Algoritma *Extended Weighted Tree Similarity* untuk Memberikan Solusi Memasak pada J2ME

[1]Yosephine Halim, [2]Ramos Somya, [3]Charitas Fibriani

[1]Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Jl. Diponegoro 52-60 Salatiga, Yuke_chan_girl@yahoo.com,
[2]Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Jl. Diponegoro 52-60 Salatiga, ramos.6005@gmail.com,
[3]Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Jl. Diponegoro 52-60 Salatiga, charitasfibriani@yahoo.com

Abstract

The aim of this thesis is to give the simplicity in cooking with limited ingredients for the users by using mobile technology (J2ME). The aim of this thesis can be done by using Extended Weighted Tree Similarity Algorithm, a calculation to find out the highest weight similarity between the menu which has been inputted in the application as a database; and the inputted ingredients from the users of this application. The conclusion of this thesis is that Algorithm Extended Weighted Tree Similarity can be implemented into mobile technology (J2ME).
Keywords*: extended weighted tree similarity, cooking smart, J2ME*

Abstrak

Tujuan dari penelitian ini adalah memberikan kemudahan dalam memasak dengan bahan yang terbatas bagi pengguna dengan menggunakan teknologi *mobile* (J2ME). Tujuan dari tesis ini dapat dilakukan dengan menggunakan Algoritma *Extended Weighted Tree Similarity*, perhitungan untuk mengetahui kesamaan bobot tertinggi antara menu yang telah diinput dalam aplikasi sebagai *database*, dan bahan-bahan dimasukkan dari pengguna aplikasi ini. Kesimpulan dari tesis ini adalah bahwa Algoritma *Extended Weighted Tree Similarity* dapat diimplementasikan ke dalam teknologi *mobile* (J2ME).
Kata Kunci : *extended weighted tree similarity, cooking smart, J2ME*

1. Pendahuluan

Resep masakan pada umumnya menggunakan media cetak/buku. Buku tersebut berisi panduan memasak untuk berbagai macam masakan. Panduan tersebut memudahkan untuk memasak berbagai jenis masakan, namun dalam mencari satu resep masakan mendapat kesulitan dalam mencari dan hanya dipermudah dengan membaca indeks resep masakan. Hal ini menyebabkan penggunaan waktu yang kurang efisien untuk mencari satu jenis menu masakan.

Pada penelitian kali ini, resep masakan akan diterapkan pada *cooking smart*. Perbedaan dengan panduan resep masakan biasa adalah terdapat fitur yang memudahkan *user* dalam mencari resep masakan. Hanya dengan milih bahan yang dimiliki atau dikehendaki, resep masakan pun bisa didapatkan berdasarkan bahan tersebut.

Pemberian solusi memasak pada penelitian ini, menggunakan algoritma *Extended Weighted Tree Similarity*, di mana proses pencarian dilakukan bukan dengan ditemukannya inputan *user* saja, melainkan dengan melakukan perhitungan tingkat kemiripan pohon utama dengan inputan user. Pohon utama tersebut dibuat terlebih dahulu sebagai panduan dalam perhitungan. Penggunaan algoritma *Extended Weighted Tree Similarity* pada penelitian ini, yaitu dengan membandingkan kemiripan antara menu masakan yang telah ada dan menu masakan yang diinputkan oleh *user*. Metode ini disusun berdasarkan *tree* yang memiliki *node* berlabel, cabang berlabel serta berbobot.

Penelitian ini menggunakan J2ME sebagai pengimplementasi dari *cooking smart* yang menggunakan *Extended Weighted Tree Similarity*. Hal ini dikarenakan mayoritas *platform*

mobile device masih menggunakan *platform java*. Tujuan lain adalah untuk membuat kemudahan dalam membawa dan membuka resep masakan di mana pun dan kapan pun.

2. Tinjauan Pustaka

Extended Weighted Tree Similarity

Tingkat kemiripan pada algoritma *Extended Weighted Tree Similarity* ditentukan dalam *range* nilai 0 sampai dengan 1. Kemiripan suatu *keyword* dilihat dari bagaimana hasil perhitungan. Apabila perhitungan tersebut mendekati nilai 1, hasil yang didapat adalah *keyword* tersebut memiliki tingkat kemiripan yang hampir sama. Sebaliknya, perhitungan mendeketi nilai 0, *keyword* tersebut tingkat kemiripannya berbeda dengan informasi yang telah ada. Algoritma ini menggunakan representasi *tree* sebagai input untuk mengkalkulasi derajat kemiripan antara 2 objek [1].

