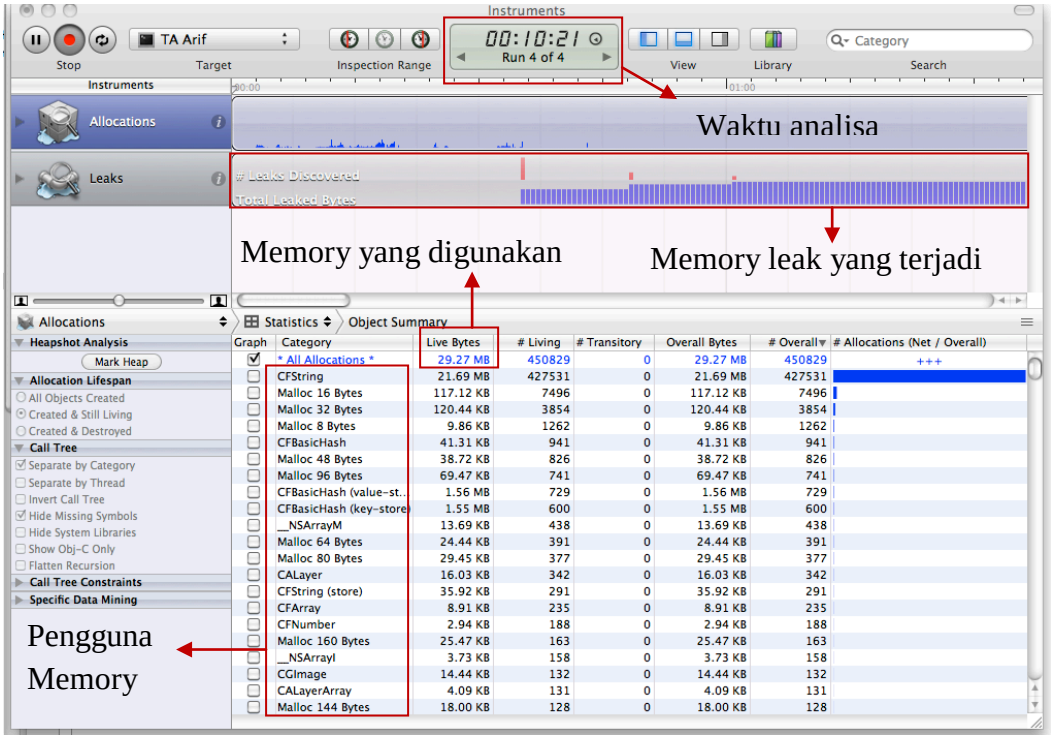
Tabel 7.6
(Lanjutan)

No	Deskripsi	Hasil Yang Diharapkan	Hasil
3.	Show Verb on /off.	User dapat memilih apakah informasi kata pada class verb akan ditampilkan.	✓
4.	Show Adjective on /off.	User dapat memilih apakah informasi kata pada class adjective akan ditampilkan.	✓
5.	Show Adverb on /off.	User dapat memilih apakah informasi kata pada class adverb akan ditampilkan.	✓
6.	Group By Class on /off.	User dapat memilih apakah informasi milik class kata yang berbeda akan berada pada layar yang berbeda.	✓
7.	Auto Complete on /off.	User dapat memilih apakah pada saat pencarian kata browser akan menampilkan daftar kata yang mungkin ingin diketikkan user.	✓
8.	PWN Layout on /off.	User dapat memilih apakah PWN layout akan digunakan.	✓
9.	Record History on /off.	User dapat memilih apakah kata yang dicari akan dicatat ke dalam history.	✓
10.	History Size.	User dapat menentukan jumlah kata yang akan dicatat oleh browser.	✓

Pengujian fitur dilakukan dengan melakukan perubahan pengaturan dan mengujinya terhadap fitur yang bersangkutan. Hasil pengujian menunjukkan bahwa pengatran yang terdapat pada halaman setting berfungsi dengan baik. Hasil yang didapatkan dari pengubahan pengaturan sesuai dengan hasil yang diharapkan. Setiap perubahan pada aktif / nonaktifnya fitur berjalan sesuai dengan pilihan yang dipilih oleh user. Misalnya penonaktifan fitur auto complete akan membuat table view pada halaman search tidak menampilkan kata-kata yang memiliki prefix yang sesuai dengan yang diinputkan oleh user.

