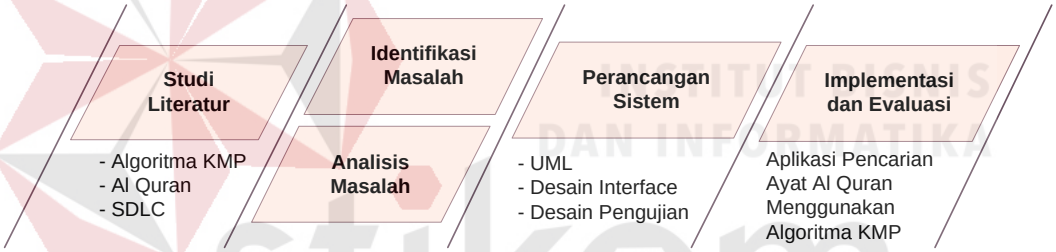
BAB III

METODE PENILITIAN

Pada penelitian ini dilakukan beberapa tahapan pengembangan sistem yang mengacu pada *System Development Life Cycle* (SDLC) yakni studi literatur, identifikasi masalah, analisis permasalahan, perancangan sistem, serta implementasi dan evaluasi. Adapun desain penelitian pada Rancang Bangun Aplikasi Pencarian Ayat Al-Qur'an Menggunakan Metode Algoritma *String Matching* Knuth-Morris-Pratt berbasis android dapat dilihat pada Gambar 3.1.



Gambar 3.1 Desain Penelitian

3.1 Studi Literatur

Studi literatur atau kajian pustaka merupakan penelusuran literatur yang bersumber dari buku, media, pakar, ataupun hasil dari penelitian orang lain yang bertujuan untuk menyusun dasar teori yang digunakan dalam melakukan penelitian. Adapun landasan teori yang digunakan penulis dapat dilihat pada Bab II Landasan Teori.

3.2 Identifikasi Permasalahan

Al-Qur'an merupakan kitab suci umat Islam yang banyak memuat ilmu pengetahuan yang banyak bermanfaat bagi manusia. Isi kandungan dari Al-Qur'an mengandung hukum-hukum, kisah, akhlak, akidah, dan asas perilaku yang dapat menuntun manusia menjadi lebih baik.

Maka dari itu, sangat perlu untuk mengetahui isi kandungan dari Al-Qur'an, karena merupakan pentunjuk bagi umat manusia. Hanya saja terdapat kesulitan dalam mencari ayat Al-Qur'an yang sesuai dengan topik yang ingin dicari. Salah satu faktor tersebut yaitu terdapat berbagai macam topik dalam satu surat dan persamaan topik pada surat yang berbeda. Sedangkan jumlah ayat dan surat pada Al-Qur'an sangatlah banyak.

Ada beberapa cara untuk melakukan pencarian ayat Al-Qur'an secara manual yakni dengan menggunakan buku index Al-Qur'an, Al-Qur'an cetakan khusus yang dilengkapi dengan index tematik, atau dengan menggunakan kitab *fathurrahman* yang membutuhkan pengetahuan lebih dalam tata bahasa Arab, hal ini tentunya akan memakan banyak waktu dalam melakukan pencarian ayat Al-Qur'an sesuai topik yang ingin dicari. Sedangkan aplikasi *smartphone*, seperti aplikasi Al-Qur'an, tafsir Al-Qur'an, terjemahan Al-Qur'an dan yang lainnya sedikit yang menyediakan fasilitas untuk mencari ayat Al-Qur'an sesuai topik yang ingin dicari.

Dari permasalahan diatas tujuan dari penelitian ini adalah menghasilkan suatu aplikasi sebagai alat bantu dalam proses pencarian ayat Al-Qur'an sesuai topik yang dicari.

3.3 Analisis Masalah

Dari permasalahan diatas maka dilakukan analisis, analisis dilakukan terhadap kebutuhan aplikasi yang akan dibangun, agar pemanfaatan aplikasi memperoleh hasil yang optimal.

Dalam penelitian ini aplikasi yang dikembangkan yaitu aplikasi berbasis android. Pemilihan sistem operasi android yaitu selain bersifat open source yang memudahkan dalam melakukan pengembangan, juga pengguna smartphone android di Indonesia yang terus meningkat setiap tahunnya. Menurut data yang dilansir International Data Corporation (IDC), untuk Q2 2012, android sendiri menguasai sekitar 52% dari total sistem operasi smartphone yang dipakai di Indonesia, Selain itu harga sebuah smartphone android yang terjangkau.

Pemilihan Algoritma Knuth-Morris-Pratt karena Algoritma Knuth-Morris-Pratt sendiri bekerja dengan cara melakukan pergeseran yang lebih sedikit dalam pencocokan string, berbeda dengan algoritma Brute Force yang melakukan pencocokan string dengan pergeseran satu per satu karakter. Selain itu, pemilihan algoritma Knuth-Morris-Pratt pada penelitian ini yaitu algoritma Knuth-Morris-Pratt sangat efektif dalam melakukan pencarian satu *pattern* atau pola kata.

Untuk itu penulis akan membangun aplikasi pencarian ayat Al-Qur'an menggunakan algoritma *string matching* Knuth-Morris-Pratt berbasis android. Untuk menyediakan fitur Pencarian ayat Al-Qur'an yang dapat mengatasi masalah yang lama, maka aplikasi ini harus menyediakan fitur yang mencakup :

1. Dapat menampilkan ayat Al-Qur'an secara digital.
2. Dapat menampilkan terjemahan sesuai dengan topik yang dicari.

Aplikasi pencarian ayat Al-Qur'an ini akan dapat membantu dan mempermudah pengguna dalam proses pencarian ayat-ayat suci Al-Qur'an secara mendalam bagi siapa saja yang membutuhkan.

3.3.1 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem merupakan deskripsi lengkap tentang perilaku sistem dengan menggambarkan interaksi pengguna dengan perangkat lunak. Analisis kebutuhan ini dibagi menjadi dua yaitu kebutuhan fungsional dan kebutuhan non fungsional. Kebutuhan fungsional adalah jenis kebutuhan yang berisikan proses-proses apa saja yang di berikan oleh aplikasi yang akan dibangun. Sedangkan kebutuhan non fungsional mendefinisikan tentang properti sistem dan batasan sistem yang berkaitan dengan kebutuhan fungsional agar dapat berfungsi dengan semestinya. Tabel 3.1 menunjukan kebutuhan sistem yang berkaitan dengan fungsi aplikasi yang akan dibangun.

Tabel 3.1 Kebutuhan Fungsional

No.	Kebutuhan Fungsional	Keterangan
1.	Pencarian ayat Al-Qur'an	Aplikasi mampu melakukan pencarian ayat Al-Qur'an sesuai dengan kata kunci yang dimasukan oleh pengguna dengan menggunakan algoritma *string matching* Knuth-Morris-Pratt.
2.	Menampilkan detail ayat Al-Qur'an	Aplikasi dapat menampilkan ayat Al-Qur'an beserta terjemahannya sesuai dengan kata kunci yang dimasukan oleh pengguna.

Sedangkan kebutuhan non fungsional yang harus dimiliki untuk mengukur kualitas dari aplikasi pencarian ayat Al-Qur'an ini ditunjukan pada tabel 3.2.

Tabel 3.2 Kebutuhan Non Fungsional

No.	Karakteristik	Keterangan
1	*Funcitionality*	Aplikasi dapat menjalankan fungsinya dengan baik.
2	*Performance*	Aplikasi memiliki kinerja yang baik dalam menjalankan fungsinya.
3	*Usability*	Aplikasi memiliki tampilan yang user-friendly
4	*Compatibility*	Aplikasi dapat berjalan di beberapa versi android.

Selain mengukur kualitas dari aplikasi analisis kebutuhan non fungsional juga bertujuan untuk mengetahui spesifikasi minimum yang di terapkan, perangkat keras dan perangkat lunak apa saja yang dibutuhkan serta siapa yang akan menggunakan sistem ini.