Dalam pencarian semantik yang menggunakan algoritma *weighted tree similarity*, metadata disusun berdasarkan *tree* yang memiliki *node* berlabel, cabang berlabel serta berbobot. Struktur metadata *tree* disusun berdasarkan informasi semantik semacam taksonomi, ontologi, *preference*, sinonim, homonim, dan *stemming*. Oleh karena itu, metadata yang digunakan dapat lebih merepresentasikan isi sebuah artikel serta hasil pencarian dapat lebih tepat (*precision*).

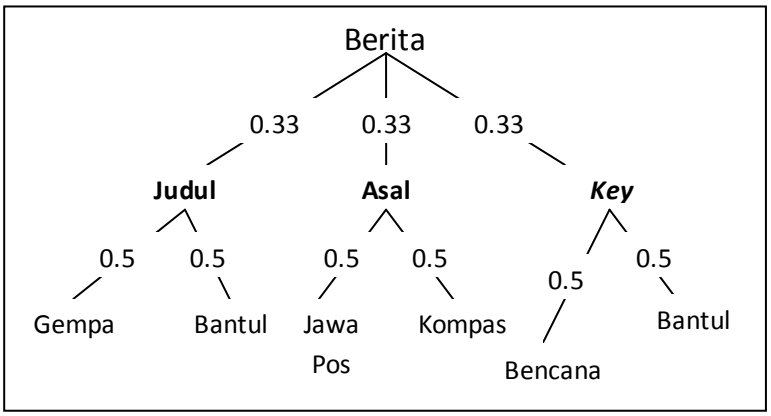
Algoritma *weighted tree similarity* memiliki keunikan karena memiliki representasi *tree* yang berbeda dengan yang lain. *Tree* yang dipergunakan memiliki *node* berlabel, cabang berlabel serta berbobot. Cabang yang berlabel memberikan pemahaman lebih kepada label *node*nya. Begitu pula bobot cabang memungkinkan memberikan tingkat kecenderungan kepada cabang tertentu lebih dari yang lain.

Nilai kemiripan tiap pasangan *subtree* berada diantara nilai 0 dan 1. Nilai 0 bermakna berbeda sama sekali sedangkan 1 bermakna identik. Kedalaman (*depth*) dan lebar (*breadth*) *tree* tidak dibatasi. Algoritma penghitung kemiripan *tree* secara rekursif menjelajahi tiap pasang *tree* dari atas ke bawah mulai dari kiri ke kanan. Algoritma mulai menghitung kemiripan dari bawah ke atas ketika mencapai *leaf node*.

Nilai kemiripan tiap pasang *subtree* di level atas dihitung berdasar kepada kemiripan *subtree* di level bawahnya. Sewaktu penghitungan, kontribusi bobot cabang juga diperhitungkan [2].

Suatu *tree* terdiri dari 1 atau lebih *parent* yang masing-masing dapat memiliki atribut berupa bobot serta nama *parent*. Setiap *parent* terdiri *identifier*, yang mempunyai atribut bobot [1].

Keseluruhan *tree* direpresentasikan dengan menggunakan fungsi *vector* dan perhitungannya dengan cara membandingkan antara isi *vector* utama yang berupa *tree* dengan *vector* inputan dari *user*. Apabila sama, maka dilakukan perkalian sesuai dengan bobot *vector* inputan dan bobot *vector* utama. Gambar 1 merupakan contoh representasi *tree* berita tentang gempa bantul.



Gambar 1 Contoh Repesentasi *Tree* [1]

Sebuah *tree* berita memiliki dari tiga *parent* yaitu, judul, asal, dan *key*. Masing-masing *parent* memiliki bobot, yang jika dijumlahkan akan bernilai 1. bobot masing-masing *parent* diperoleh dari persamaan (1).

$$W_i = 1 / n \quad (1)$$

dimana
W_i : bobot *parent* ke-*I*, *n* : total *parent* yang ada

Setiap *parent* memiliki anak yang disebut dengan *identifier*. Berdasarkan Gambar 1, yang menjadi *identifier* dari *parent* judul adalah gempa dan bantul. *Identifier* dari *parent* asal adalah jawa pos dan kompas, sedangkan *identifier* untuk *parent key* adalah bencana dan bantul. Setiap *identifier* juga memiliki bobot yang diperoleh dari persamaan (2).

$$W_{ind} = frek / n \quad (2)$$

dimana:
W_{ind} : bobot *identifier* ke-*i*
frek : jumlah kemunculan *identifier*
n : total *identifier* yang ada

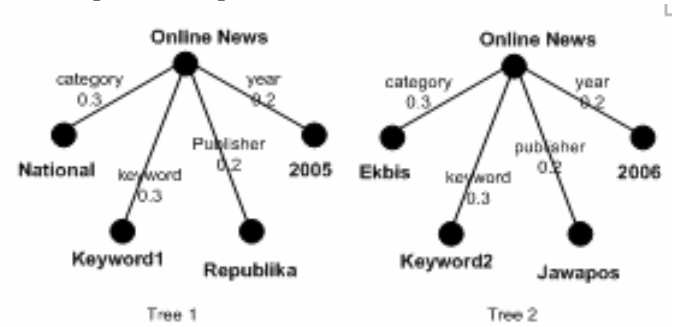
Setelah menemukan bobot *parent* dan *identifier*, dapat dilakukan langkah selanjutnya dengan mengetahui *tree* inputan dari *user*.

