IMPLEMENTASI ALGORITMA *KNUTH MORRIS PRATT* PADA PENCARIAN KOLEKSI MASAKAN KHAS BALI BERBASIS *ANDROID*

I Putu Agus Doni Setiawan

Program Studi Teknik Informatika, STMIK Widya Cipta Dharma
Jl. Prof. M. Yamin No. 25 Samarinda Kalimantan Timur 75123
Telp: (0541) 736071, Fax: (0541) 203492
E-mail:putudoni73@gmail.com

ABSTRAK

Algoritma *Knuth Morris Pratt* adalah salah satu algoritma pencarian *string* yang dimana semua informasi yang ada disimpan dan digunakan untuk melakukan pergeseran lebih jauh dan algoritma ini menghemat perbandingan, yang selanjutnya akan meningkatkan kecepatan pencarian. Algoritma ini menemukan semua kemunculan dari *pattern* dengan panjang *n* di dalam teks dengan panjang *m* dengan kompleksitas waktu *O*(*m*+*n*). Algoritma ini hanya membutuhkan *O*(*n*) ruang dari memory internal jika teks dibaca dari file eksternal. Semua besaran *O* tersebut tidak tergantung pada besarnya ruang alpabeth. Algoritma ini merupakan jenis *Exact String Matching Algorithm* yang merupakan pencocokan *string* secara tepat dengan susunan karakter dalam *string* yang dicocokkan memiliki jumlah maupun urutan karakter dalam *string* yang sama.

Aplikasi resep masakan Bali dikembangkan dengan metode pengembangan sistem RAD (*Rapid Aplication Development*), metode pengumpulan data dengan wawancara, observasi dan studi pustaka, analisa perancangan sistem dengan menggunakan UML (*Unifed Modelling Language*). Dalam mengembangkan aplikasi resep masakan Bali digunakan bahasa pemrograman JAVA, XML, *SQLite*, *Android Studio* sebagai *Editor Android*, dan *Adobe Photoshop* sebagai *editor image*. Metode pengujian pada pencarian koleksi masakan khas Bali menggunakan metode pengujian *White Box* dan *Black Box*.

Hasil dari penelitian yaitu algoritma *Knuth Morris Pratt* dapat di implementasikan dalam pencarian koleksi masakan khas Bali berbasis *Android* menggunakan bahasa pemrograman JAVA. Dengan adanya aplikasi ini maka masyarakat umum dapat koleksi masakan khas Bali hanya melalui *gadget*.

Kata Kunci : *Android*, Resep Masakan, Resep Bali, *Knuth Morris Pratt*

---

1. PENDAHULUAN

Bali merupakan salah satu daerah yang memiliki potensi besar di bidang pariwisata. Pulau Bali menjadi sangat dikenal oleh dunia internasional karena keindahan alamnya, kesenian, berbagai ragam budaya, dan tradisi sosial kemasyarakatan yang dijiwai oleh agama Hindu. Seiring dengan berkembangnya industri pariwisata dan meningkatnya kompetisi di antara tempat tujuan wisata, kebudayaan lokal menjadi hal yang berharga sebagai produk dan aktivitas untuk menarik wisatawan. Wisata kuliner yang merupakan salah satu budaya lokal mempunyai peran penting karena masakan juga bisa menjadi sumber pengalaman wisatawan. Wisata kuliner muncul dari keinginan para wisatawan itu sendiri yang ingin mendapatkan dua (2) pengalaman tidak saja dari keindahan alam, tetapi juga dari menggunakan wisata kuliner sebagai alat penarik wisatawan dan banyak yang menggunakan pariwisata untuk mempromosikan kuliner daerahnya.

Saat ini, resep kuliner khas Bali banyak tersedia di koran, majalah maupun tabloid. Resep tersebut tidak terklasifikasi menurut kategori jenis resepnya, sehingga menimbulkan kesulitan pada saat pencarian menurut jenis makanan, bahan atau lainnya. Bisa juga dengan pengarsipan koleksi resep masakan dibutuhkan rak khusu untuk menyimpan kumpulan koran, majalah, dan tabloid resep yang dimiliki agar tetap awet dan tersebut untuk menjaga koleksi resep-tidak ada resep yang hilang.

Mengingat adanya teknologi yang semakin hari semakin berkembang, khususnya teknologi *Android* yang semakin pesat. Pada umumnya, alat pencarian dalam *Andorid*menggunakan beberapa algoritma, seperti *bruto force* ataupun *knuth morris pratt* yang

digunakan untuk mempermudah dalam penyeleksian *string*. Namun beberapa algoritma membutuhkan waktu yang lama dan kurang efisien. Oleh sebab itu, ingin diimplementasikan algoritma yang diduga cepat dan efisien dalam pencarian *string* resep masakam khas Bali, yaitu algoritma *Knuth Morris Pratt* (KMP). Dengan demikian, penelitian ini diberi judul "Implementasi Algoritma *Knuth Morris Pratt* pada Pencarian Koleksi Masakan Khas Bali Berbasis *Android*."

2. RUANG LINGKUP PENELITIAN

Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka rumusan masalah penelitian ini sebagai berikut :

"Bagaimanakah pengimplementasian algoritma *Knuth Morris Pratt* pada pencarian koleksi masakan khas Bali berbasis *Android* ?"