1. Kebutuhan Perangkat Keras

Kebutuhan perangkat keras yang diperlukan untuk membangun aplikasi pencarian ayat Al-Qur'an menggunakan algoritma *string matching* Knuth-Morris-Pratt berbasis android ini ditunjukan pada Tabel 3.3.

Tabel 3.3 Kebutuhan Perangkat Keras Perancangan

No	Perangkat Keras	Spesifikasi
1	Processor	Inter(R) Core(TM) 2 Quad Q8400 2.6GHz
2	LCD	LG 18.5”
3	VGA	1 GB
4	RAM	2 GB
5	Hardisk	500 GB

Sedangkan kebutuhkan perangkat untuk implementasi yaitu dengan menggunakan android emulator , smartphone android, atau aplikasi bluestack.

2. Kebutuhan Perangkat Lunak

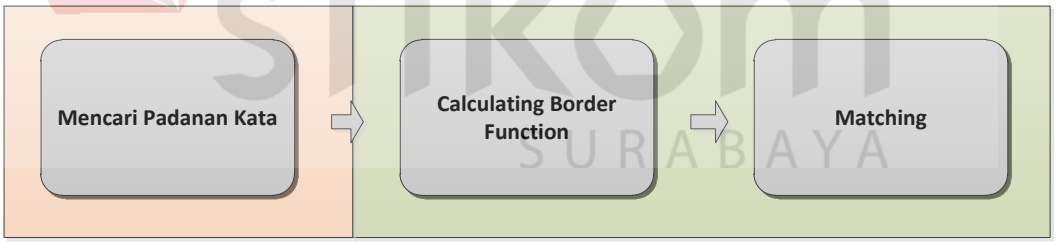
Perangkat lunak yang digunakan untuk merancang aplikasi ini adalah :

Tabel 3.4 Kebutuhan Perangkat Lunak Perancangan

No	Perangkat Lunak
1	Java Development Kit (JDK)
2	IDE Eclipse
3	Android SDK
4	Android Development Tools
5	IBM Rational Rose Enterprise
6	Enterprise Architech
7	WireframeSketcher

3.3.2 Blok Diagram Aplikasi Pencarian Ayat Al-Qur'an

Pembuatan aplikasi perncarian ayat Al-Qur'an menggunakan *database* sebagai media penyimpanan ayat Al-Qur'an dan terjemahannya beserta daftak kata sinonim. Untuk proses pencarian dilakukan dengan cara mencocokan kata kunci yang diinputkan oleh *user* dengan terjemahan Al-Qur'an yang ada pada *database* dengan menggunakan algoritma Knuth-Morris-Pratt, algoritma Knuth-Morris-Pratt sendini menggunakan *preprocessing* yaitu *calculating border function* dan proses *matching*. Sedangkan proses mencari padanan kata yaitu mencari kata yang memiliki arti sama dengan kata kunci yang dinputkan oleh *user,* sebagai contoh kata "shalat", "sholat", dan "solat", ketiganya memiliki arti yang sama. Desain blok diagram untuk Rancang Bangun Aplikasi Pencarian Ayat Al-Qur'an Menggunakan Algoritma *String Matching* Knuth-Morris-Pratt Berbasis Android dapat dilihat pada gambar 3.2.



Gambar 3.2 Blok Diagram Gambaran Umum Aplikasi

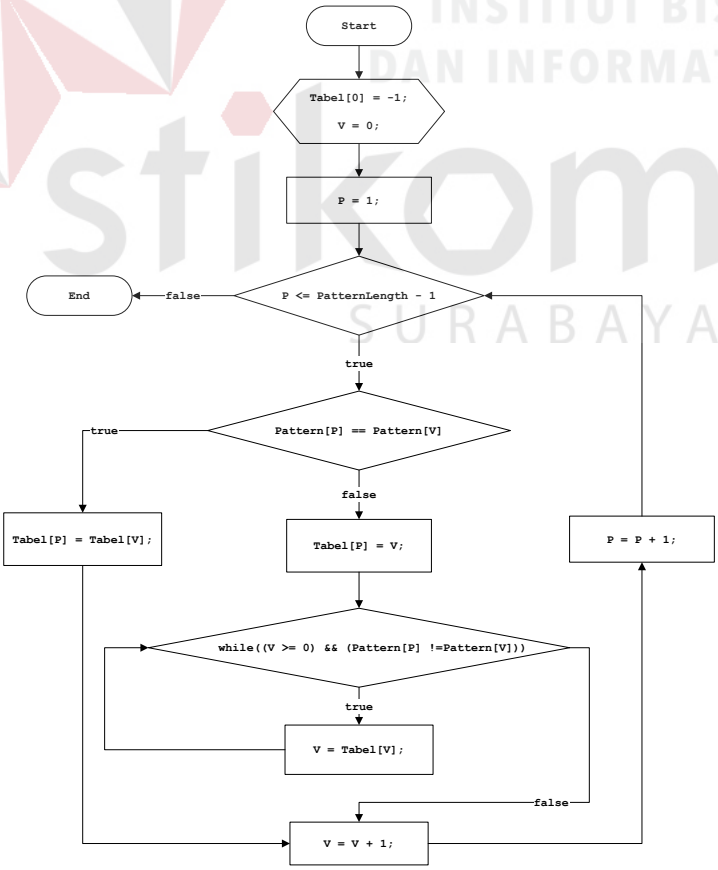
a. Mencari Padanan Kata

Mencari padanan kata adalah proses awal pada percarian ayat Al-Qur'an, proses ini bukan merupakan bagian dari algoritma Knuth-Morris-Pratt. Pada proses ini sistem akan mencari daftar padanan kata yang ada pada *database* kemudian mencocokannya dengan kata kunci yang diinputkan oleh *user*. Jika kata kunci yang diinput ada pada daftar padanan kata, maka akan dilakukan pencarian

pada semua padanan kata yang sesuai dengan kata kunci. Sebagai contoh, jika user menginputkan kata kunci “sholat” dan daftar padanan kata adalah “sholat”, “shalat”, dan “solat”, maka akan dilakukan pencarian dengan tiga kata padanan tersebut satu per satu. Dan jika kata kunci yang diinput tidak memiliki padanan kata, maka sistem akan langsung melakukan proses selanjutnya, yaitu menghitung nilai pinggiran.

b. *Calculating Border Function*

Tahap kedua dari proses pencarian ayat Al-Qur’an yaitu *Calculating Border Function* atau menghitung nilai pinggiran dari *pattern* atau kata kunci yang telah di inputkan oleh user. Gambar 3.3 adalah *flowchart* untuk proses menghitung nilai pinggiran.



Gambar 3.3 *flowchart* Menghitung Nilai Pinggiran

Sebagai contoh diberikan *pattern* dengan kata dasar “agama”, langkah pertama dari menghitung nilai pinggiran yaitu dengan memberikan nilai pinggiran -1 pada index ke-0 dan inisialisasi posisi *V* pada index ke-0 seperti yang ditunjukan pada gambar 3.4.

Cycles						
Pattern		a	g	a	m	a
Table		-1				
Index	-1	0	1	2	3	4

V

Gambar 3.4 Inisialisasi Nilai Pinggiran index ke-0 dan posisi *V*

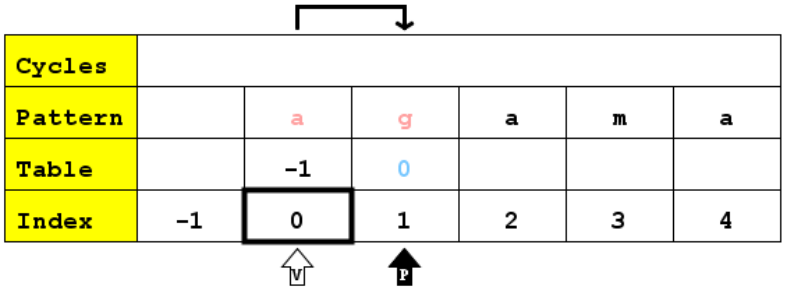
Proses selanjutnya yaitu menentukan posisi *P* pada index ke-1 untuk memulai pencocokan karakter pada tabel pattern, pencocokan karakter dilakukan apabila *P* <= *panjang pattern* – 1, apabila kondisi tersebut *true* maka akan dilakukan perbandingan antara karakter pada *pattern[V]* dan *pattern[P]* dan jika *false* maka perhitungan nilai pinggiran selesai. Gambar 3.6 menunjukan karakter *pattern[V]* tidak sama dengan karakter *pattern[P]*.