7.3 Uji Coba Penggunaan Memory

Uji coba ini dilakukan dengan menggunakan bantuan tool Instruments milik Xcode. Tujuan dari percobaan ini adalah agar jumlah memory yang digunakan oleh aplikasi WordNet Browser diketahui. Ketika Instruments dijalankan maka jumlah memori yang sedang digunakan akan ditampilkan beserta dengan class yang menggunakan memory tersebut. Selain itu apabila terdapat memory leak maka Instruments juga akan menunjukkannya.



Gambar 7.1
Analisa Halaman Search Menggunakan Instruments

Pada gambar 7.1 ditunjukkan penggunaan Instruments dalam menganalisa memory yang digunakan oleh halaman search. Ditunjukkan bahwa halaman search menggunakan memory sebesar 29,27 MB dan pengguna memory terbanyak adalah class String yang menggunakan memory sebesar 21,69 MB. Selain itu juga ditunjukkan bahwa masih terjadi memory leak ketika halaman search memproses kata-kata yang mungkin ingin diketikkan oleh pengguna. Untuk mengatasi hal tersebut maka pemeriksaan secara manual harus dilakukan.

Tabel 7.7
Tabel Penggunaan Memory

Halaman	Fitur yang dilakukan	Memory yang digunakan (MB)
Loading	Melakukan Pre-Processing Lexical Database Files.	23,21
Search	Menampilkan semua kata.	28,96
Search	Fitur Auto-Complete.	29,27
Result	Menampilkan semua informasi kata.	30,08
Result	Menambahkan kata ke dalam bookmark.	30,37
Result	Menghapus kata dari bookmark.	30,37
Result	Melakukan navigasi kata.	30,51
Bookmark	Menampilkan semua kata yang telah ada pada bookmark.	30,39
Bookmark	Memasuki mode edit.	30,39
Bookmark	Memindahkan posisi kata.	30,41
Bookmark	Melakukan penghapusan kata dengan tombol delete, delete all dan gesture.	30,43
History	Menampilkan semua kata yang pernah dicari.	30,43
Setting	Merubah setting.	31,30

Pada tabel 7.7 ditunjukkan daftar memory yang digunakan oleh aplikasi WordNet Browser pada setiap halaman. Dapat dilihat bahwa memory yang digunakan berkisar antara 23 MB sampai 31 MB dimana pada umumnya aplikasi menggunakan 30 MB memory. Dapat disimpulkan bahwa aplikasi WordNet Browser tidak menggunakan banyak memory.

7.4 Uji Coba Kualitas

Percobaan ini dilakukan dengan menggunakan kuesioner. Diambil 20 orang responden untuk menjawab pertanyaan yang disediakan pada kuesioer. Dari 20 orang responden tersebut 12 orang diantaranya menggunakan kamus minimal

2 kali sebulan, dan 8 orang diantaranya menggunakan kamus dengan frekuensi kurang dari 2 kali sebulan. Mayoritas responden adalah laki-laki dengan prosentase perbandingan 90:10 dan rentang usia peserta kuesioner berkisar antara 21-36 tahun. Semua responden merupakan pengguna iPhone / iPad / iTouch dan sebagian diantaranya adalah pengembang aplikasi pada iOS atau MacOS.

Percobaan ini akan memberikan kuesioner untuk mengetahui pendapat pengguna mengenai kualitas aplikasi yang dibuat. Terdapat 8 pertanyaan yang diberikan kepada pengguna untuk mengetahui apakah aplikasi yang dibuat telah mencapai sasaran untuk menjadi sebuah WordNet Browser yang baik dan dapat membantu pengguna dalam mengetahui informasi yang dimiliki oleh suatu kata. Pertanyaan-pertanyaan yang diberikan berkisar pada pendapat pengguna setelah menggunakan WordNet Browser.