Batasan Masalah

Berdasarkan latar belakang dan rumusan masalah, maka batasan masalah penelitian adalah :

1. Aplikasi dibuat dengan bahasa pemrograman XML, JAVA, dan *SQLite* untuk *database*.
2. Simulasi data resep untuk aplikasi diambil dari koleksi resep Rumah Makan Bebek Tepi Sawah, Bigmall Samarinda.
3. Metode pengembangan sistem mengikuti tahapan RAD (*Rapid Application Development*) jenis *Extreme Programming*(XP).
4. Untuk menunjukkan bahwa sistem mengikuti algoritma *Knuth Morris Pratt*, maka sistem akan diuji dengan simulasi *string* pencarian menu favorit dari arah kiri, kanan, dan tengah.

3. BAHAN DAN METODE

Adapun bahan dan metode yang digunakan dalam sistem ini adalah :

2.1.1 Algoritma *Knuth Morris Pratt*

J.C, Prasad et al (2010), Algoritma *KnuthMorrisPratt* (KMP) dikembangkan oleh D. E. Knuth, bersama dengan J. H. Morris dan V. R.Pratt. Untuk pencarian *string* dengan menggunakan algoritma *BruteForce*, setiap kali ditemukan ketidakcocokan *pattern* dengan teks, maka *pattern* akan digeser satu karakter ke kanan. Algoritma ini membandingkan pola dengan teks dari kiri ke kanan.

Menurut Rasool, Akhtar et al (2012), Pada algoritma *KnuthMorris-Pratt*, kita memelihara informasi yang digunakan untuk melakukan jumlah pergeseran. Algoritma KMP digunakan untuk bekerja pada arsitektur yang mendukung *string* paralel ukuran yang lebih besar.

Persoalan pencarian *string* dirumuskan sebagai berikut :

1. Sebuah teks (*text*), yaitu sebuah (*long*) *string* yang panjangnya n karakter.
2. *Pattern*, yaitu sebuah *string* dengan panjang m karakter (m<n) yang akan dicari dalam teks.

Cari lokasi pertama di dalam teks yang bersesuaian dengan *pattern*. Aplikasi permasalahan pencocokan *string* biasa ditemukan dalam pencarian sebuah kata dalam dokumen (misalnya menu *Find* dalam *Microsoft Word*).

Dalam algoritma pencarian *string*, teks diasumsikan berada di dalam memori, sehingga bila kita mencari *string* di dalam sebuah arsip, maka semua isi arsip perlu dibaca terlebih dahulu, kemudian disimpan di dalam memori. Jika *pattern* muncul lebih dari sekali di dalam teks, maka pencarian hanya akan memberikan keluaran berupa lokasi *pattern* ditemukan pertama kali.

Algoritma menggunakan informasi tersebut untuk membuat pergeseran yang lebih jauh dan melakukan evaluasi waktu. Ada 2 tahap untuk menyelesaikan Algoritma *Knuth MorrisPratt* yaitu *Prefix Function* dan *String Matching*.

1. Algoritma *Knuth MorrisPratt String Matching*

   Algoritma *Knuth MorrisPratt* merupakan salah satu algoritma yang sering digunakan untuk menyelesaikan masalah pencocokan *string*. Algoritma ini adalah penyempurnaan dari algoritma pencocokan *string* dengan menggunakan algoritma *brute force*. Pada algoritma *brute force*, setiap kali ditemukan ketidakcocokan *pattern* dengan teks, maka *pattern* akan digeser satu ke kanan. Sedangkan pada algoritma *Knuth MorrisPratt*, kita memelihara informasi yang digunakan untuk melakukan jumlah pergeseran. Algoritma menggunakan informasi tersebut untuk membuat pergeseran angka lebih jauh, tidak hanya satu karakter seperti pada algoritma *brute force*. Dengan algoritma *Knuth MorrisPratt* ini, waktu pencarian dapat dikurangi secara signifikan. Algoritma *KnuthMorris Pratt* ini dikembangakan oleh D. E. Knuth, bersama-sama dengan J. H. Morris dan V. R. Pratt.

   Algoritma *Knuth MorrisPratt* memelihara informasi yang digunakan saat melakukan pergeseran. Informasi ini digunakan untuk melakukan pergeseran yang lebih jauh, tidak seperti *brute force* yang melakukan pergeseran perkarakter. Pergeseran dilakukan berdasarkan *suffix* kesamaan *suffix* dan *prefix* dalam *pattern* dan yang ditemukan di dalam teks. Dalam algoritma *Knuth MorrisPratt* ini kita akan menemui beberapa definisi yang nantinya akan digunakan dalam algoritma ini.

2. Langkah-Langkah Algoritma *Knuth Morris Pratt*

1) Mencari Fungsi Pinggiran

Algoritma *Knuth MorrisPratt* melakukan proses awal atau *preproccesing* terhadap *pattern* P dengan menghitung fungsi pinggiran (dalam literatur lain menyebut fungsi *overlap*, fungsi *failure*, dsb) yang mengindikasikan pergeseran s terbesar yang mungkin dengan menggunakan perbandingan yang dibentuk sebelum pencarian *string*. Fungsi pinggiran hanya bergantung pada karakter-karakter di dalam *pattern*, dan bukan pada karakter-karekter di dalam teks yang dicari. Oleh karena itu, dapat dilakukan perhitungan fungsi awalan sebelum pencarian *string* dilakukan.

pseucode algoritma *Knuth Morris Pratt* untuk menghitung fungsi pinggiran.

```
begin
 length ←0
   Prefix[0] ←0
   j←0
 for i ←1upto length step 1 do
     while j>0 & P[j+1] ≠ P[i] do
          j ←Prefix[j]
     if P[j+1] = P[i] then
          j ←j+1
     Prefix[i] ←j
          return Prefix
end
```

Keterangan :

Fungsi tersebut akan menghasilkan *output* berupa *arrayinteger* yang merupakan angka-angka pinggiran untuk setiap posisi iterasi pada *pattern*. Barulah kemudian dapat diproses pencocokkan antara *pattern* dan teks yang diberikan.