Cycles						
Pattern		a	g	a	m	a
Table		-1				
Index	-1	0	1	2	3	4

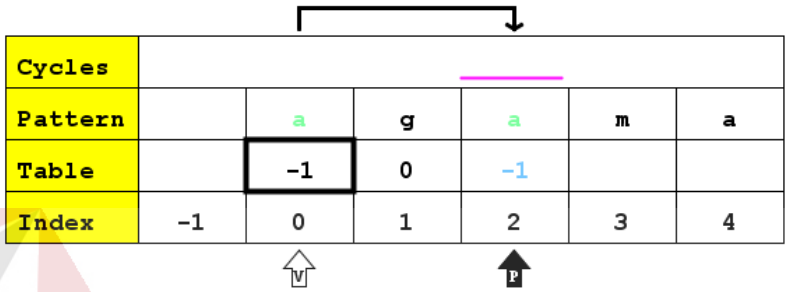
V P

Gambar 3.5 Inisialisasi Posisi *P*

Jika karakter pada *pattern[V]* dan *pattern[P]* tidak sama maka nilai pinggiran dari *pattern[P]* = *V* dan jika sama maka nilai pinggiran dari *pattern[P]* = nilai pinggiran *pattern[V]*.

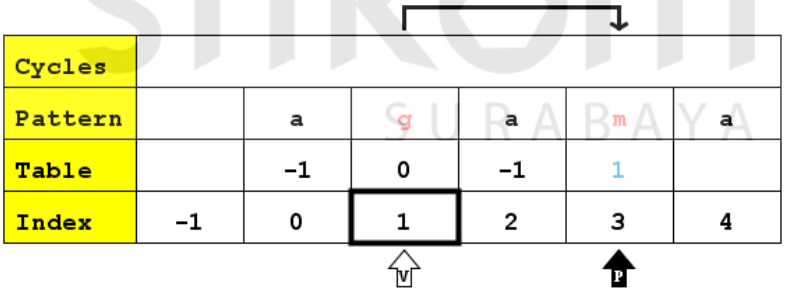


Gambar 3.6 *Pattern[V]* dan *Pattern[P]* tidak sama



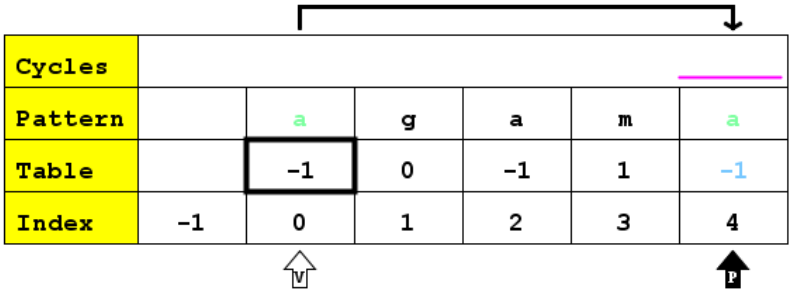
Gambar 3.7 *Pattern[V]* dan *Pattern[P]* Sama

Apabila terjadi kecocokan karakter antara *pattern[V]* dan *pattern[P]* maka pencarian akan dilakukan lagi pada *pattern[V]+1* dan *pattern[P]+1* seperti yang ditunjukan pada gambar 3.8.



Gambar 3.8 Posisi *Pattern[V]* dan *Pattern[P]* Setelah Terjadi Kecocokan

Jika karakter pada *pattern[V]* dan *pattern[P]* tidak sama maka nilai pinggiran dari *pattern[P] = V*, kemudian pencocokan kembali dilakukan dengan posisi *V = 0* dan posisi *P+1* seperti yang ditunjukan pada gambar 3.9.

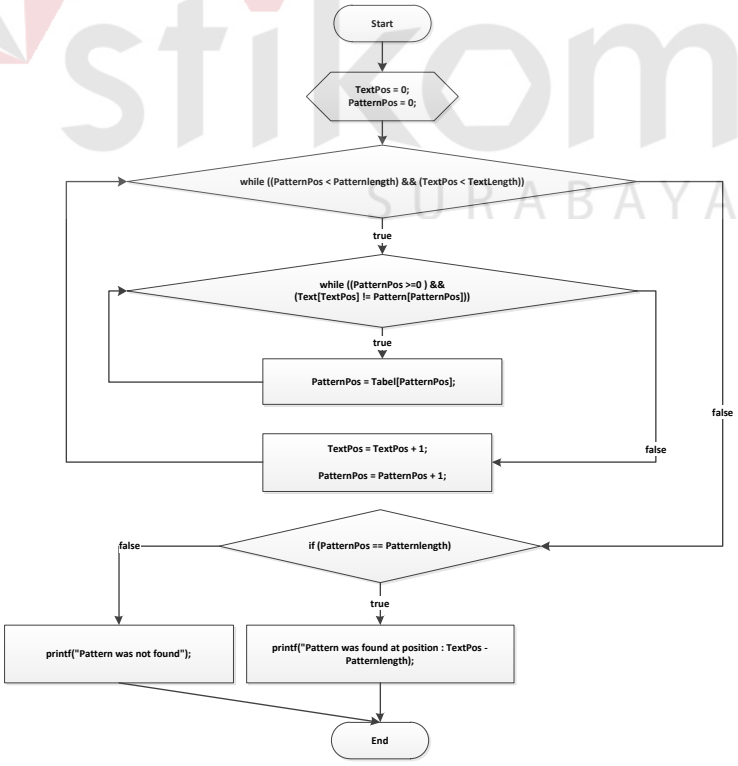


Gambar 3.9 Posisi *Pattern[V]* =0 *Pattern[P]* = 4

Setelah dilakukan perhitungan diatas, maka nilai pinggiran pada pattern "agama" yaitu -1, 0, -1, 1, -1.

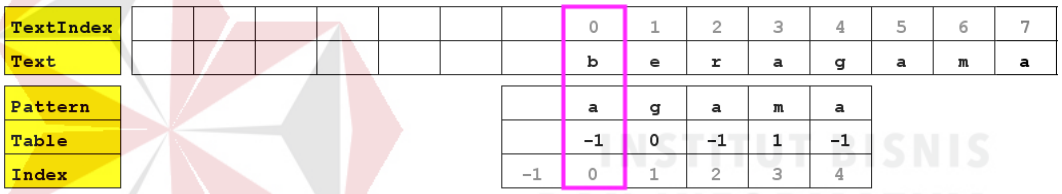
c. *Matching*

Matching merupakan proses terakhir dari pencarian ayat Al-Qur'an. Setelah ditemukan nilai pinggiran dari *pattern* tersebut, maka proses selanjutnya yaitu akan dilakukan pencarian *string* yang dinputkan pada data-data ayat yang ada pada *database*. Gambar 3.10 adalah *flowchart* untuk proses *matching*.



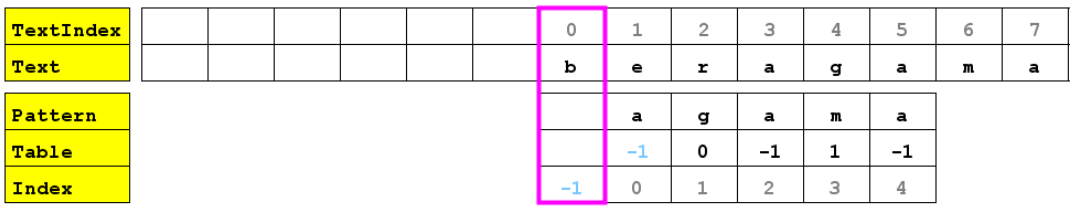
Gambar 3.10 *flowchart Matching*

Sebagai contoh diberikan *pattern* "agama" dan *string (text)* "beragama" untuk memulai proses *matching*. Langkah pertama pada proses *matching* ini yaitu inisialisasi variabel *TextPos = 0* dan *PatternPos = 0*, variabel ini berfungsi untuk membadingkan karakter antara *string* (*text*) dan *pattern*, nilai 0 sendiri yaitu posisi index pada *string* dan *pattern*. Langkah selanjutnya yaitu melakukan pengecekan apakah *PatternPos* < *Patternlength* dan *TextPos* < *TextLength*, jika *true* maka akan dilakukan pencocokan antara karakter pada *Text[TextPos]* dan *Pattern[PatternPos]*. Gambar 3.11 menjelaskan bagaimana proses *matching* dilakukan.