Pertanyaan-pertanyaan yang terdapat pada kuesioner dapat dikelompokkan menjadi tiga bagian. Bagian pertama meliputi performa dari aplikasi yaitu kecepatan pemrosesan, kualitas informasi serta interface dan fitur-fitur yang ada. Bagian kedua berupa perbandingan dengan aplikasi lain yang sejenis. Dan bagian yang terakhir berupa pertanyaan apakah responden mau menggunakan aplikasi ini untuk keperluan mencari informasi suatu kata dan apakah responden akan mengunduh aplikasi ini bila aplikasi ini dipublikasikan dan diletakkan pada iTunes. Hasil yang didapatkan dari kuesioner akan ditampilkan dengan menggunakan prosentase jawaban yang didapatkan. Untuk pertanyaan bagian pertama, prosentase dan pertanyaan yang diberikan ditampilkan oleh tabel 7.8 dimana terdapat lima buah pertanyaan dan tiga jenis jawaban yang disediakan.

Tabel 7.8 menunjukkan pertanyaan dan prosentase jawaban yang didapatkan untuk bagian pertama. Untuk setiap pertanyaan terdapat tiga jenis jawaban yang diberikan yaitu baik, cukup, dan kurang. Jawaban yang ada telah dihitung sehingga terbentuk prosentase. Prosentase dihitung dengan rumus (jumlah jawaban / 20) dikalikan dengan 100%. Berdasarkan prosentase jawaban yang ditunjukkan oleh tabel 7.8 dapat disimpulkan bahwa aplikasi WordNet

Browser yang dibuat telah cukup baik walaupun masih terdapat kekurangan pada penyajian informasi dan informasi yang dimiliki.

Tabel 7.8
Pertanyaan dan Prosentase Jawaban Untuk Bagian Pertama

Pertanyaan	Baik	Cukup	Kurang
Bagaimana pendapat anda mengenai performa (kecepatan pencarian, waktu loading, dll) aplikasi ini?	90%	10%	0%
Bagaimana pendapat anda mengenai kelengkapan informasi yang disediakan?	50%	35%	15%
Bagaimana pendapat anda mengenai fitur-fitur yang dimiliki oleh aplikasi WordNet Browser ini?	50%	50%	0%
Bagaimana pendapat anda mengenai cara penyajian informasi aplikasi ini (perbedaan warna, pengaturan posisi informasi, dll)?	60%	35%	5%
Secara keseluruhan bagaimana pendapat anda mengenai aplikasi WordNet Browser ini?	55%	45%	0%

Terdapat beberapa kekurangan pada aplikasi WordNet Browser ini, salah satu kekurangannya adalah pada kualitas data yang diberikan. Sebanyak 15% responden mengatakan bahwa kualitas data masih blm cukup. Kualitas data ini akna terus diperbaiki seiring dengan perkembangan WordNet Bahasa Indonesia. Selain masalah kualitas data, 5% responden mengatakan bahwa tampilan yang

diberikan masih belum cukup baik. Untuk mengatasi hal tersebut akan dilakukan percobaan untuk menemukan tampilan yang lebih baik, misalnya pemberian warna lain pada kata tertentu, penggantian background, struktur penyusunan informasi yang berbeda.

Tabel 7.9
Pertanyaan dan Prosentase Jawaban Untuk Bagian Kedua

Pertanyaan	Lebih Baik	Sama Saja	Lebih Buruk
Bagaimana kualitas aplikasi ini dibanding dengan aplikasi sejenis?	28%	67%	5%

Pada bagian kedua diberikan tiga jenis jawaban yaitu lebih baik, sama saja, dan lebih buruk. Pada saat pengisian kuesioner dilakukan disediakan aplikasi Dictionary.com sebagai aplikasi pembanding. Dari hasil yang ditunjukkan pada tabel 7.9 dapat diketahui bahwa aplikasi WordNet Browser ini telah memiliki kualitas yang sama dengan aplikasi Dictionary.com, 28% dari responden juga menyatakan bahwa WordNet Browser yang dibuat lebih baik daripada aplikasi Dictionary.com. Walaupun demikian 5% dari responden menyatakan bahwa aplikasi ini lebih buruk dari pada aplikasi sejenis.

