**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.



6.1.3

6.1.5

6.1.4

6.1.2

6.1.1

**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.

Kembali pada mode view



Icon Perpindahan

Tab bar baru

Checkbox

**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];

1. NSString \*value=

[[NSString alloc]initWithFormat:@"%i",[listOfWords count]-1- destinationIndexPath.row];

1. [data setObject:[value retain] forKey:[key retain]];
2. [[NSNotificationCenter defaultCenter] postNotificationName:@"moveBookmark" object: [data retain]];
3. NSMutableArray \*temp = [[NSMutableArray alloc]init];
4. for(int i=0;i<[listOfBookmarkedWords count];i++)
5. {
6. [temp addObject:[[listOfBookmarkedWords objectAtIndex:i] stringByReplacingOccurrencesOfString:@"\_" withString:@" "]];
7. }
8. temp = [[[temp reverseObjectEnumerator] allObjects]mutableCopy];
9. listOfWords = [temp retain];
10. [temp release];
11. [data release];
12. [key release];
13. [value release];
14. }

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.

Prosedur didSelectRow

Prosedur didDeselectRow

**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];

1. [toBeDeleted addObject:cell.textLabel.text];}
2. }

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];

1. [cell setAccessoryType:UITableViewCellAccessoryNone];
2. NSMutableArray \*cancel = [[NSMutableArray alloc]init];
3. for(NSString \*str in toBeDeleted)
4. {
5. if ([str isEqualToString:cell.textLabel.text])
6. [cancel addObject:str];
7. }

1. [toBeDeleted removeObjectsInArray:cancel];
2. [cancel release];
3. }

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];

1. listOfWords = [temp retain];
2. [temp release];
3. [bookmarkTableView deleteRowsAtIndexPaths:

erase withRowAnimation:UITableViewRowAnimationRight];

1. [[NSNotificationCenter defaultCenter]

postNotificationName:@"updateBookmark" object: listOfWords];

1. [erase release];
2. [self refresh];
3. }

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.

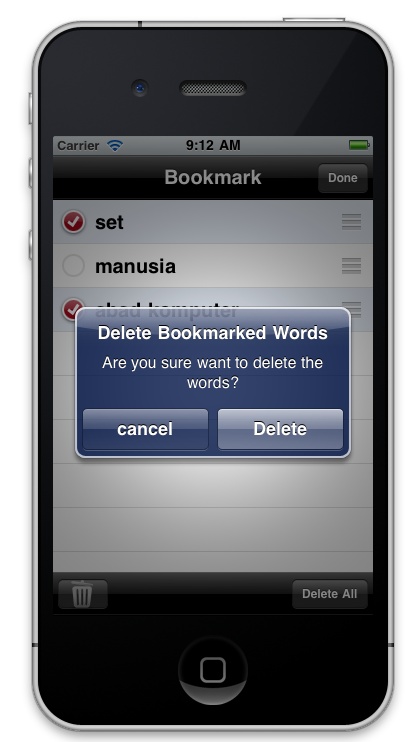
**Segmen Program 6.9 Pengenalan Gesture**

1. UISwipeGestureRecognizer \*gesture =

[[[UISwipeGestureRecognizer alloc]initWithTarget:self action:@selector(deleteGesture)]autorelease];

1. gesture.direction = UISwipeGestureRecognizerDirectionRight;
2. [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. for(NSString \*content in [words object])
5. {
6. [listOfBookmarkedWords addObject:[content stringByReplacingOccurrencesOfString:@" " withString:@"\_"]];
7. }
8. [listOfBookmarkedWords writeToFile:[[self documentPath] stringByAppendingPathComponent:@"/bookmark.plist"] atomically:YES];
9. bookmark.listOfBookmarkedWords = listOfBookmarkedWords;
10. activeList = listOfBookmarkedWords;
11. [bookmark refresh];
12. }

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.