Gambar 3.11 Proses Pencocokan *Pattern* dengan *String*

Pada gambar diatas, dilakukan pencocokan pertama antara karakter pada *Text[TextPos]* dan karakter pada *Pattern[PatternPos]* dimana nilai *TextPos = 0* dan *PatternPos = 0*. Karena *Text[TextPos]* dan *Pattern[PatternPos]* tidak cocok, maka *PatternPos = Tabel[PatternPos]*. Dalam hal ini posisi *PatternPos* akan bergeser ke index -1 sesuai dengan nilai pinggiran dari *PatternPos*.



Gambar 3.12 Pergeseran *PatternPos = Tabel[PatternPos]*

Setelah terjadi ketidakcocokan antara *Text[TextPos]* dan *Pattern[PatternPos]* maka akan dilakukan lagi pencocokan kedua pada index berikutnya dengan nilai *TextPos + 1* dan *PatternPos + 1*.

TextIndex							0	1	2	3	4	5	6	7
Text							b	e	r	a	g	a	m	a

Pattern			a	g	a	m	a
Table			-1	0	-1	1	-1
Index	-1	0	1	2	3	4	

Gambar 3.13 Pencocokan Kedua pada *Text[1]* dan *Pattern[0]*

Karena *Text[1]* dan *Pattern[0]* tidak cocok pada pencocokan kedua, maka *PatternPos = Tabel[PatternPos]* dan *PatternPos* akan bergeser ke index -1 sesuai dengan nilai pinggiran dari *PatternPos* dan dilakukan lagi pencocokan percobaan ketiga dengan nilai *TextPos + 1* dan *PatternPos + 1*.

TextIndex						0	1	2	3	4	5	6	7	
Text						b	e	r	a	g	a	m	a	

Pattern		a	g	a	m	a
Table		-1	0	-1	1	-1
Index	-1	0	1	2	3	4

Gambar 3.14 Pencocokan Ketiga pada *Text[2]* dan *Pattern[0]*

Pada pencocokan ketiga kembali terjadi ketidakcocokan antara *Text[2]* dan *Pattern[0]*, maka nilai *PatternPos = Tabel[PatternPos]* dan *PatternPos* akan bergeser ke index -1 sesuai dengan nilai pinggiran dari *PatternPos* dan dilakukan lagi pencocokan keempat dengan nilai *TextPos + 1* dan *PatternPos + 1*. Pada pencocokan keempat terjadi kecocokan antara *Text[3]* dan *Pattern[0]*, maka akan dilakukan pencocokan berikutnya dengan nilai *TextPos + 1* dan *PatternPos + 1*. Pada contoh ini pencocokan berikutnya yaitu pencocokan keempat sampai kedelapan, karakter pada *Text[TextPos]* dan karakter pada *Pattern[PatternPos]*

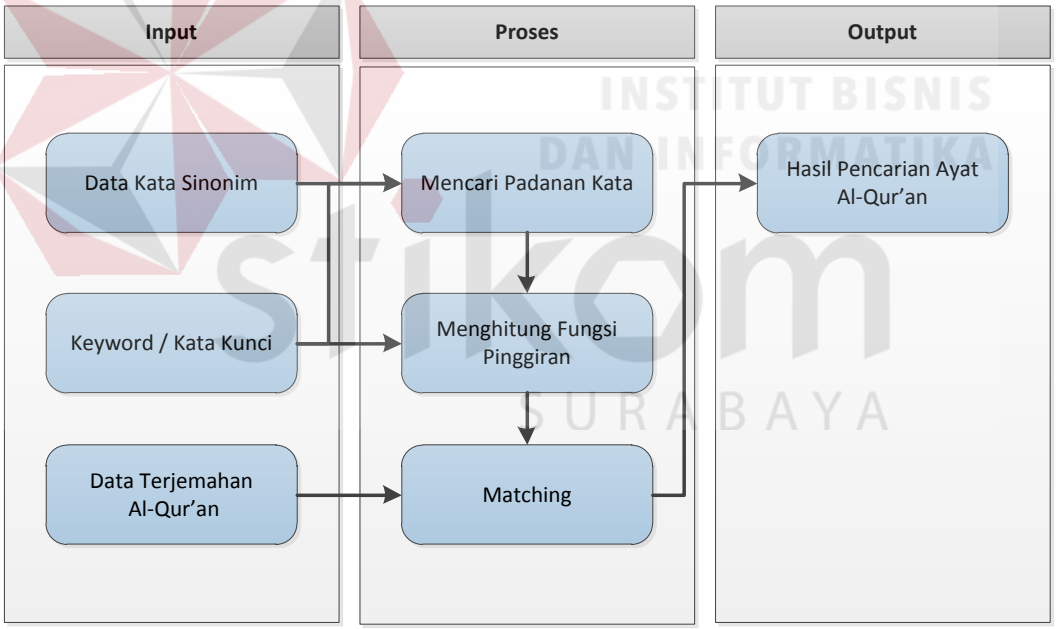
adalah sama, maka pencarian akan berakhir dan akan menghasilkan posisi index dimana pattern ditemukan. Gambar 3.15 menjelaskan pattern ditemukan.

Text Index	-1	0	1	2	3	4	5	6	7
Text		b	e	r	a	g	a	m	a
Pattern		a	g	a	m	a			
Table		-1	0	-1	1	-1			
Index	-1	0	1	2	3	4			

Gambar 3.15 Pattern Ditemukan

3.3.3 IPO Diagram Aplikasi Pencarian Ayat Al-Qur'an

Kebutuhan input, proses, dan output dari aplikasi yang akan dibuat digambarkan dengan diagram IPO berikut ini :



Gambar 3.16 IPO Diagram Pencarian Ayat Al-Qur'an

Pada aplikasi pencarian ayat Al-Qur'an ini memiliki 3 inputan, satu inputan dari *user* yaitu kata kunci dari ayat Al-Qur'an yang ingin dicari, dua data yang di *load* dari *database* yaitu data ayat Al-Qur'an dan data kata sinonim. Proses pencarian sendiri memiliki tiga tahapan, tahap pertama yaitu mencari

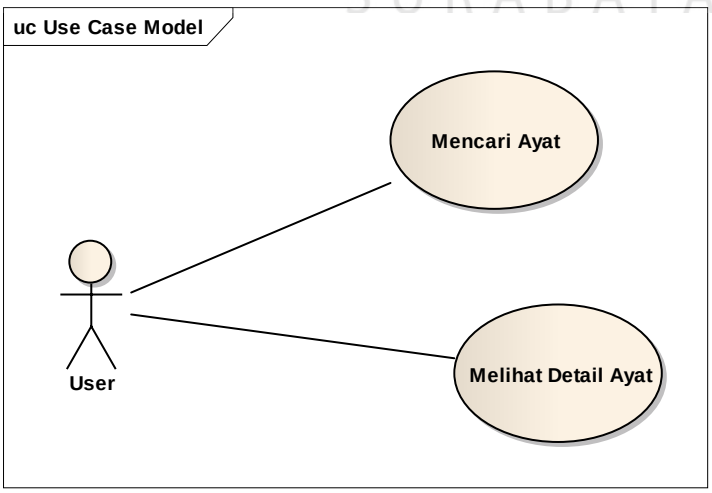
padanan kata dari kata kunci yang diinput oleh *user*, tahap kedua yaitu menghitung nilai fungsi pinggiran dari padanan kata dan kata kunci yang diinputkan dari user, fungsi pinggiran ini berguna untuk meminimalisir pergeseran *index* pada proses *mathing* nanti. Tahap terakhir dari proses yaitu *matching* atau pencocokan kata kunci dengan data terjemahan Al-Qur'an yang ada pada database. Output dari proses ini yaitu sistem akan menampilkan ayat-ayat Al-Qur'an yang sesuai dengan kata kunci tersebut.