Dengan membandingkan dua *tree* yang memiliki nama *parent* yang sama, perhitungan tingkat kemiripan dapat dilakukan. Perhitungan dilakukan dengan membandingkan tiap *parent* yang sama beserta dengan *identifier* yang dimilikinya [1]. Perhitungan tersebut dapat dilihat pada persamaan (3).

$$BK = \sum ((W_{(i)} * W_{(j)}) * (\sum W_{ind(i)} * W_{ind(j)})) \quad (3)$$

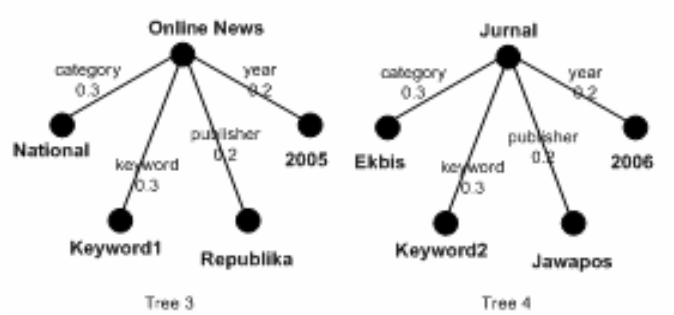
dimana:
BK : bobot kemiripan
$W_{(i)}$: bobot *parent* utama ke-i
$W_{(j)}$: bobot *parent* inputan ke-j
$Wind_{(i)}$: bobot *identifier* utama ke-i
$Wind_{(j)}$: bobot *identifier* inputan ke-j

Pada kondisi nyata proses pencocokan tidak selalu akan menemukan kesamaan baik pada label *root* maupun pada label *node*. Gambar 4 adalah pasangan *tree* yang memiliki label *root* sama tetapi label *node* berbeda serta pasangan *tree* yang memiliki label *root* dan label *node* berbeda. Penjelasan tersebut dapat dilihat pada Gambar 2 [3].



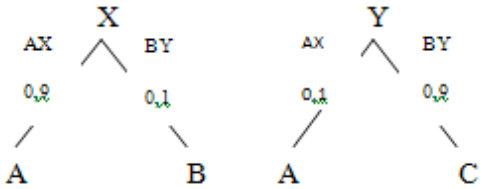
Gambar 2 Pasangan Tree yang Memiliki Label *Root* Sama tetapi Label *Node* Berbeda[3]

Pada Gambar 2 dan Gambar 3, label *node* untuk kedua pasangan *tree* itu berbeda sehingga *similarity* kedua pasangan *tree* ini berdasarkan label *node* adalah 0. Akan tetapi, jika perhitungan nilai *similarity* juga melibatkan unsur label *root* maka secara logika pasangan *tree* pada Gambar 2 masih memiliki kesamaan dibandingkan dengan pasangan *tree* pada Gambar 3 [3]



Gambar 3 Pasangan *Tree* yang Memiliki Label *Root* dan Label *Node* Berbeda [3]

Pada kasus ini Algoritma *Extended Weighted Tree similarity* memberikan nilai *Node-Indentity Fraction (N)* untuk pasangan *tree* pertama sebesar 0,1 dan nilai N untuk pasangan *tree* kedua sebesar 0. Kasus tertentu, pemberian bobot juga berpengaruh pada hasil *similarity*, terdapat dua pohon yaitu pohon x dan pohon y. pada pohon x terdapat label *root* dan label *node* yang berbeda. Penjelasan ini dapat dilihat pada Gambar 4 [3].