Untuk mengatasi hal tersebut akan dilakukan peningkatan kualitas aplikasi baik dari segi fitur yang diberikan maupun tampilan yang dimiliki agar dapat menyajikan informasi yang ada dengan lebih baik. Diharapkan pada perkembangannya aplikasi ini dapat memiliki kualitas yang baik dan tidak ada responden yang menyatakan bahwa kualitas aplikasi lebih buruk dibandingkan dengan aplikasi sejenis yang lain.

Pada bagian ketiga disediakan dua jenis jawaban yaitu ya dan tidak. Pertanyaan pada bagian ini difokuskan pada apakah responden mau mengunduh dan menggunakan aplikasi ini. Dari hasil yang ditunjukkan pada tabel 7.10 diketahui bahwa responden mau mengunduh dan menggunakan aplikasi WordNet Browser untuk mencari informasi yang dimiliki oleh suatu kata.

Tabel 7.10
Pertanyaan dan Prosentase Jawaban Untuk Bagian Ketiga

Pertanyaan	Ya	Tidak
Apakah anda tertarik untuk menggunakan aplikasi ini untuk keperluan mencari informasi suatu kata?	75%	25%
Apabila aplikasi ini dipublikasikan, apakah anda tertarik untuk mengunduhnya?	75%	25%

Dari hasil kuesioner yang didapatkan dapat disimpulkan bahwa aplikasi WordNet Browser yang dibuat pada Tugas Akhir ini cukup baik dan telah memenuhi tujuan dibuatnya aplikasi ini. Contoh lembaran kuesioner dan hasil yang didapatkan dapat dilihat pada halaman lampiran.

7.5 Kritik dan Saran

Dari kuesioner yang disebarkan terdapat beberapa saran-saran yang diberikan oleh para responden. Sebagian besar sarang yang diberikan berhubungan dengan fitur yang disediakan pada aplikasi. Berikut ini adalah beberapa saran yang diberikan:

1. Pada halaman search sebaiknya untuk pertama kali tidak ada data yang ditampilkan karena belum tentu sesuai dengan apa yang akan dicari user.
2. Sebaiknya disediakan fitur untuk menampilkan secara berkelompok, misalnya dikelompokkan berdasarkan kelompok sinonim tertentu.
3. Untuk fitur penghapusan kata dengan menggunakan gesture sebaiknya diberikan keterangan. Karena user tidak akan tahu apabila penghapusan bisa dilakukan dengan menggunakan gesture.
4. Warna background yang digunakan sebaiknya tidak terlalu mencolok atau menarik perhatian.

5. Hasil fitur suggestion akan lebih baik apabila diletakkan pada table view di halaman search.
6. Aplikasi ini memiliki fungsionalitas yang lebih baik dari aplikasi sejenis, namun resolusi layar perlu diperhatikan karena ketika dijalankan pada iPad masih terlihat buram.
7. Flow aplikasi kurang intuitif.
8. Terdapat kata-kata yang tidak tercantum pada EYD (Ejaan yang disempurnakan).
9. Transisi antar tab yang diberikan tidak standar.
10. Menu sebaiknya menggunakan bahasa Indonesia.

Saran-saran yang didapatkan akan diterapkan secara bertahap dan diujicobakan akan program dapat semakin berkembang. Diharapkan aplikasi akan menjadi lebih baik.