3.4 Perancangan Sistem

Perancangan sistem ini menggunakan *Unified Modeling Language* (UML) yang terdiri dari dari *use case diagram, flow of event, activity diagram, sequence diagram,* dan *class diagram.*

3.4.1 Use Case Diagram Aplikasi Pencarian Ayat Al-Qur'an

Kebutuhan-kebutuhan sistem di atas dapat dimodelkan dengan diagram *use case* berikut ini :



Gambar 3.17 Use Case Diagram Aplikasi Pencarian Ayat Al-Qur'an

Interaksi antara user dengan sistem dapat digambarkan dalam *use case* diagram pada gambar 3.17 Pada gambar tersebut terdapat satu aktor yaitu *user* sebagai pengguna aplikasi dan enam *use case*, yaitu :

1. Mencari ayat Al-Qur'an

   Use case pencarian ayat Al-Qur'an adalah proses pencarian dengan menginputkan kata kunci atau ayat yang ingin dicari.

2. Melihat detail ayat Al-Qur'an

   User dapat melihat detail ayat Al-Qur'an dari hasil pencarian dari pencarian yang dihasilkan oleh aplikasi beserta terjemahannya.

3.4.2 *Flow of Event*

Dari *use case* yang ada dibutuhkan *flow of event* untuk menjelaskan lebih jelas aliran proses yang terjadi pada tiap *use case*. *Flow of event* meliputi deskripsi singkat kondisi awal, aliran kejadian aliran utama, aliran kejadian alternatif, dan kondisi akhir. *Flow of event* yang dibuat adalah *flow of event* untuk *use case* mencari ayat Al-Qur'an dan melihat detail ayat Al-Qur'an.

A. *Flow of Event* Mencari Ayat Al-Qur'an

Flow of event Mencari ayat Al-Quran merupakan alur dari proses pencarian ayat Al-Quran. Untuk *flow of Event* mencari ayat Al-Qur'an dapat dilihat pada tabel 3.5.

Tabel 3.5 Flow of Event Mencari Ayat Al-Qur'an

Nama Use Case	**Mencari Ayat Al-Qur'an**
Deskripsi Singkat	User akan meinputkan kata kunci kemudian sistem akan menampilkan hasil pencarian.
Aktor	User
Kondisi Awal	Aktor menginput kata kunci
Alur Utama	1. User menginput keyword pada halaman pencarian. 2. User menklik tombol cari. 3. Sistem melakukan pencarian padanan kata yang mirip dengan kata kunci. 4. Sistem menghitung nilai pinggiran dari kata kunci dan padanan kata 5. Sistem melakukan pencocokan kata kunci dengan terjemahan Al-Qur'an. 6. Sistem menampilkan hasil pencarian yang sesuai kata kunci kepada user.
Alur Alternatif	1. User menginput kata kunci kurang dari 3 karakter 1a. Sistem menampilkan pesan kata kunci harus lebih dari 3. 2. User tidak menginput kata 2a. Sistem akan menampilkan pesan kata kunci harus diisi.
Kondisi Akhir Suskes	Hasil pencarian berdasarkan kata kunci.
Kondisi Akhir Gagal	-Hasil Pencarian tidak ada.

B. *Flow of Event* Melihat Detail Ayat Al-Qur'an

Flow of event untuk *use case* melihat detail ayat Al-Qur'an merupakan alur dari proses melihat detail ayat dari hasil pencarian sebelumnya. *Flow of event* melihat detail aya Al-Quran dapat dilihat pada tabel 3.6.

Tabel 3.6 Flow of Event Melihat Detail Ayat Al-Qur'an

Nama Use Case	**Melihat Detail Ayat Al-Qur'an**
Deskripsi Singkat	User memilih ayat dari salah satu dari hasil pencarian kemudian sistem akan menampilkan detail ayat beserta terjemahannya.
Aktor	User
Kondisi Awal	Aktor memilih ayat
Alur Utama	1. sistem menampilkan hasil pencarian kepada user 2. user memilih salah satu ayat dari hasil pencarian. 3. sistem menampilkan detail ayat dan terjemahan kepada user.
Alur Alternatif	-
Kondisi Akhir Suskes	Detail ayat Al-Qur'an dan terjemahan.
Kondisi Akhir Gagal	-

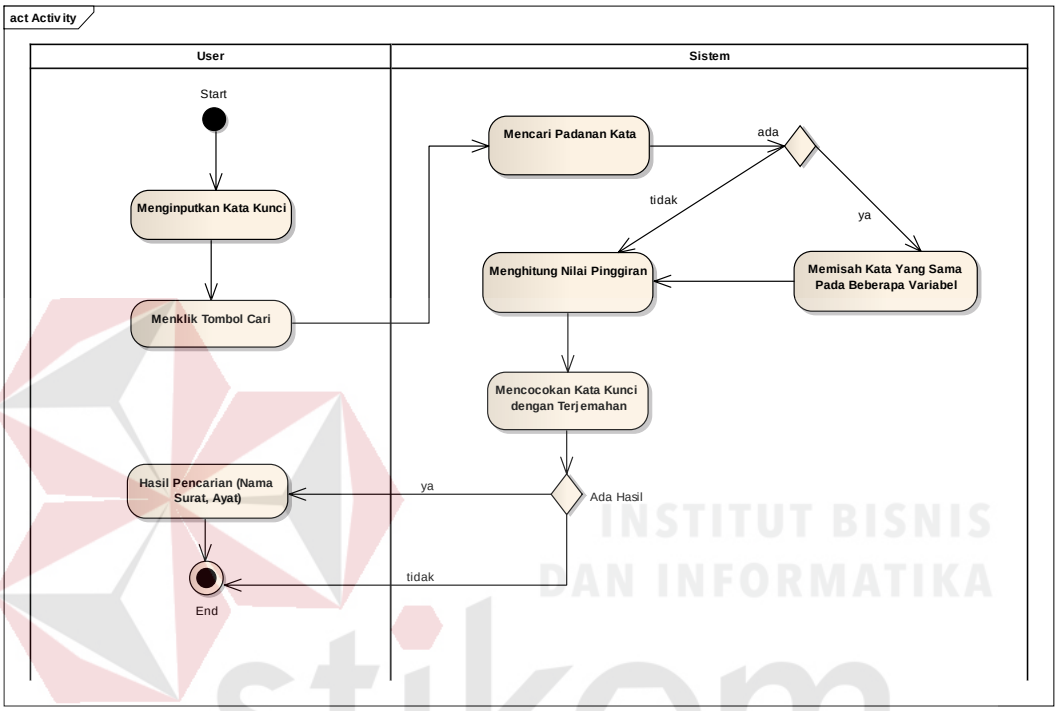
3.4.3 Activity Diagram

Dari use case yang ada dibutuhkan *activity* diagram untuk menjelaskan secara jelas aliran proses pada tiap *use case*. *Activity* diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram akan dijelaskan secara lengkap di bawah ini.