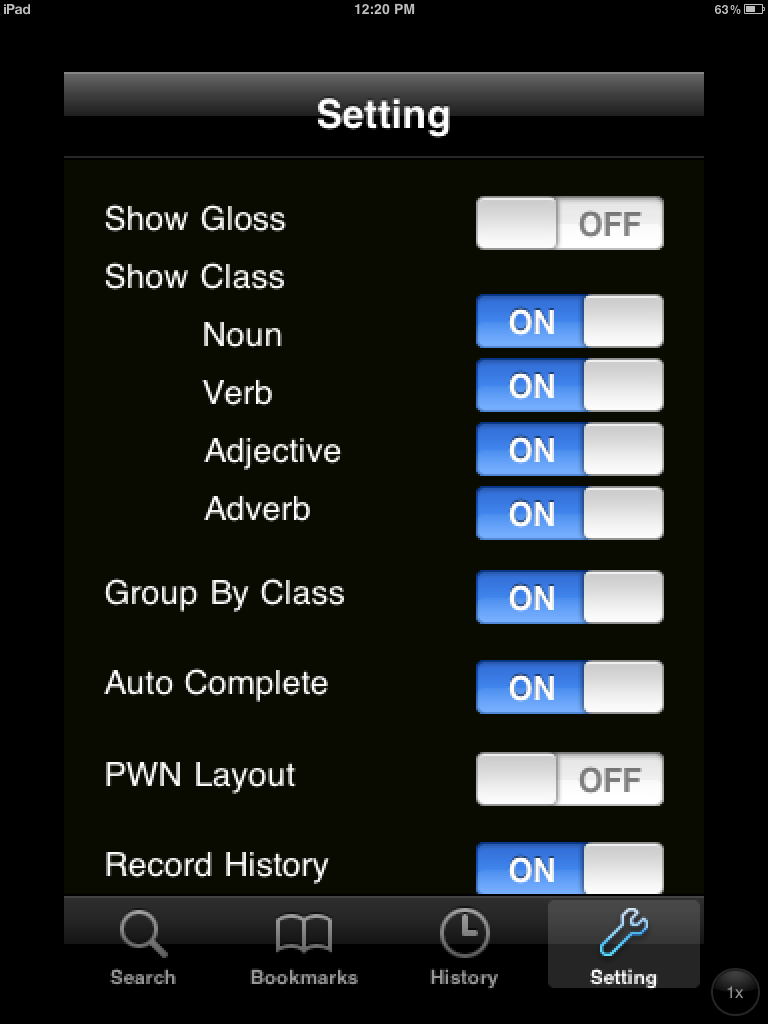
**Gambar 6.13**

**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.



6.3.1

**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.

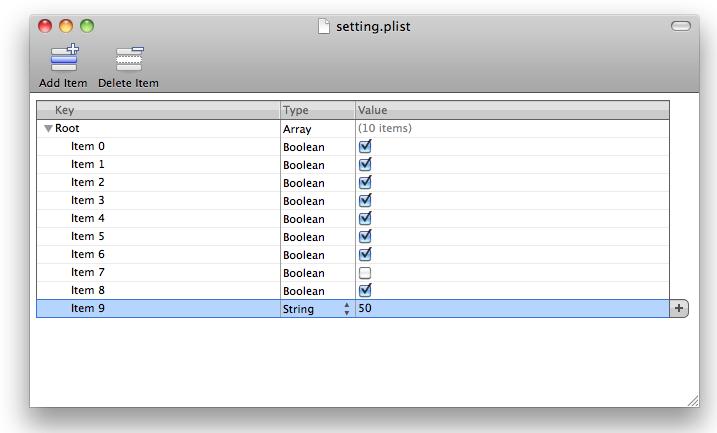
**6.3.1 Mencatat Perubahan Pengaturan**

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**

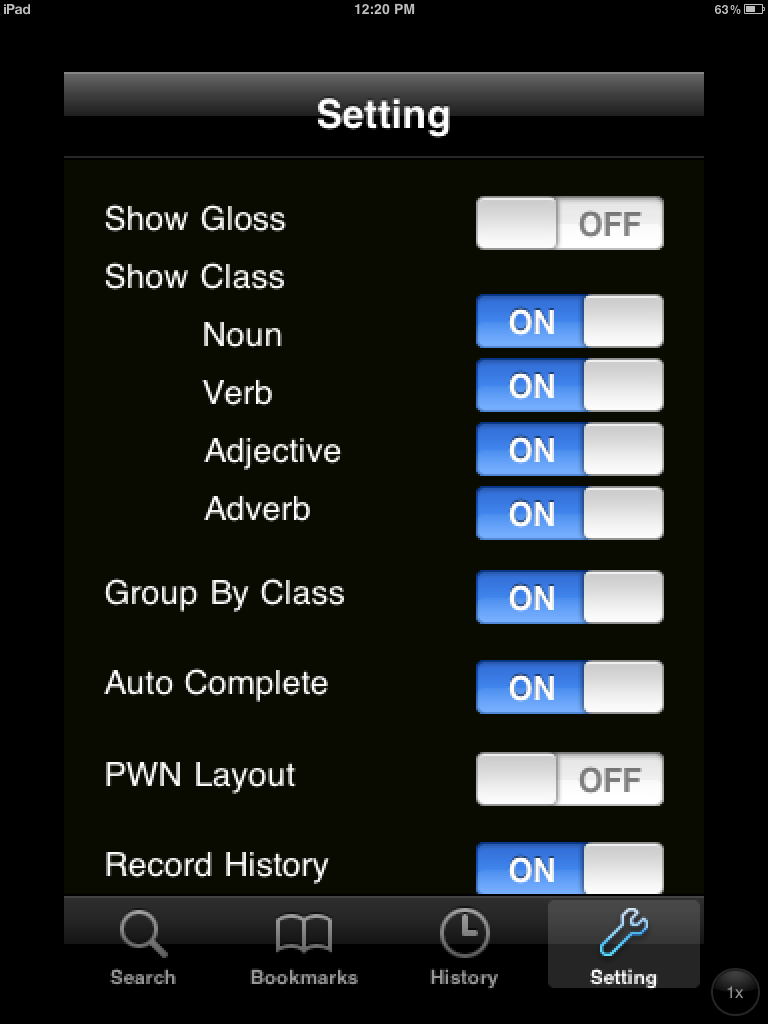
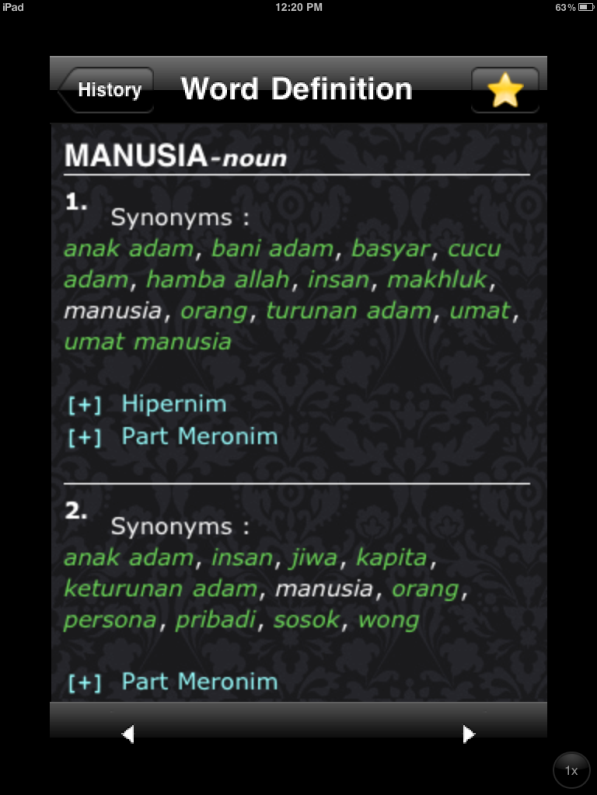
1. -(void) getSetting
2. {
3. if ([[NSFileManager defaultManager] fileExistsAtPath:[[self documentPath] stringByAppendingPathComponent:@"/setting.plist"]])
4. {
5. NSMutableArray \*temp = [[NSMutableArray alloc]initWithContentsOfFile:[[self documentPath] stringByAppendingPathComponent:@"/setting.plist"]];
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)**

1. {
2. [historyList removeObjectAtIndex:0];
3. }
4. [historyList writeToFile:[[self documentPath] stringByAppendingPathComponent:@"/history.plist"] atomically:YES];
5. }
6. indexHistory = [[setting size]intValue]-1;
7. [temp release];
8. }
9. else {
10. NSMutableArray \*temp = [[NSMutableArray alloc]init];