BAB VIII
PENUTUP

Bagian ini merupakan bagian terakhir yang berisi kesimpulan dan saran bagi pembaca buku ini. Kesimpulan dan saran ini diharapkan akan berguna bagi pembaca agar lebih memahami dan dapat melakukan pengembangan pada penelitian lebih lanjut tentang ekstraksi Lexical Database serta WordNet Browser yang digunakan untuk menampilkan informasi yang didapatkan.

8.1 Kesimpulan

Pada bagian ini disusun kesimpulan dari hasil yang diperoleh selama dilakukan ekstraksi informasi pada Lexical Database dan pembuatan WordNet Browser untuk iPhone dan iPad Berikut ini merupakan kesimpulan yang telah dibuat:

1. WordNet telah memiliki sejarah yang lama dan dalam perkembangannya, tampilan yang digunakan untuk menyajikan informasi telah mengalami banyak perubahan agar menjadi lebih baik. Untuk menerapkan tampilan tersebut ke dalam iPhone yang memiliki desain tampilan yang baru diperlukan berbagai percobaan untuk mendapatkan tampilan yang tepat dan dapat menggabungkan keduanya.
2. Dalam mendesain interface untuk iPhone dan iPad perlu diperhatikan mengenai kebiasaan dari pengguna iPhone dan iPad. Peletakan tombol pada posisi yang tidak biasa akan membingungkan pengguna.
3. Penggunaan bahasa pemrograman C++ akan sangat membantu dalam melakukan pemrosesan data dalam jumlah besar. Pada tugas akhir ini penggunaan C++ sangat mempersingkat waktu yang diperlukan untuk mengolah Lexical Database Files.
4. Memory Management merupakan faktor yang perlu diperhatikan dalam pembuatan aplikasi iPhone dan iPhone. Apabila memory management tidak dilakukan maka aplikasi akan menggunakan banyak memory dari

device sehingga aplikasi hanya dapat berjalan untuk waktu yang singkat sebelum aplikasi diberhentikan oleh device.

5. Desain Interface sangat penting dalam pembuatan sebuah WordNet Browser. Design yang dibuat haruslah menyajikan informasi secara lengkap kepada user tetapi tidak memberikan kesan sesak dan tidak membuat user untuk malas membaca.
6. Memory leak merupakan masalah yang harus diperhatikan. Apabila memory leak terjadi maka aplikasi akan berjalan lebih lambat selain itu ketika memory yang digunakan melebihi batas maka aplikasi akan secara otomatis ditutup oleh device.
7. Instruments merupakan aplikasi yang berguna untuk mengatasi memory leak. Dengan menggunakan instruments maka memory leak dalam aplikasi akan dapat dideteksi. Selain itu instruments juga akan menampilkan jenis variabel yang menyebabkan terjadinya memory leak.
8. WordNet Browser yang dibuat pada tugas akhir ini akan kompatibel dengan Lexical Database bahasa lain. Selama format yang digunakan pada Lexical Database tersebut merupakan format standar.

Diharapkan kesimpulan-kesimpulan yang telah didapatkan ini dapat berguna bagi para pengembang aplikasi iOS dan juga bagi para pengembang aplikasi WordNet browser yang lain.

8.2 Saran

Selain kesimpulan pada subbab 8.1, dari proses pembuatan ekstraktor Lexical Database dan WordNet Browser, terdapat beberapa saran yang mungkin berguna bagi pembaca dalam mengembangkan topik ini lebih lanjut.

1. Pertimbangkan pemanfaatan sqlite untuk menyimpan hasil ekstraksi informasi Lexical Database. Dengan menggunakan sqlite ekstraksi tidak perlu dilakukan berkali-kali sehingga dapat mengurangi waktu loading aplikasi.

2. Manfaatkan feedback yang disediakan oleh iTunes untuk mengembangkan WordNet browser yang lebih baik. Dengan feedback tersebut dapat diketahui kelemahan dari browser yang perlu diperbaiki.

Diharapkan saran yang diberikan dapat berguna bagi pengembang ekstraktor Lexical Database yang lain dan dapat mengingkatkan performa dari ekstraktor Lexical Database yang dibuat pada Tugas Akhir ini.