Langkah-langkah mencari fungsi pinggiran :

(1) Menentukan deklarasi
i, j = *index*
P = *pattern*
b(j) = fungsi pinggiran

(2) Menentukan *pattern* yang dicari, kemudian isi kolom *pattern.*

(3) Buat tabel fungsi pinggiran seperti contoh di bawah ini

Tabel 2.2 Fungsi Pinggiran untuk *Pattern*

i, j	**1**	**2**	**3**	**4**	**5**
P					
b(j)					

Prefix ←

(4) Menambahkan *index* di atas *pattern* untuk memudahkan pergeseran *index* [*i, j*].

(5) Meletakkan posisi [*i*] berada di *index* awal [0].

(6) Meletakkan posisi [*j*] berada di *index* kedua [1].

(7) Untuk nilai awalan *prefix* pasti bernilai 0.

(8) *Pattern* [*i*] dibandingkan dengan *pattern* [*j*], jika *pattern* sama maka *prefix* [*j*] bernilai 1 jadi ([*i*] + 1) dan ([*j*] + 1) namun jika *pattern* [*i*] dan [*j*] tidak sama maka *prefix* [*j*] bernilai 0 dan ([*j*] + 1) dan seterusnya hingga *j* berada di akhir *pattern*.

2) Pencocokan String

Kemudian cara untuk melakukan pencocokan *string* dengan menggunakan algoritma *Knuth MorrisPratt*. Menurut Kourie, Justin et al (2011), secara sistematis langkah-langkah yang dilakukan algoritma *Knuth MorrisPratt* pada saat mencocokkan *string* :

1. Algoritma *Knuth MorrisPratt* mulai mencocokkan *pattern* pada awal teks.
2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
   a. Karakter di *pattern* dan di teks yang dibandingkan tidak cocok (*mismatch*).
   b. Semua karakter di *pattern* cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
   c. Algoritma kemudian menggeser *pattern* berdasarkan tabel, lalu mengulangi langkah 2 sampai *pattern* berada di ujung teks.

   Pencocokan karakter dari kiri ke kanan mencari *Prefix* terpanjang dari P[0..j-1] yang juga merupakan *suffix* dari P [1..j-1], untuk menghindari pergeseran yang tidak perlu. Hasil dari pencarian *prefix* terpanjang disimpan dalam tabel yang disebut juga sebagai *failurefunction*. Misalkan

panjang *String* yang telah diperiksa dan cocok = n dan nilai dari *failurefunction* adalah M, maka dilakukan pergeseran sebanyak (n-m).

Di bawah ini adalah *pseucode* algoritma *Knuth Morris Pratt* untuk pencarian *string*.

```
i ←1, j ←1, k ←1
while n – k ≥ m do
        while j ≤ m & T[i] = P[j] do
                i ← i + 1
                j ←j + 1
        if j > m then output k
        if Prefix[j - 1] > 0 then
                k ← i – Prefix[j - 1]
        else
                if i = k then i ←i + 1
                k ←i
        if j > 1 then j ← Prefix[j - 1] + 1
```

3.2 *Android*

Menurut Meier (2009), *Android* adalah sebuah *softwarestack* bersifat *open source* yang mencakup sistem operasi, *middleware*, dan *key applications* beserta sekumpulan *Application Programming Interface* (API) untuk merancang sebuah aplikasi *mobile* dengan menggunakan bahasa pemrograman *Java*. Aplikasi berbasis *Android* dapat diciptakan, dikembangkan secara bebas serta dapat dengan mudah diunduh dan digunakan sesuai kebutuhan pengguna.

Menurut H., Nazruddin Safaat (2012), dalam bukunya "Pemrograman Aplikasi *MobileSmartphone* dan *Tablet* PC berbasis Android", mengungkapkan bahwa *Android* adalah *mobile* platform pertama yang lengkap, tebuka, dan bebas. Hal ini diuraikan sebagai berikut:

1) Lengkap (*CompletePlatform*)

   Para desainer aplikasi dapat melakukan pendekatan yang komprehensif ketika mereka sedang mengembangkan platform *Android*. *Android* merupakan sistem operasi yang aman dan banyak menyediakan tools dalam membangun *software* dan memungkinkan untuk peluang pengembangan aplikasi.

2) Terbuka (*Open Source Platform*)

   Platform *Android* disediakan melalui lisensi *open-source*. Pengembang dapat dengan bebas untuk mengembangkan aplikasi.

3) Bebas (*Free Platform*)

   Android adalah platform yang bebas untuk para *developer* aplikasi untuk membuat aplikasi mereka sendiri. Tidak ada lisensi atau biaya *royalty* untuk dikembangkan pada platform *Android*. Tidak ada biaya keanggotaan. Tidak diperlukan biaya pengujian. Tidak ada kontrak yang diperlukan. Aplikasi untuk android dapat didistribusikan dan diperdagangkan dalam bentuk apapun.

3.3 Java

Menurut Supriyatno (2010), *Java* merupakan sebuah bahasa pemrograman berorientasi objek yang dapat berjalan pada *platform* yang berbeda, baik di *Windows*, *Linux*, serta sistem operasi lainnya.