A. Activity Diagram Mencari Ayat Al-Qur'an

Proses pada gambar 3.18 dimulai dari *user* menginputkan kata kunci berupa kata dasar atau topik yang ingin dicari, kemudian user menekan *button* cari untuk memulai proses pencarian dan sistem akan melakukan proses pencarian kata yang mirip dengan daftar kata sinonim, jika ada maka kata-kata tersebut akan dipisah dalam beberapa variabel kemudian sistem menghitung nilai pinggiran dari

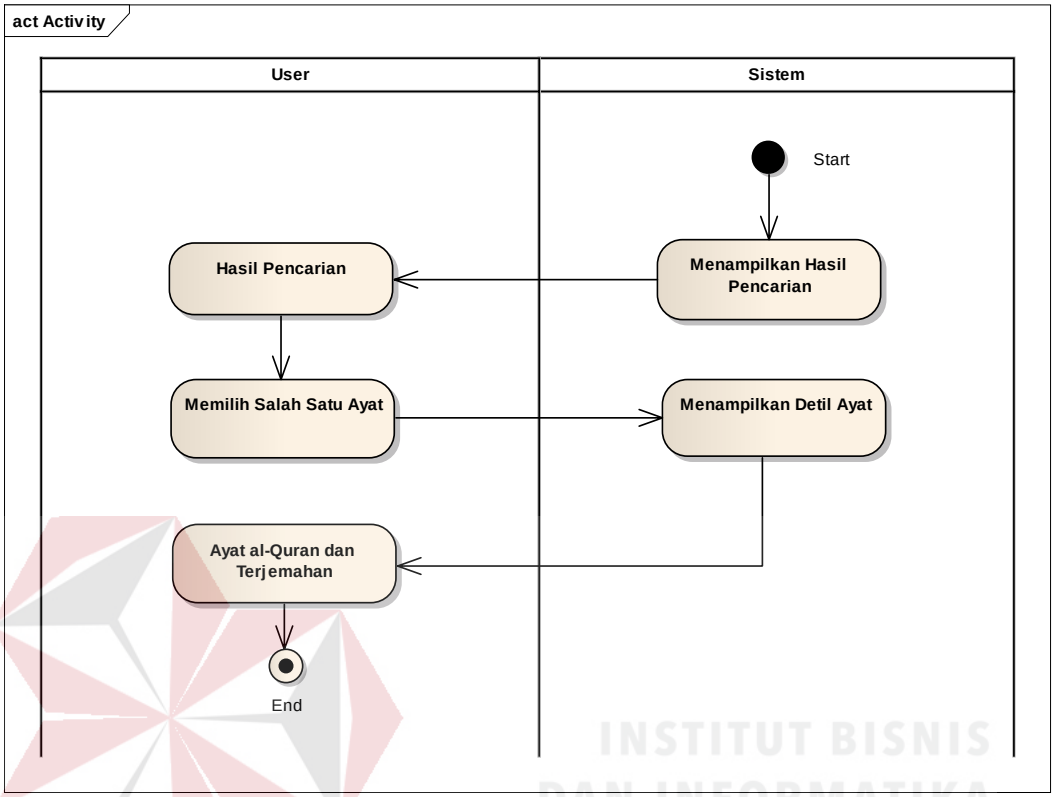
kata kunci tersebut dan kemudian sistem akan melakukan pencocokan dengan terjemahan yang ada pada *database*. jika menemukan kecocokan antara kata kunci dan terjemahan maka sistem akan menampilkan hasil pencarian dalam bentuk list nama surat dan ayat.



Gambar 3.18 Activity Diagram Mencari Ayat Al-Qur'an

B. Activity Diagram Melihat Detail Ayat Al-Qur'an

Pada gambar 3.19 menjelaskan tentang aliran aktifitas pada *use case* melihat detail ayat Al-Qur'an. Proses ini merupakan lanjutan dari proses sebelumnya yaitu mencari ayat Al-Qur'an. Output hasil pencarian dari proses mencari ayat Al-Qur'an itulah yang merupakan aktifitas pertama dari proses ini. Mulanya sistem akan menampilkan hasil pencarian kepada user berupa nama surat, dan ayat, kemudian user akan memilih salah satu ayat dari hasil pencarian dan sistem akan menampilkan detail dari ayat tersebut.



Gambar 3.19 Activity Diagram Melihat Detail Ayat Al-Qur'an

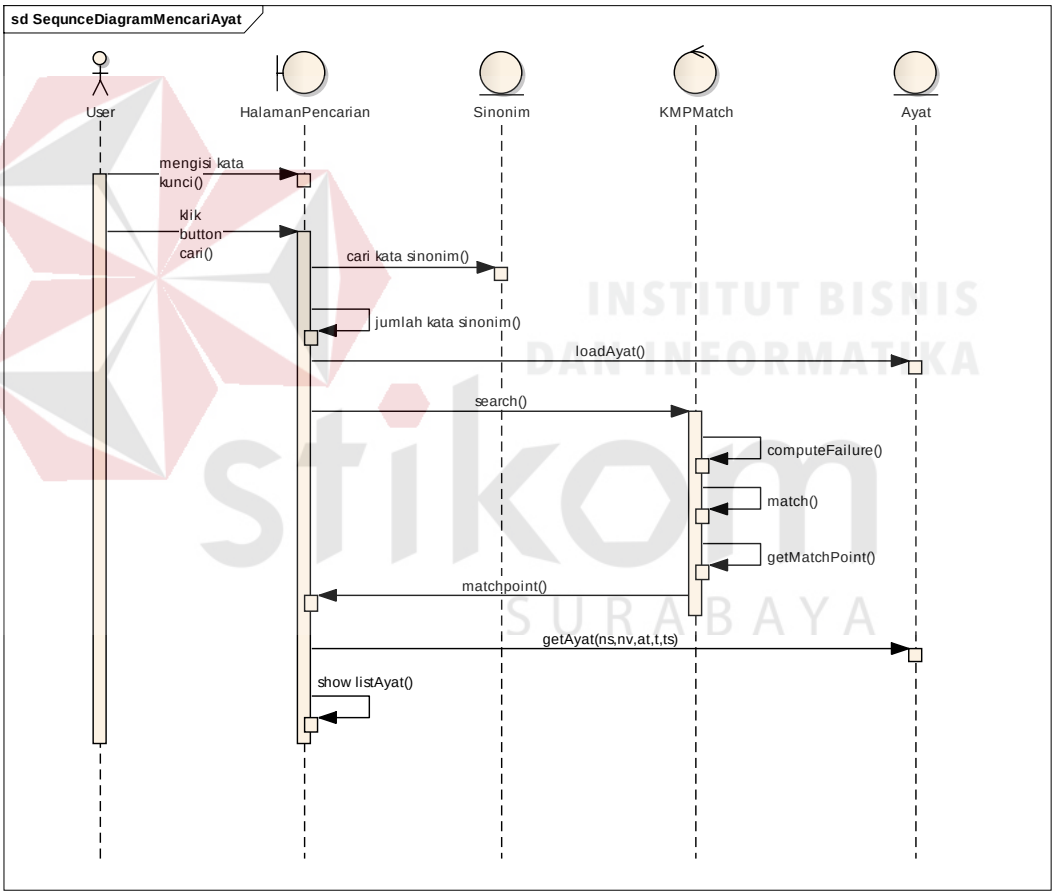
3.4.4 Sequence Diagram

Untuk melihat tahap demi tahap kejadian masing-masing *use case* pada *use case* diagram, maka dapat digunakan *sequence* diagram. *Sequence* diagram secara grafis menggambarkan bagaimana objek berinteraksi antara satu sama lain melalui pesan ekusi pada sebuah *use case* atau operasi.