DAFTAR PUSTAKA

Anonymous. *Cocoa Application Tutorial*, Apple Inc. 2009

Dalrymple, Mark, Scott Knaster. *Learn Objective-C on the Mac.* Apress. 2009.

Dan Pilone, Tracey Pilone, *Head First iPhone Development*. O'Reilly Media. 2009

Kochan, Stephen G. *Programming in Objective-C 2.0 (2nd Edition)*. Addison-Wesley Professional. 2009

Mark, Dave, Jeff La Marche. *Beginning iPhone Development: Exploring The iPhone SDK*. Apress. 2009

Pangestu. S., *Pembuatan Prototype Database Lexical untuk Bahasa Indonesia yang mengacu pada Wordnet*. 2007

Sadun, Erica. *The iPhone Developer's Cookbook : Building Applications with the iPhone SDK*. Addison-Wesley. 2009

Wei-Meng Lee, *iOS 4 Application Development.* Wiley Publishing. 2010

RIWAYAT HIDUP

Nama	**:**	Arif Muliadi Chandra
Alamat Asal	**:**	Jl. Manyar Jaya 9 / 16, Surabaya
Tempat/Tanggal Lahir	**:**	Surabaya, 22 Agustus 1989

Jenjang Pendidikan:

- 1995 – 1998 SDK Theresia 1, Surabaya
- 1998 – 2001 SD Kr Petra 11, Surabaya
- 2001– 2004 SMP Kr Petra 3, Surabaya
- 2004 – 2007 SMA Kr Petra 2, Surabaya
- Sejak 2007 Sekolah Tinggi Teknik Surabaya, Surabaya
  (Program Studi S1 Jurusan Teknik Informatika)

Pengalaman Kerja:

- Januari 2009 – Juli 2009 Asisten Honorer Laboratorium Komputer STTS
- Agustus 2009 – Juli 2011 Asisten Tetap Laboratorium Komputer STTS

LAMPIRAN A
KUESIONER

Kuesioner

Identitas Responden

Nama :_____

Status : (mahasiswa / telah bekerja / dll) *

Jenis Kelamin : (laki-laki / perempuan) *

Usia : _____tahun

Apakah anda sering menggunakan kamus? (>2 kali sebulan)

a. Ya b. Tidak

WordNet Browser

WordNet Browser adalah sebuah aplikasi yang menyediakan definisi, sinonim, dan relasi semantik dan lexical kata. Relasi semantik mencakup hipernim dan hiponim, holonim dan meronim. Untuk relasi lexical mencakup antonim. Apabila pengguna memiliki ketertarikan atau keperluan terhadap suatu kata maka aplikasi WordNet Browser menyediakan fasilitas bookmark. Kata-kata yang telah dibookmark dapat diubah posisinya serta dapat dihapus sesuai dengan keinginan pengguna. Semua kata yang telah dicari pengguna akan dicatat dan dapat dilihat pada halaman history.

Melalui kuesioner ini, saya Arif Muliadi Chandra, mahasiswa Sekolah Tinggi Teknik Surabaya (STTS) jurusan Teknik Informatika meminta bantuan anda untuk menilai aplikasi WordNet Browser ini. Atas partisipasi dan kesediaan anda dalam mengisi kuesioner ini saya ucapkan terima kasih. **) coret yang tidak perlu*

I. Pertanyaan:

Berikan tanda silang (x) pada jawaban yang anda anggap paling sesuai.