Menurut Mulyana (2008), Java merupakan bahasa pemrograman yang berorientasi objek (OOP - *Oject Oriented Programming*). Dalam bahasa OOP seperti *Java*, objek (*object*) merupakan *entitas* fundamental yang secara efektif dapat digunakan untuk merepresentasikan entitas nyata.

Jadi, kita dapat membuat sebuah aplikasi dengan java pada sistemoperasi *linux* dan selanjutnya menjalankan atau menginstal aplikasi tersebut pada sistemoperasi *Windows* dan juga sebaliknya tanpa mengalami masalah. Dengan menggunakan *Java*, kita dapat mengembangkan banyak aplikasi yang dapat digunakan pada lingkungan yang berbeda, seperti pada: *Desktop, Mobile, Internet,* dan lain-lain.

3.4 Android Studio

Android Studio adalah sebuah IDE untuk *Android Development* yang diperkenalkan google pada acara Google I/O 2013. *Android Studio* merupakan pengembangkan dari Eclipse IDE, dan dibuat berdasarkan IDE *Java populer*, yaitu IntelliJ IDEA. *Android Studio* merupakan IDE resmi untuk pengembangan aplikasi *Android*.

Sebagai pengembangan dari Eclipse, *Android Studio* mempunyai banyak fitur-fitur baru dibandingkan dengan Eclipse IDE. Berbeda dengan Eclipse yang menggunakan *Ant*, *AndroidStudio* menggunakan *Gradle* sebagai *buildenvironment*. Fitur-fitur lainnya adalah sebagai berikut :

a. Menggunakan *Gradle-based build system* yang fleksibel.
b. Bisa mem-*build multiple* APK .
c. *Templatesupport* untuk *GoogleServices* dan berbagai macam tipe perangkat.
d. Layout editor yang lebih bagus.

e. *Built-in support* untuk Google *Cloud Platform*, sehingga mudah untuk integrasi dengan Google *Cloud Messaging* dan *App Engine*.
f. *Import library* langsung dari *Maven repository*
g. dan masih banyak lagi lainnya

Ketika awal saya berkenalan dengan *Android*, saya menggunakan Eclipse sebagai IDE untuk membuat aplikasi *Android*. Jika dibandingkan dengan *AndroidStudio* memang dari sisi *build* lebih baik dibandingkan Eclipse, *karena Android Studio* menggunakan *Gradle*. Ditambah lagi berbeda dengan Eclipse, kita tidak perlu lagi dipusingkan dengan dependencies package pada *Android Studio*. Satu hal tambahan lagi yang membuat *Android Studio* unggul adalah dukungan *layoutxmleditor* secara *visual* yang jauh lebih baik daripada Eclipse. Walaupun begitu, *Android Studio* saat ini masih dalam tahap beta dan belum mempunyai dukungan untuk NDK/*Native DevelopmentKit*.

1) *Android Studio* Sebagai *CodeEditor* Cerdas
Inti dari *Android Studio* adalah *editor* kode cerdas mampu *code completion* dengan cerdas, *refactoring* dan analisis kode. *Editor* yang baik membantu Anda menjadi pengembang aplikasi *Android* lebih produktif.
2) Tersedia *Template* dan bisa Integrasi *GitHub*
New Project Wizards membuatnya lebih mudah dari sebelumnya untuk memulai sebuah *project* baru. Memulai *project* menggunakan kode *template* untuk pola seperti navigasi laci dan melihat halaman dan bahkan mengimpor contoh kode Google dari *GitHub*.
3) *Multi-Screen App Development*
Membangun aplikasi untuk ponsel *Android*, *tablet*, *Android Wear*, *Android TV*, *Android Auto* dan Google *Glass*. Dengan *Project Android View* baru dan model dukungan di *Android Studio*, lebih mudah untuk mengelola *project* aplikasi dan sumber daya.
4) *Virtual Devices For All Shapes And Sizes*
Android Studio hadir pra-dikonfigurasi dengan *emulator* gambar yang optimal. Diperbarui dan efisien *Virtual Device Manager* memberikan profil perangkat yang telah ditetapkan untuk perangkat *Android* umum.
5) *Android Builds Evolved, With Gradle*
Membuat beberapa APK untuk aplikasi *Android* Anda dengan fitur yang berbeda menggunakan *project* yang sama. Mengelola aplikasi dependesi dengan *Maven*. Membangun APK dari *Android Studio* atau *command line*.
6) *Package Explorer*
Package Explorer, merupakan jendela yang menampilkan *project-project* yang sudah kita buat dalam sebuah *workspace*. Jendela ini layaknya *explore* pada sistem operasi *windows*, yang berguna untuk mengexplore *project* yang ada dalam sebuah *workspace.*
7) *JavaEditor*
Java Editor, merupakan jendela yang menampilkan *coding java* yang sedang kita kerjakan. Dari sini kita mengetikkan perintah-perintah dalam *syntaxjava* untuk membuat sebuah aplikasi.
8) *Console*
Console, merupakan jendela yang menampilkan hasil *output* dari aplikasi yang kita buat. Agar hasil *output*nya mau berjalan, tekan *RUN* pada *toolbar*. Jika yang muncul *error*, berarti aplikasi yang kita buat memang memiliki *error*, dan *console* ini akan menunjukkan dimana letak *error* nya.
9) *Toolbar*
Toolbar, berisi *shortcut* perintah dalam bentuk *icon*. Seperti *create new* (*project, folder*), *create new visual classes* (GUI *withSwing*), *Save, Run, Createnew package*, *Create new class* dan masih banyak lagi

3.3 Tahapan Pengembangan Mutimedia

Pada aplikasi ini penulis menggunakan Metode yang digunakan dalam pengembangan sistem adalah metode *Extreme Programming* (XP). *Extreme Programming* (XP) adalah implementasi model *Software* dari *Rapid Application Development (RAD)* dan merupakan salah satu metode pengembangan *software* yang termasuk dalam *Agile Software Development*. XP menggunakan pendekatan *object-oriented*.