Gambar 4 Pasangan Tree yang Memiliki Bobot yang Berbeda [3]

Similarity kedua label pada cabang "AX" adalah 1 karena keduanya sama, sedangkan cabang "BY" yang punya label berbeda adalah 0. Total Nilai *similarity* adalah nilai *similarity* didapat dari perhitungan yang dapat dilihat pada perhitungan (3).

$$\begin{aligned} \textit{Similarity}(X,Y) = \quad & \text{sim}(\text{“A”.”A”}) * [(\text{bobot ”AX”} + \text{bobot “AX”}) /2] + \\ & \text{sim}(\text{“B”,”C”}) * [(\text{bobot ”BY”} + \text{bobot “BY”}) /2] \end{aligned} \tag{3}$$

Hasil pengamatan ini menunjukkan bahwa nilai *similarity* dua *tree* yang memiliki label *node* persis sama menyebabkan distribusi bobot cabangnya tidak berpengaruh terhadap nilai *similarity*. Nilai bobot akan memiliki pengaruh jika ada perbedaan nilai minimal pada salah satu cabangnya [3].

Java 2 Micro Edition

Komponen-komponen J2ME terdiri dari *Java Virtual Machine* (JVM) yang digunakan untuk menjalankan aplikasi *Java* pada *emulator* atau *handheld device*, *Java API* (*Application Programming Interface*) dan *tools* lain untuk pengembangan aplikasi *Java* semacam *emulator Java Phone*, *emulator* Motorolla dari J2ME *wireless toolkit*. Dalam pengembangan aplikasi *wireless* dengan *Java*, J2ME dibagi menjadi dua buah bagian diantaranya ialah bagian *configuratio*n dan *profile* [4].

J2ME merupakan tulang punggung bagi perkembangan teknologi *m-commerce* saat ini. Beberapa keunggulan dari *user*an J2ME adalah (1) Menciptakan aplikasi yang bersifat *portable*, (2) Sistem keamanan yang baik, (3) Aplikasi bisa digunakan dalam mode *offline* ataupun *online*,

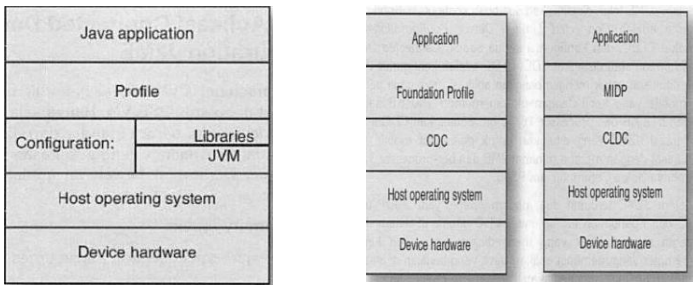
(4) Aplikasi mengadopsi karakteristik utama *Java*, yaitu "*Write once, run anywhere, any time, and over any device*" [5].

Di dalam mengembangkan aplikasi *Java*, J2ME sesungguhnya merupakan bagian integral dari *Java 2 Standard Edition* atau sering disebut J2SE. Oleh karena J2ME merupakan *subset* dari J2SE, maka tidak semua *library* J2SE dapat digunakan pada J2ME. Namun sebaliknya, Sun mengembangkan *library* khusus, di mana *library* ini tidak ada pada J2SE. Dalam implementasinya teknologi J2ME memiliki batasan terutama yang berkaitan dengan perangkat keras, merek *mobile phone*, atau kemampuan *mobile phone* tersebut.

Di dalam J2ME telah didefinisikan dua konfigurasi. Konfigurasi pertama adalah *Connected Limited Device Configuration* atau disingkat CLDC, dan konfigurasi yang kedua adalah *Connected Device Configuration* atau disingkat CDC. CLDC adalah konfigurasi J2ME yang ditujukan untuk mengembangkan aplikasi *Java* dengan perangkat *mobile* yang kecil dengan ukuran memori 160KB sampai 512KB dan *processor* 16/32 bit. Sedangkan CDC adalah konfigurasi J2ME yang ditujukan untuk perangkat *mobile* yang lebih besar dengan memori minimal 2MB dan ber-*processor* 32bit.

Di dalam J2ME terdapat 2 macam *profile* yaitu MIDP (*Mobile Information Device Profile*) untuk CLDC dan *Foundation Profile* untuk CDC. *Profile* di dalam J2ME berperan sebagai layer yang menyediakan fungsi API kepada programmer yang membuat aplikasi *Java* yang berjalan di atasnya. *Configuration* juga menyediakan API seperti *profile*, hanya saja fungsi API di level *configuration* tidak direkomendasikan untuk diakses oleh programmer secara langsung. *Configuration* bersama dengan *profile* pada akhirnya menciptakan J2ME *runtime environment*. Arsitektur dari J2ME secara umum dan khusus dapat dilihat pada Gambar 5 [6].

Aplikasi yang berjalan pada sebuah perangkat yang mendukung MIDP disebut dengan MIDlets, atau lebih singkatnya MIDlet merupakan aplikasi yang dibuat menggunakan *Java 2 Micro Edition* dengan *profile Mobile Information Device Profile* (MIDP). MIDP dikhususkan untuk digunakan pada *handset* dengan kemampuan CPU, memori, *keyboard* dan layer yang terbatas, seperti *handphone*, pager, PDA dan sebagainya [4].