A. Sequence Diagram Mencari Ayat Al-Qur'an

Pada gambar 3.20 dimulai dari *user* membuka aplikasi dan menginputkan *keyword* atau kata kunci pada halaman pencarian, kemudian user menklik button cari dan sistem akan mencari padanan kata yang mirip dengan kata kunci,

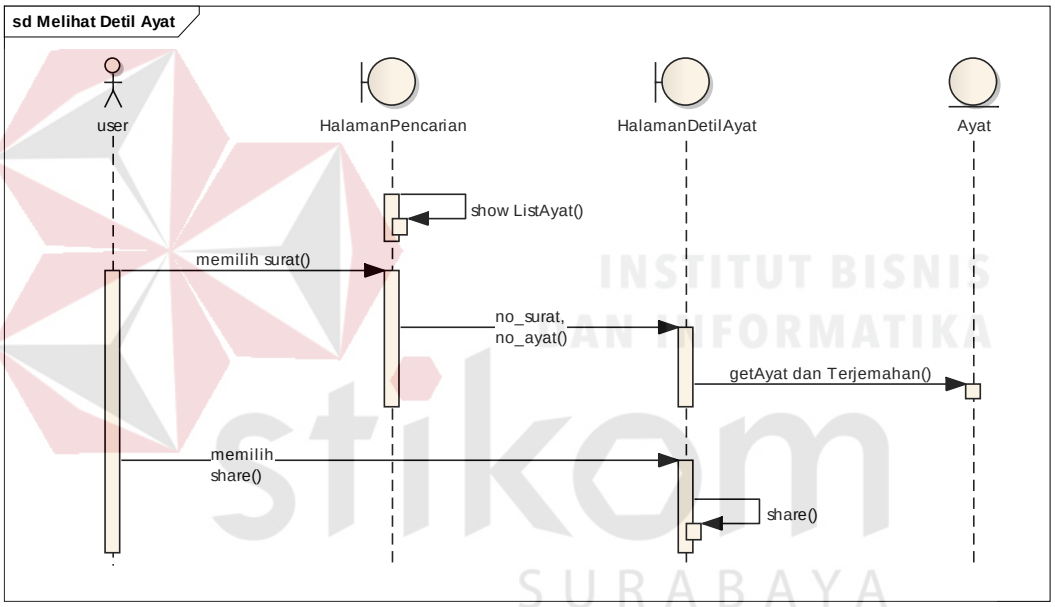
selanjutnya sistem akan mengambil data terjemahan pada database Al-Qur'an dan melakukan pencarian. Pencarian sendiri terdiri dari proses menghitung nilai pinggir, dan *matching* atau pencocokan *keyword* dengan database terjemahan Al-Qur'an, dan mengirimkan *matchpoint* kepada halaman pencarian yaitu pada *index* berapa terjadi kecocokan tersebut. Kemudian sistem akan mengambil surat beserta ayat yang sesuai dengan hasil pencarian. Kemudian menampilkannya pada halaman pencarian.



Gambar 3.20 Sequence Diagram Mencari Ayat Al-Qur'an

B. Sequence Diagram Melihat Detail Ayat Al-Qur'an

Pada gambar 3.21 merupakan lanjutan dari *sequence* diagram mencari ayat Al-Qur'an dimana halaman pencarian sebagai *boundary* memberikan hasil pencarian kepada *user* yaitu berupa list surat beserta ayat. Kemudian *user* memilih salah satu ayat dari hasil pencarian untuk melihat detil dari ayat tersebut, dan sistem akan menampilkan ayat dari *entity* Al-Qur'an pada halaman detail ayat sesuai dengan nomor surat dan ayat.

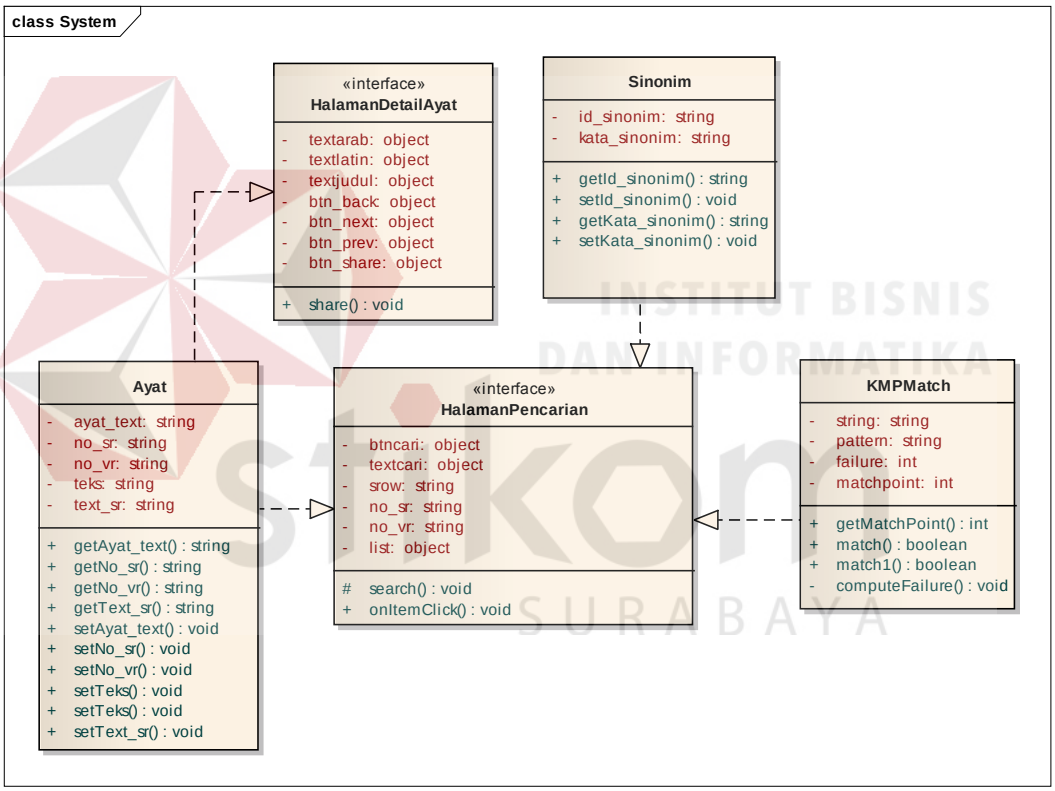


Gambar 3.21 Sequence Diagram Melihat Detil Ayat Al-Qur'an

3.4.5 Class Diagram

Class diagram digunakan untuk menampilkan kelas-kelas atau paket-paket dalam sistem dan relasi antar mereka. Biasanya, dibuat beberapa diagram kelas untuk satu sistem. Satu *class diagram* menampilkan subset dari kelas-kelas dan relasinya. *Class diagram* lainnya mungkin menampilkan kelas-kelas termasuk *attribut* dan operasi dari kelas-kelas pembentuk diagram.

Class diagram adalah alat perancangan terbaik untuk tim pengembang perangkat lunak. *Class diagram* membantu tim pengembang mendapatkan pola kelas-kelas dalam sistem, struktur sistem sebelum menuliskan kode program, dan membantu untuk memastikan bahwa sistem adalah rancangan terbaik dari beberapa alternatif rancangan.(Sholiq, 2010). *Class diagram* memberikan gambaran sistem secara statis dan relasi antar mereka. *Class diagram* pada aplikasi pencarian ayat Al-Qur'an ini dapat dilihat pada Gambar 3.22.



Gambar 3.22 Class Diagram Aplikasi Pencarian Ayat Al-Qur'an

3.4.6 Struktur Tabel

1. Nama Tabel : Quran Indonesia

   Primary Key : -

   Foreign Key : -

   Fungsi : Menyimpan data terjemahan Al-quran

Tabel 3.7 Struktur Tabel Terjemahan Al-Quran

No.	Nama Kolom	Tipe Data	Lebar	Keterangan
1	No_sr	varchar	7000	Nomor Surat
2	No_vr	varchar	7000	Nomor Ayat
3	Ayat_teks	teks		Teks terjemahan ayat

2. Nama Tabel : Quran Arab

   Primary Key : -

   Foreign Key : -

   Fungsi : Menyimpan data Al-quran

Tabel 3.8 Struktur Tabel Al-Quran

No.	Nama Kolom	Tipe Data	Lebar	Keterangan
1	No_sr	varchar	7000	Nomor Surat
2	No_vr	varchar	7000	Nomor Ayat
3	Ayat	Teks		Teks ayat Al-Quran