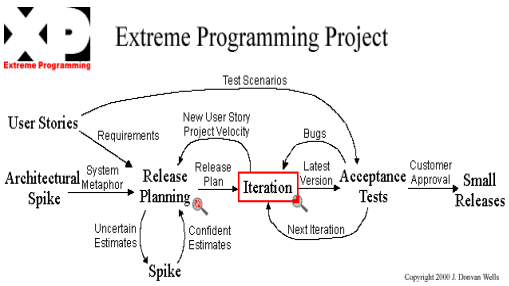
Dalam XP, terdapat 5 nilai yang menjadi pondasi yaitu *communication, simplicity, feedback, courage,* dan *respect.* Komunikasi yang efektif antara pengembang perangkat lunak dan pihak-pihak yang terlibat sangatlah penting. Dalam XP, desain dijadikan kebutuhan *intermediate*. Desain dibuat sederhana mungkin agar mudah mengimplementasikan *code*. Disini dapat terjadi perubahan struktur desain atau perubahan *sourcecode* tanpa mengubah fungsi utamanya *(refactoring).Feedback* akan diberikan saat peningkatan dan pengimplementasian perangkat lunak.

Berikut merupakan proses *Extreme Programming* menurut Pressman (2010):

2. *Planning.* Tahap *planning* dimulai dengan membuat *user stories* yang menggambarkan *output*, fitur, dan fungsi-fungsi dari *software*

yang akan dibuat. *Userstories* tersebut kemudian diberikan bobot seperti prioritas dan dikelompokkan untuk selanjutnya dilakukan proses *delivery* secara *incremental*.

3. *Design.Design* di *ExtremeProgramming* mengikuti prinsip *Keep It Simple* (KIS). Untuk *design* yang sulit, *Extreme Programming* akan menggunaan *Spike Solution* dimana pembuatan design dibuat langsung ke tujuannya. *Extreme Programming* juga mendukung adanya *refactoring* dimana *softwaresystem* diubah sedemikian rupa dengan cara mengubah stuktur kode dan menyederhanakannya namun hasil dari kode tidak berubah.
4. *Coding*. Proses coding pada XP diawali dengan membangun serangkaian unit test. Setelah itu pengembang akan berfokus untuk mengimplementasikannya. Dalam *Extreme Programming* diperkenalkan istilah *Pair Programming* dimana proses penulisan program dilakukan secara berpasangan. Dua orang *programmer* saling bekerjasama di satu komputer untuk menulis program. Dengan melakukan ini akan didapat *real-time problem solving* dan *real-time quality assurance.*
5. *Testing.* Tahap ini dilakukan pengujian kode pada unit *test*. Dalam *Extreme Programming*, diperkenalkan XP *acceptancetest* atau biasa disebut *customertest*. Tes ini dilakukan oleh customer yang berfokus kepada fitur dan fungsi sistem secara keseluruhan. *Acceptance test* ini berasal dari user stories yang telah diimplementasikan.



Gambar 2.3 *Extreme Programming Project*

3.4 Alat Bantu Perancangan Aplikasi

UML merupakan singkatan dari *"Unified Modelling Language"* yaitu suatu metode permodelan secara *visual* untuk sarana perancangan sistem berorientasi objek, atau definisi UML (*Unified Modelling Language*) yaitu sebagai suatu bahasa yang sudah menjadi standar pada visualisasi, perancangan dan juga pendokumentasian sistem *software*. Saat ini UML sudah menjadi bahasa standar dalam penulisan *blue printsoftware*.

Tujuan atau Fungsi dari Penggunaan UML

1) Dapat memberikan bahasa permodelan *visual* kepada pengguna dari berbagai macam pemrograman maupun proses rekayasa.
2) Dapat menyatukan praktek-praktek terbaik yang ada dalam permodelan.
3) Dapat memberikan model yang siap untuk digunakan, merupakan bahasa permodelan *visual* yang ekspresif untuk mengembangkan sistem dan untuk saling menukar model secara mudah.
4) Dapat berguna sebagai *blue print*, sebab sangat lengkap dan detail dalam perancangannya yang nantinya akan diketahui informasi yang detail mengenai koding suatu program.
5) Dapat memodelkan sistem yang berkonsep berorientasi objek, jadi tidak hanya digunakan untuk memodelkan perangkat lunak *(software) saja*.
6) Dapat menciptakan suatu bahasa permodelan yang nantinya dapat dipergunakan oleh manusia maupun oleh mesin.

Jenis diagram UML

1) *Usecase* Diagram

   Use case diagram yaitu salah satu jenis diagram pada UML yang menggambarkan interaksi antara sistem dan aktor, *use case* diagram juga dapat men-deskripsikan tipe interaksi antara si pemakai sistem dengan sistemnya. Di bawah ini deskripsi *Use Case*

   (a) Diagram *use case* merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat.

   (b) Diagram *use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat.