Gambar 5 Arsitektur J2ME Secara Umum dan khusus [6]

MIDlet memiliki struktur direktori sebagai berikut:

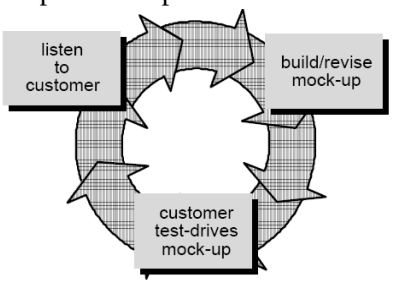
1. Src, menyimpan *source code* untuk MIDlet dan kelas lain yang diperlukan.
2. Res, menyimpan sumber daya yang dibutuhkan oleh MIDlet, misalnya gambar *icon*.
3. Lib, menyimpan *file* JAR atau ZIP yang berisi *library* tambahan yang dibutuhkan MIDlet.
4. Bin, menyimpan *file* JAR, JAD dan *file manifest* yang berisi muatan komponen MIDlet [7].

3. Metode Perancangan

Metode yang akan digunakan pada sistem ini menggunakan Metode *Prototyping Model*. Metode ini merupakan proses iteratif dalam pengembangan sistem di mana kebutuhan diubah ke dalam sistem yang bekerja (*working system*) yang secara terus menerus diperbaiki melalui kerjasama antara *user* dan analis. *Prototype* juga bisa dibangun melalui beberapa *tool* pengembangan untuk menyederhanakan proses. *Prototyping* merupakan bentuk *Rapid Application Development* (RAD). RAD memiliki beberapa kelemahan, di antaranya (1) RAD mungkin mengesampingkan prinsip-prinsip rekayasa perangkat lunak, (2) Menghasilkan

inkonsistensi pada modul-modul system, (3) Tidak cocok dengan standar, (4) Kekurangan prinsip *reusability* komponen [8].

Metode *Prototyping model* dapat dilihat pada Gambar 6.



Gambar 6 Metode Pengembangan *Prototyping* [8]

Tahapan pada *prototyping* adalah sebagai berikut:

1. Analis bekerja dengan tim untuk mengidentifikasi kebutuhan awal untuk sistem.
2. Langkah selanjutnya adalah membuat *prototype*. Ketika *prototype* telah selesai, *user* bekerja dengan *prototype* itu dan menyampaikan pada analis apa yang mereka sukai dan tidak mereka sukai.
3. Analis menggunakan *feedback* ini untuk memperbaiki *prototype*.
4. Versi baru diberikan kembali ke *user*.

Ulangi langkah-langkah tersebut sampai *user* merasa puas.

Pada model *prototype* memiliki beberapa keuntungan seperti, melibatkan *user* dalam analisis dan desain, memiliki kemampuan menangkap kebutuhan secara kongkret daripada abstrak. *Prototype* dapat digunakan secara *standalone* dan dapat memperluas SDLC (*System development life cycle*) [8].

Tahapan Metode *Prototyping*

Tahapan-tahapan *prototype* dalam penelitian ini dinyatakan telah sesuai dengan kebutuhan *user* pada *prototype* kedua. Tahapan-tahapan tersebut yaitu :

***Prototyping* 1**

Tahapan awal analis mengumpulkan data awal yang berdasarkan pada keinginan dari *user*. Pengumpulan data tersebut diambil dari kuisioner yang disebar sebanyak 30 lembar. Penyebaran kuisioner tersebut dikategorikan berdasarkan jenis kelamin wanita dan umur antara 17-25 tahun. Hasil dari kuisioner tersebut diketahui; umumnya wanita tidak menyukai memasak dan lebih suka untuk membeli masakan yang telah saji, dan kesulitan dalam menentukan menu masakan yang cocok dengan bahan yang dimiliki.

Setelah melakukan kuisioner, tahapan selanjutnya dalam metode *prototyping* yaitu *build/revise mock-up* atau membangun aplikasi secara cepat. Dalam tahap ini dihasilkan aplikasi *cooking smart prototype* pertama dihasilkan proses memasuki tahapan selanjutnya yaitu *customer test driver mock-up*. Pada tahap ini aplikasi *cooking smart* dapat diserahkan kepada *user* untuk dievaluasi untuk mengetahui kekurangan dan kendala-kendala pada *prototype* pertama.