1. Bagaimana pendapat anda mengenai performa (kecepatan pencarian, waktu loading,dll) aplikasi ini?
   a. Baik b. Cukup. c.Kurang
2. Bagaimana pendapat anda mengenai kelengkapan informasi yang disediakan?
   a. Baik b. Cukup. c.Kurang
3. Bagaimana pendapat anda mengenai fitur-fitur yang dimiliki oleh aplikasi WordNet Browser ini?
   a. Baik b. Cukup. c.Kurang
4. Bagamana pendapat anda mengenai cara penyajian informasi aplikasi ini (perbedaan warna, pengaturan posisi informasi,dll)?
   a. Baik b. Cukup. c.Kurang
5. Secara keseluruhan bagaimana pendapat anda mengenai aplikasi WordNet Browser ini?
   a. Baik b. Cukup. c.Kurang
6. Bagaimana kualitas aplikasi ini dibanding dengan aplikasi sejenis?
   a. Lebih Baik b. Sama Saja. c.Lebih Buruk
7. Apakah anda tertarik untuk menggunakan aplikasi ini untuk keperluan mencari informasi mengenai suatu kata?
   a. Ya b. Tidak
8. Apabila aplikasi ini akan dipublikasikan, apakah anda tertarik untuk mengunduhnya?
   a. Ya b. Tidak

II. Kritik dan Saran

Kuesioner

Identitas Responden

Nama : Camila

Status : (mahasiswa / telah bekerja / dll) *

Jenis Kelamin : (laki-laki / perempuan) *

Usia : 22 tahun

Apakah anda sering menggunakan kamus? (>2 kali sebulan)

a. Ya b. Tidak

WordNet Browser

WordNet Browser adalah sebuah aplikasi yang menyediakan definisi, sinonim, dan relasi semantik dan lexical kata. Relasi semantik mencakup hipernim dan hiponim, holonim dan meronim. Untuk relasi lexical mencakup antonim. Apabila pengguna memiliki ketertarikan atau keperluan terhadap suatu kata maka aplikasi WordNet Browser menyediakan fasilitas bookmark. Kata-kata yang telah dibookmark dapat diubah posisinya serta dapat dihapus sesuai dengan keinginan pengguna. Semua kata yang telah dicari pengguna akan dicatat dan dapat dilihat pada halaman history.

Melalui kuesioner ini, saya Arif Muliadi Chandra, mahasiswa Sekolah Tinggi Teknik Surabaya (STTS) jurusan Teknik Informatika meminta bantuan anda untuk menilai aplikasi WordNet Browser ini. Atas partisipasi dan kesediaan anda dalam mengisi kuesioner ini saya ucapkan terima kasih. **) coret yang tidak perlu*

I. Pertanyaan:

Berikan tanda silang (x) pada jawaban yang anda anggap paling sesuai.

1. Bagaimana pendapat anda mengenai performa (kecepatan pencarian, waktu loading,dll) aplikasi ini?
   a. Baik b. Cukup. c.Kurang
2. Bagaimana pendapat anda mengenai kelengkapan informasi yang disediakan?
   a. Baik b. Cukup. c.Kurang
3. Bagaimana pendapat anda mengenai fitur-fitur yang dimiliki oleh aplikasi WordNet Browser ini?
   a. Baik b. Cukup. c.Kurang
4. Bagaimana pendapat anda mengenai cara penyajian informasi aplikasi ini (perbedaan warna, pengaturan posisi informasi,dll)?
   a. Baik b. Cukup. c.Kurang
5. Secara keseluruhan bagaimana pendapat anda mengenai aplikasi WordNet Browser ini?
   a. Baik b. Cukup. c.Kurang
6. Bagaimana kualitas aplikasi ini dibanding dengan aplikasi sejenis?
   a. Lebih Baik b. Sama Saja. c.Lebih Buruk
7. Apakah anda tertarik untuk menggunakan aplikasi ini untuk keperluan mencari informasi mengenai suatu kata?
   a. Ya b. Tidak
8. Apabila aplikasi ini akan dipublikasikan, apakah anda tertarik untuk mengunduhnya?
   a. Ya b. Tidak

II. Kritik dan Saran

- Flow aplikasi kurang intuitif
- Ada kata" yg ga jelas / ga ~~ga~~ EYD =P
- Transisi ~~ke~~ antar tab tdk standar (kayanya ga perlu animasi page curl)
- Menu sebaiknya pakai bhs Indonesia juga