   (c) Diagram *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Yang ditekankan pada diagram ini adalah "apa" yang diperbuat sistem, dan bukan "bagaimana"

(d) Sebuah *use case* merepresentasikan sebuah interaksi antara aktor (*user* atau sistem lainya) dengan sistem.

(e) *Use case* menjelaskan secara sederhana fungsi sistem dari sudut pandang *user*.

2) *Activity* Diagram

Activity diagram atau diagram aktivitas yaitu salah satu jenis diagram pada UML yang dapat memodelkan proses-proses apa saja yang terjadi pada sistem.

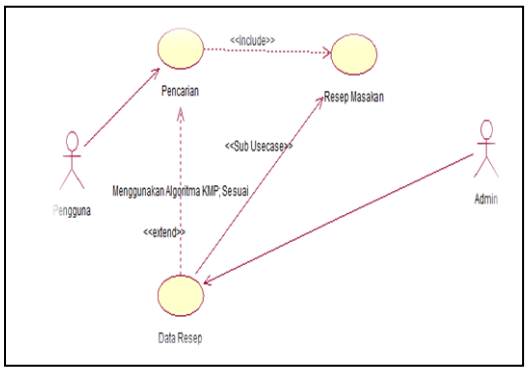
3) *Sequence* Diagram

Sequence diagram yaitu salah satu jenis diagram pada UML yang menjelaskan interaksi objek yang berdasarkan urutan waktu, *sequence* diagram juga dapat menggambarkan urutan atau tahapan yang harus dilakukan untuk dapat menghasilkan sesuatu seperti pada *use case* diagram.

4 RANCANGAN SISTEM

Desain ini merupakan proses untuk membuat atau menciptakan obyek baru. Dalam sub bab ini akan membahas mengenai desain visualisasi, rancangan dan dokumentasi sistem dengan menggunakan alat pengembangan sistem *Unified Modelling Language* (UML), berikut desain yang digunakan oleh peneliti dengan 3 metode, yaitu *Use Case* Diagram, *Activity* Diagramdan *Squence* Diagram.

1. *Use Case* Diagram

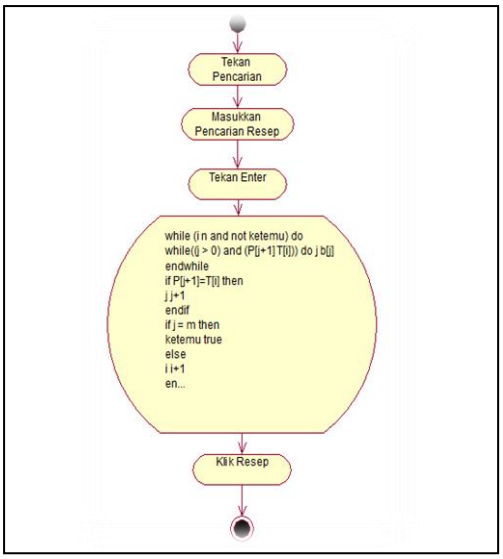


Gambar 4.1 *Use Case Diagram*

Berdasarkan gambar 4.1 pengguna dapat melakukan pencarian dengan mengklik pencarian kemudian algoritma akan mencari resep yang di cari. Pencarian resep bergantung pada data resep yang di buat oleh admin agar dapat menemukan resep yang dicari oleh pengguna.

2. *Activity* Diagram

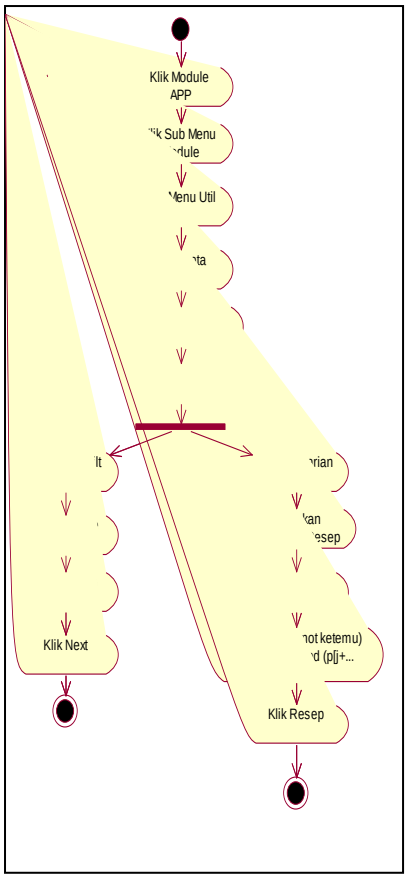
Pada gambar 4.2 *Activity* diagram pencarian inilebih memfokuskan diri pada eksekusi dan alur sistem daripada bagaimana sistem itu dirancang. *Activity* diagrammenunjukkan aktivitas sistem dalam bentuk kumpulan aksi-aksi. Pada aktivitas ketika tekan *icon* pencarian maka aktivitas aplikasi ini akan mencari kata kunci apa yang *user* ketik di dalam kolom pencarian maka pencarian akan mencari data koleksi masakan khas Bali menggunakan algoritma *Knuth Morris Pratt*.



Gambar 4.2 *Activity* DiagramPencarian

Pada gambar 4.3 A*ctivity* diagram admin di bawah ini menggambarkan kejadian atau aktivitas yang terjadi oleh Admin. Dimana dimulai dengan memilih module app di aplikasi *Android Studio* kemudian sub menu *module Util* kemudian pilih file *java* yaitu data.*java* kemudian inputkan resep masakan dan kemudian *save*.