***Prototyping* 2**

Pada tahap penyerahan *prototype* pertama didapatkan informasi baru tentang kebutuhan aplikasi yang akan dibangun nantinya, didapat kebutuhan sistem dari hasil kuisioner tahap dua yaitu (1) Aplikasi dapat menyimpan menyimpan sejumlah resep masakan, (2) Bahan masakan yang dipakai berdasarkan gizi masakan, yaitu karbohidrat, protein, dan vitamin mineral, (3) Aplikasi dapat memberikan solusi berdasarkan bahan yang dimiliki/dikehendaki.

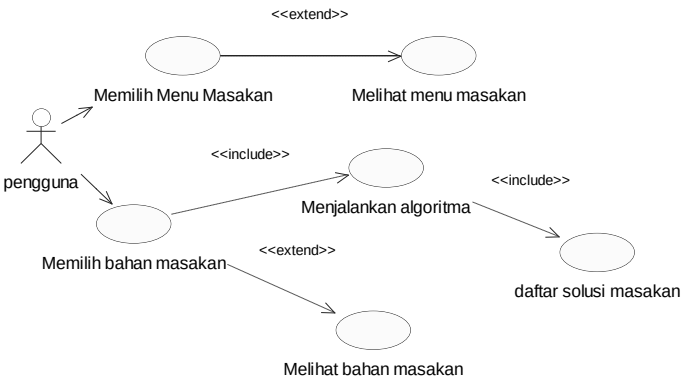
Setelah mendapatkan informasi baru tentang kebutuhan aplikasi *cooking smart*, dapat dikembangkan aplikasi *cooking smart prototype* pertama sesuai dengan hasil evaluasi tersebut menjadi *prototype* kedua. Pada pembangunan *prototype* kedua aplikasi ditekankan pada pemberian solusi memasak berdasarkan bahan. Selain aplikasi *cooking smart* memanfaatkan teknologi *mobile* yaitu J2ME, sebagai sarana pengembangannya. Hasil dari *prototype* kedua

dievaluasi kembali dan telah memenuhi kebutuhan dari *user* yang umumnya wanita muda. Dan tahapan dari *prototyping* berakhir pada *protoype* kedua dikarenakan hasil aplikasi telah memenuhi kebutuhan *user*.

Perancangan analisa sistem *Cooking Smart* menggunakan metode UML (*Unified Modelling Language*) akan dipaparkan sebagai berikut:

Use Case Diagram

User yang menggunakan aplikasi ini memiliki dua aktivitas. Aktivitas pertama, *user* dapat memilih menu masakan. Namun dalam menu masakan, *user* pasti melihat menu masakan tersebut. Aktivitas kedua, *user* dapat memilih bahan masakan. Pada saat *user* telah memilih bahan masakan, user dapat melihat bahan masakan tersebut sebelum pada akhirnya menjalankan algoritma untuk mendapatkan solusi memasak. Penjelasan singkat di atas dapat dilihat pada Gambar 7.

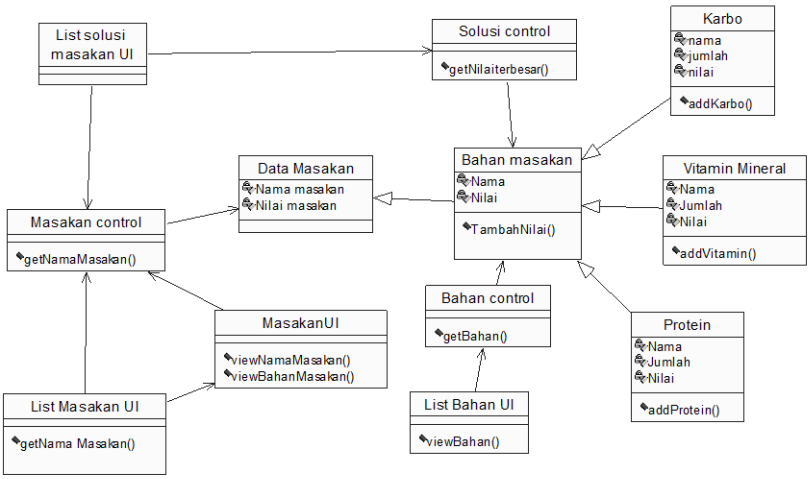


Gambar 7 *Use Case Diagram*

Class Diagram

Pada *class diagram* dijelaskan alur data yang ada pada aplikasi ini. Terdapat data masakan yang berisi nama masakan dan nilai masakan. Tujuan dari nilai masakan adalah untuk menyimpan nilai masakan yang tertinggi dan berguna untuk memberikan solusi yang terbaik. Data masakan memiliki sebuah *controller* yang bernama masakan *control*. Tujuan dari masakan *control* adalah untuk mendapatkan data dari data masakan yang berupa nama masakan dan bahannya. Masakan *control* digunakan oleh *list* masakan UI dan Masakan UI serta *list* solusi masakan UI.