11. for(int i=0;i<9;i++)
12. {
13. if (i!=7)
14. [temp addObject:[[NSNumber alloc]initWithBool:YES]];
15. else
16. [temp addObject:[[NSNumber alloc]initWithBool:NO]];
17. }
18. [temp addObject:@"50"];
19. [setting setShowGloss:[[temp objectAtIndex:0]boolValue]];
20. [setting setNoun:[[temp objectAtIndex:1]boolValue]];
21. [setting setVerb:[[temp objectAtIndex:2]boolValue]];
22. [setting setAdj:[[temp objectAtIndex:3]boolValue]];
23. [setting setAdv:[[temp objectAtIndex:4]boolValue]];
24. [setting setGroupByClass:[[temp objectAtIndex:5]boolValue]];
25. [setting setAutoComplete:[[temp objectAtIndex:6]boolValue]];
26. [setting setPwnLayout:[[temp objectAtIndex:7]boolValue]];
27. [setting setRecordHistory:[[temp objectAtIndex:8]boolValue]];
28. [setting setSize:[temp objectAtIndex:9]];
29. maxHistory = [[setting size]intValue];
30. [temp writeToFile:[[self documentPath] stringByAppendingPathComponent:@"/setting.plist"] atomically:YES];
31. [temp release];
32. }
33. }

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.