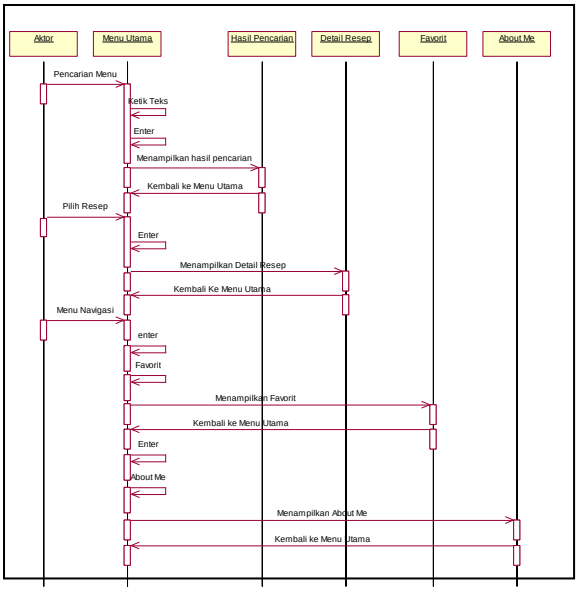
Untuk pengujian apakah *database* tersebut masuk ke dalam sistem yang dibuat yaitu dengan membuat apk pada aplikasi tersebut menggunakan *toolsBuilt* kemudian klik *GenerateSigned* APK kemudian isi biodata *Keystore* dan klik *next* maka aplikasi *Android Studio* akan membangun aplikasi yang sudah dibuat tadi menjadi *format* .APK, disini *database* diuji dengan melakukan pencarian resep masakan melalui aplikasi yang sudah dibuat dengan aktivitas yaitu klik pencarian kemudian inputkan resep yang akan dicari tekan enter kemudian algoritma akan mencari resep masakan apabila ditemukan akan terlihat menu yang dicari kemudian klik resep untuk melhat detail resep masakan



Gambar 4.3 *Activity* Diagram Admin

3. *Squence Diagram*

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh objek dan pencarian yang diletakkan diantara objek-objek ini di dalam *usecase*. Adapun *Sequence diagram* yang digunakan pada aplikasi yang akan dibangun yaitu sebagai berikut :



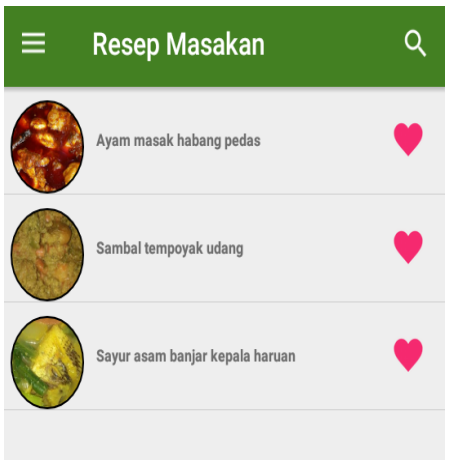
Gambar 4.7 *Squence* Diagram

Pada saat *Actor* masuk ke dalam menu utama aplikasi, ada *icon* pencarian resep menu masakan, *actor* mengetik teks di dalam kotak dialog pencarian tersebut lalu tekan enter dan menampilkan hasil pencarian apa yang di ketik oleh *Actor* lalu kembali ke menu utama. *Actor* memilih salah satu resep masakan dan tekan enter lalu menampilkan detail resep masakan tersebut, *Actor* kembali lagi ke menu utama dan klik menu navigasi tekan menu favorite lalu akan tampil menu favorite kemudian kembali lagi ke menu utama. Ada menu *about me* kemudian tekan enter lagi akan tampil detail *about me* dan *Actor* kembali lagi ke menu utama.

4 IMPLEMENTASI

4.1 Tampilan Menu Home

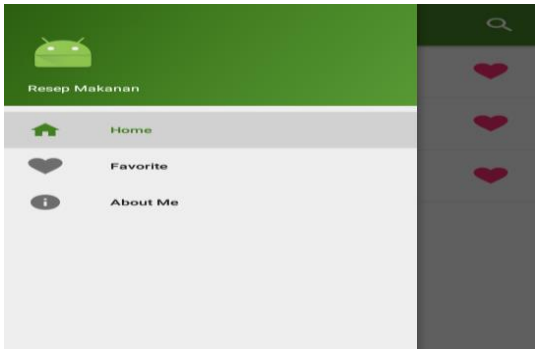
Adapun tampilan *form* menu utama yang terdapat pilihan menu utama, menu navigasi dan *icon*pencarian untuk mencari resep masakan yang di pilih *user*. *User* bias menyukai/menyimpan resep masakan tersebut di dalam menu utama dengan klik *icon* warna merah muda. Adapun tampilan *form* menu utama yang dapat dilihat pada gambar 4.8.