Selain terdapat data masakan, terdapat juga turunan dari data masakan yang berupa bahan masakan. Bahan masakan berisi nama bahan dan nilai, pada nama bahan dibagi berdasarkan gizi bahan tersebut, yaitu karbohidrat, protein dan vitamin mineral. Untuk masing-masing gizi tersebut, terdapat nama, nilai dan jumlahnya. Bahan masakan memiliki dua *controller*, yaitu bahan *control* dan solusi *control*. Bahan *control* berguna untuk menampilkan data bahan pada *list* bahan UI, sedangkan solusi control digunakan untuk menghitung bahan dan menentukan nilai tertinggi pada bahan tersebut dan digunakan *list* solusi masakan UI. Penjelasan singkat tentang *class diagram* dapat dilihat pada Gambar 8.



Gambar 8 *Class Diagram*

Perancangan pohon pada tiap masakan dibuat berdasarkan 3 kriteria, yaitu karbohidrat, protein, dan vitamin mineral. Tiap-tiap masakan dibagi 3 berdasarkan kriteria tersebut. Tiap kriteria diisi berdasarkan bahan resep masakan yang terkandung untuk masing-masing kategori gizi. Sama halnya dengan perancangan pohon inputan, yang berbeda adalah nama masakan dicari dengan perhitungan bobot kemiripan tertinggi.

4. Hasil dan Pembahasan

Aplikasi yang dibuat merupakan aplikasi *mobile* yang berisikan tentang panduan memasak. Panduan memasak dibuat sederhana dengan maksud dapat dipahami oleh semua kalangan. Untuk semakin membuat aplikasi ini tidak hanya sebatas memberikan panduan memasak, aplikasi ini menambahkan menu yaitu, menu memberikan solusi.

Penelitian ini berfokuskan pada menu pemberian solusi memasak dengan algoritma *Extended Weighted Tree Similarity* sebagai landasannya. Penyimpanan data masakan menggunakan fungsi *Vector*. Fungsi *Vector* sama dengan fungsi *ArrayList*, namun fungsi *ArrayList* tidak dapat digunakan pada J2ME.

Vector memiliki dua atribut utama yaitu, kapasitas dan penambahan kapasitas. Penambahan kapasitas menentukan berapa jumlah indeks yang akan ditambahkan jika indeks saat ini sudah tidak mencukupi [9].

Untuk mempermudah *user*, aplikasi dibuat dengan bantuan gambar yang berhubungan dengan isi menu tersebut. Fungsi yang digunakan adalah *canvas*, dengan adanya fungsi tersebut aplikasi dapat dimodifikasi dengan gambar dan diatur *background* warna tampilannya. Aplikasi ini berisikan beberapa menu, yaitu menu Resep Masakan, menu Solusi Masakan, dan menu Keluar, seperti pada Gambar 9(a).

Apabila *user* memilih menu Resep Masakan, aplikasi akan langsung memunculkan daftar menu masakan. Dengan bentuk *list* memudahkan *user* dalam memilih dan bisa langsung mengakses data tersebut, selain itu dalam *list* tersebut dilengkapi *icon* yang merupakan gambar dari hasil masakan. Bentuk *list* masakan dapat dilihat pada Gambar 9(b).

User dengan bebas memilih masakan yang tersedia pada aplikasi *Cooking Smart*. Ketika *user* memilih salah satu menu masakan yang ada, halaman tampilan akan berubah dengan tampilan menu masakan tersebut. Isi dari tampilan tersebut berupa bahan masakan dan cara membuat masakan serta ditampilkan gambar dari masakan tersebut. Tujuan dari gambar masakan adalah memberikan gambaran tentang hasil jadi masakan yang akan dibuat. Tampilan menu masakan dapat dilihat pada Gambar 9(c).

Gambar 9 (a)Tampilan Awal Menu Resep Masakan, (b) *Form* Daftar Resep Masakan, (c) *Form* Resep Masakan Gado-gado Siram

Untuk mempermudah *user* dalam memilih bahan masakan, pemilihan bahan digunakan *checkbox* sehingga *user* pun dapat memilih lebih dari satu bahan masakan. Tampilan *list* bahan masakan dapat dilihat pada Gambar 10.

Gambar 10 *Form* Memilih Bahan Masakan

Untuk mempermudah cara perhitungan bobot perhitungan, nilai *identifier* antara masakan dan inputan *user* dikalikan terlebih dahulu. Dan bobot kemiripan akan didapat dari nilai perkalian antara bobot *parent* dan bobot *identifier*. Contoh menu Solusi Masakan dengan bahan nasi, jagung, ayam, selada dan tomat dapat dilihat pada Gambar 11(a) dan Gambar 11(b).

Gambar 11(a) Tampilan Memilih Bahan Nasi, Jagung dan Ayam

Gambar 11(b) Tampilan Memilih Bahan Selada dan Tomat

Gambar 16 merupakan contoh *user* yang sedang mencari solusi memasak dengan bahan nasi, jagung, ayam, selada dan tomat. Setelah selesai memilih bahan masakan akan muncul tampilan pada Gambar 12.

Gambar 12 *Form List* Resep Solusi Memasak

Gambar 12 merupakan hasil perhitungan *Extended Weighted Tree Similarity* yang telah berupa hasil akhir, yaitu resep masakan dengan nilai bobot kemiripan tertinggi. Resep solusi masakan dapat dalam jumlah banyak, tidak menutup kemungkinan hasil tertinggi suatu masakan adalah sama. Apabila *user* ingin melihat resep masakan yang menjadi solusi, *user* tinggal memilih *button* pilih dan resep masakan tersebut akan langsung keluar.

5. Simpulan

Aplikasi *Cooking Smart* dapat memberikan kemudahan bagi *user* untuk menggunakan aplikasi ini. Hal ini dikarenakan aplikasi diimplementasikan pada aplikasi *mobile* yaitu J2ME. Pemilihan *platform java* melihat dari banyaknya aplikasi *mobile* yang menggunakan *platform java* seperti Nokia, Sony Ericson dan Blackberry. Aplikasi *mobile* memberikan kemudahan dalam menggunakan di mana, kapan dan siapa saja.

Penerapan algoritma *Extended Weighted Tree Similarity* pada J2ME dengan fungsi *vector* memberikan kemudahan dalam membuat *tree* dan membandingkan antara *tree* utama dan *tree* inputan. Pemanfaatan fungsi *vector* dapat digunakan untuk menyimpan data masakan yang berupa bahan-bahan masakan. Bahan-bahan masakan tersebut dimasukkan dalam *vector* dan dibentuk menjadi pohon yang digunakan sebagai perbandingan.

Algoritma *Extended Weighted Tree Similarity* membantu dalam pemberian solusi resep masakan terbaik dengan berdasarkan bahan masakan yang dipilih. Hasil solusi tersebut didapat dari perhitungan nilai terbaik atas kemiripan bahan masakan yang telah ada dengan bahan masakan yang dipilih oleh *user*. Solusi masakan dapat digunakan sebagai panduan mencocokkan bahan yang dimiliki atau dikehendaki dikarenakan keterbatasan bahan masakan.

6. Daftar Pustaka

[1] Yulianti, Tjiong Debora, 2010, *Perancangan Dan Implentasi Personalisasi Modul Similaritas Pencarian Lowongan Kerja*, Universitas Kristen Satya Wacana

[2] Sarno, Riyanarto, dan Faisal Rahutomo, 2008, Penerapan Algoritma Weighted Tree Similarity untuk Pencarian Semantik, Juti 7: 35-42

[3] Solihin, Firdaus, dan Riyanarto Sarno, 2006, Penerapan Arsitektur Agent Matcher Menggunakan Algoritma Extended Weighted Tree Similarity untuk Menyediakan Informasi yang Optimal pada Handheld Device, Seminar Nasional Pascasarjana VI 2006 :38-43

[4] Bp, Agung, Kodrat IS dan Adjie Wibowo. 2005. Simulasi Aplikasi Java 2 Platform Micro Edition (J2ME) - Java Midlet pada Jadwal Ujian. http:/www.elektro.undip.ac.id/ (diakses tanggal 10 Februari 2011)

[5] Irawan. 2008. *Java Mobile untuk Orang Awam*. Palembang: Penerbit Maxikom

[6] Irawanto, Djon. 2007. *Membangun Object Oriented Software dengan Java dan Object Database*. Jakarta: PT ElexMedia Komputindo

[7] Nyura, Yusni, 2010, Pembuatan Aplikasi Pembelajaran Bahasa Inggris pada Handphone dengan J2ME, Jurnal Informatika Mulawarman 5:18-27
[8] Fatta, Hanif Al. 2007. *Analisis dan Perancangan Sistem Informasi untuk Keunggulan Bersaing Perusahaan dan Organisasi Modern*. Yogyakarta: C.V ANDI OFFSET (Penerbit Andi)
[9] Hakim, Rachmand S. dan Ir. Susanto, M. Si, 2009 , *Mastering Java: Konsep Pemrograman Java dan Penerapannya untuk Membuat Software Aplikasi*, Jakarta: PT Elex Media Komputindo