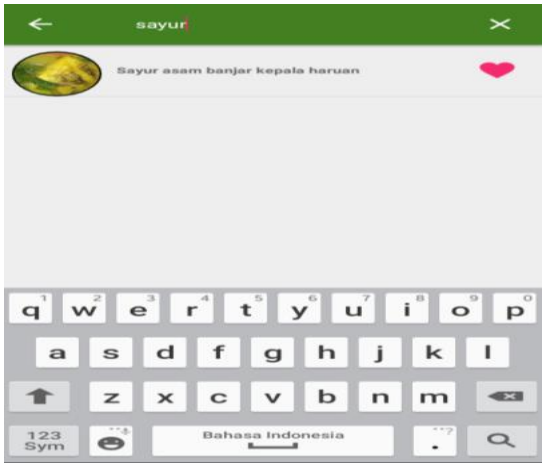
Gambar 4.8 Halaman Menu Utama

4.2 Tampilan Menu Materi

Pada Pada tampilan halaman navigasi ini seperti gambar di bawah ini terdapat 3 (tiga) menu yaitu menu Home, menu Favorite, dan menu About Me. Ketika *user* klik menu Home maka akan muncul semua resep masakan khas Bali kemudian ketika *user* klik menu Favorite maka akan menampilkan resep masakan sesuai masakan favorite *user*.



4.3 Pada gambar 4.10 di bawah ini menunjukkan halaman pencarian pada aplikasi ini menggunakan algoritma *Knuth Morris Pratt*. *Output* nya muncul sesuai dengan judul resep masakan yang di *input*kan oleh Admin.



5 KESIMPULAN

Berdasarkan Bedasarkan uraian pada bab-bab sebelumnya maupun pembahasan yang telah dikemukakan maka dapat ditarik kesimpulan sebagai berikut:

1. Algoritma *Knuth Morris Pratt* dapat di implementasikan ke dalam aplikasi pencarian koleksi resep masakan khas Bali berbasis *Android*.
2. Aplikasi ini dapat membantu para pengguna dalam mencoba memasak salah satu resep masakan khas Bali.
3. Dengan adanya pengujian *White Box* dan *Black Box* maka aplikasi ini sudah berjalan dengan baik karena *White Box* digunakan untuk pengujian *source code* jika terdapat kesalahan *input* dari *user* akan tampil pemberitahuan bahwa data yang dicari tidak ada, sedangkan *Black Box* digunakan untuk pengujian *button* pada aplikasi sehingga menampilkan apa yang di inginkan *user* dengan benar.

6 SARAN

Berdasarkan dari kesimpulan yang telah dikemukakan diatas, maka saran-saran yang dapat diberikan adalah sebagai berikut :

1. Aplikasi ini masih sangat sederhana dan dapat ditingkatkan menjadi aplikasi yang lebih baik lagi.
2. Adanya simulasi untuk pencarian *string* menggunakan algoritma *Knuth Morris Pratt* ini.
3. Dalam aplikasi pencarian koleksi resep masakan khas Bali berbasis *Android* ini dapat ditambahkan menu *input*, agar *user* dapat menambahkan resep ke dalam aplikasi.
4. Adanya aplikasi perbandingan antara algoritma *Knuth Morris Pratt* dengan algoritma yang lainnya.
5. Menambahkan *icon share* untuk *user* bisa membagikan resep masakan khas Bali ke media sosial.

Meningkatkan aplikasi ini berbentuk *online* agar *user* bisa membuat akun dan menyimpan resep masakan khas Bali ke dalam akun tersebut

7 DAFTAR PUSTAKA

Abi Mahan Zaky, 2015. *Implementasi Algoritma Knuth Morris Pratt pada Perancangan Game Hanacaraka*. http://lib.unnes.ac.id/20733/. Diakses 26 November 2016.

Anhar, ST. 2010, *Panduan Menguasai PHP & MySQL Secara Otodidak.* Jakarta: Mediakita.

Bima Laksamana, 2013. *Implementasi Algoritma Knuth Morris Pratt pada Alat Penerjemah Suara*. http://elib.unikom.ac.id/download.php?id=270320. Diakses 26 November 2016

Dhanta, R, 2009, *Pengantar Ilmu Kompter*, Surabaya, INDAH.

Hermawan, S. Stephanus, 2011, *Mudah Membuat Aplikasi Android*. Yogyakarta: Andi Offset.

Kadir, Abdul, 2008, *Belajar Database Menggunakan MySQL*. Yogyakarta: Andi Offset.

Meier, Reto. (2009), *Professional Android™ Application Development*, Wiley Publishing, Inc., Indianapolis, Indiana.

Mulyana. Dr. Ing. Eueung, 2008, *Belajar Java Secara Visual dan Interaktif*. Bandung: Andi Offset.

Nazruddin Safaat H. 2012 (Edisi Revisi). *Pemograman Aplikasi Mobile*

Rizky, Soetam, 2011. *Konsep Dasar Rekayasa Perangkat Lunak*. Prestasi Pustaka. Jakarta.

Sidik. Betha, 2014, *Pemrograman Web dengan PHP*. Bandung: Informatika Bandung.

Sirenden, Bernadus Herdi dan Laekha, Dachi Ester. 2012, *Buat Sendiri Aplikasi Petamu Menggunakan CodeIgniter dan Google Maps API.* Jakarta: Andi Offset.

Sommerville, Ian. 2011 (Dalam Pengembangan Eka Arriyanti). *Software Engineering (Rekayasa Perangkat Lunak)*. Jakarta: Erlangga

Supriyatno, 2010, *Menggunakan Java dan MySQL Untuk Pemula*. Jakarta: Mediakita.

Lestari, Umi Tri. 2014. Pembuatan Aplikasi Kumpulan Resep Masakan Jawa Tradisional Berbasis Android, *http://repositry.amikom.ac.id/files/Publikasi_10.11.4441.pdf.* Diakses 01 April 2016.

Wibowo, T., Wibowo, A. & Sari R.P. 2012. *Pembuatan Aplikasi Untuk Mendeteksi Kebenaran Perintah Sql Query Menggunakan Metode Knuth-*