3. Nama Tabel : Sinonim

   Primary Key : Id_Sinonim

   Foreign Key : -

   Fungsi : Menyimpan data sinonim

Tabel 3.9 Struktur Tabel Sinonim

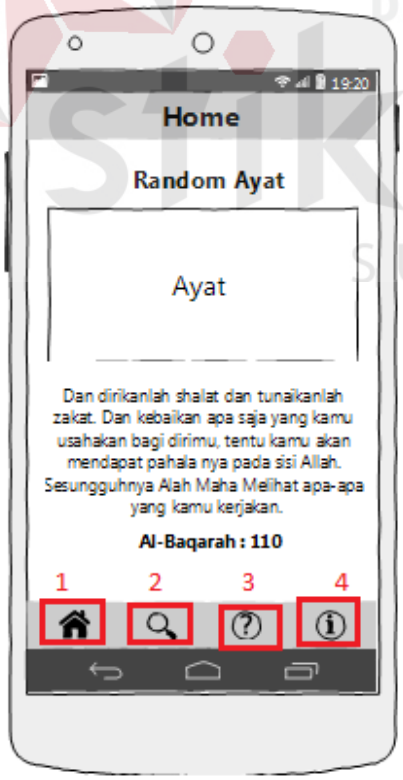
No.	Nama Kolom	Tipe Data	Lebar	Keterangan
1	Id_Sinonim	Integer		Id sinonim
2	Kata_Sinonim	varchar	500	Kata sinonim

3.5 Perancangan User Interface

Perancangan antar muka perangkat lunak atau *user interface* sangat diperlukan agar pengguna dapat berinteraksi dengan aplikasi, sehingga dibutuhkan perancangan secara detail mengenai desain aplikasi berdasarkan informasi yang ditampilkan pada layar *device*. Tampilan yang akan dibuat adalah tampilan halaman utama, halaman pencarian, dan halaman detail ayat.

3.5.1 Tampilan Halaman Utama

Pada gambar 3.23 merupakan desain tampilan pada halaman utama aplikasi. Pada saat pengguna membuka aplikasi pengguna akan ditampilkan dengan ayat random dari Al-Qur'an dan beberapa menu navigasi pada bagian bawah.

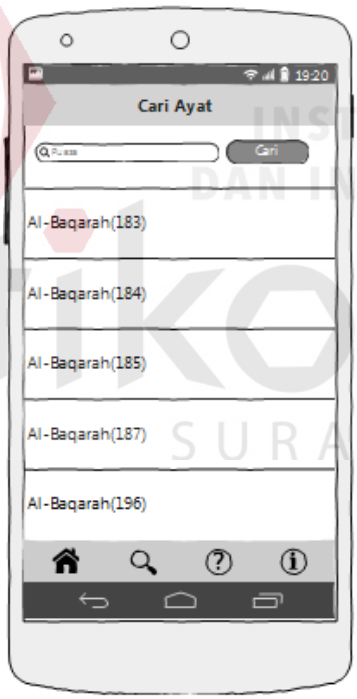


Keterangan :
1. Halaman Home
2. Halaman Cari Ayat
3. Halaman Petunjuk
4. Halaman Info

Gambar 3.23 Desain Tampilan Halaman Utama

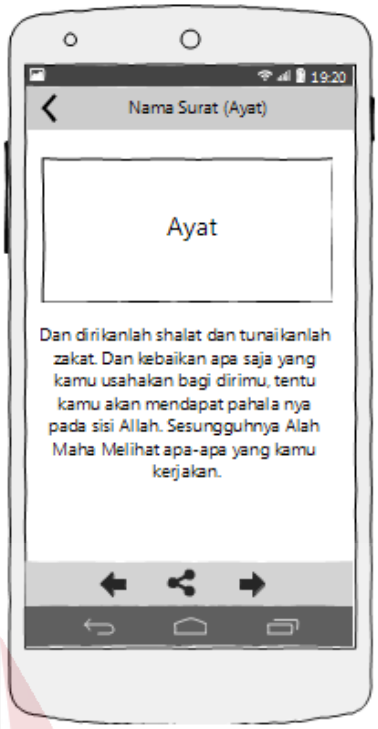
3.5.2 Tampilan Halaman Pencarian

Halaman pencarian adalah tampilan pada saat pengguna ingin melakukan pencarian ayat. Pengguna akan ditampilkan dengan sebuah halaman dengan satu *textboxt* dan satu *button* untuk memulai pencarian ayat Al-Qur'an. Halaman ini juga menampilkan hasil pencarian berupa nama surat beserta ayat berdasarkan kata kunci. Pengguna dapat memilih salah satu surat dan ayat pada hasil pencarian untuk menampilkan detil dari ayat tersebut. Tampilan form utama dari aplikasi dapat dilihat pada Gambar 3.24 berikut ini.



Gambar 3.24 Desain Tampilan Halaman Pencarian

3.5.3 Tampilan Halaman Detail Ayat



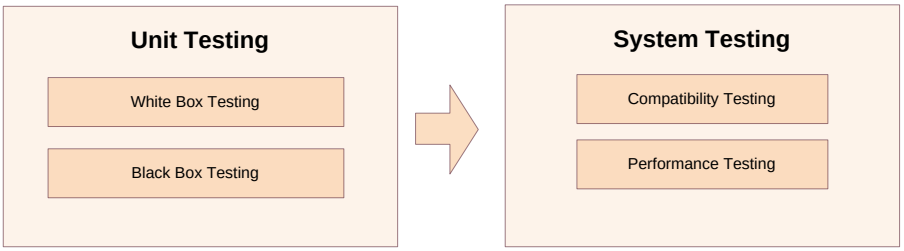
Gambar 3.25 Desain Tampilan Halaman Detail Ayat

Tampilan halamat detail ayat akan menampilkan detail dari hasil pencarian ayat sebelumnya, halaman ini akan menampilkan ayat dalam bahasa Arab dan terjemahan bahasa Indonesia secara lengkap kepada pengguna.

3.6 Desain Uji Coba

Desain uji coba bertujuan untuk memastikan bahwa aplikasi telah dibuat dengan benar dan sesuai dengan tujuan yang diharapkan. Pada pengujian aplikasi ini akan dilakukan 2 tahapan pengujian, yang pertama yaitu *unit testing*, yang terdiri dari *black box testing* dan *white box testing*. Kedua yaitu *system testing*, yang terdiri dari *requirement testing*, dan *performance testing*. Untuk pengujian kinerja aplikasi ini variabel yang diamati yaitu nilai *precision* atau ketepatan

aplikasi dalam menghasilkan ayat yang dicari. Tingkat ketepatan pada aplikasi pencarian ayat ini dikatakan baik jika nilai *presicion* mendekati 1 atau 100%. Desain tahapan pengujian ini dapat dilihat pada Gambar 3.26.



Gambar 3.26 Desain Tahapan Pengujian

3.6.1 Unit Testing

A. White Box Testing

Tujuan dari pengujian ini yaitu untuk mengetahui secara internal cara kerja suatu perangkat lunak. Pada pengujian ini fungsi yang diuji yaitu proses mencari padanan kata, proses menghitung *border function* dan *matching* pada algoritma Knuth-Morris-Pratt. Teknik pengujian yang digunakan pada pengujian ini yaitu menghitung *Cyclomatic Complexity* dari proses-proses tersebut. *Source Code* pada pengujian ini dapat dilihat pada Gambar dibawah ini.

```
1       begin
2       int js = 0;
3       sinonimCursor = dbsinonim.rawQuery("SELECT id_sinonim,
        kata_sinonim FROM sinonim where  kata_sinonim like
        ('%"+textcari.getText().toString()+"%') limit 1", null);
4       if (sinonimCursor.moveToFirst()) {
5       String is = sinonimCursor.getString(0).toString();
6       String ks = sinonimCursor.getString(1).toString();
7       items = ks.split(",");
8       for (String item : items){
9       js++; }
10      search(textcari.getText().toString());
11      end;
```

Gambar 3.27 *Source Code* Mencari Padanan Kata

```
1       begin
2       failure[0] = -1;
3       int i = 0;
4       for (int j = 1;
5       j <= pattern.length() - 1;
6       j++) {
7       if (pattern.charAt(j)== pattern.charAt(i))
8       failure[j] = failure[i];
9       else failure[j] = i;
10      while ((i >= 0) && (pattern.charAt(j)!= pattern.charAt(i)))
11      i = failure[i];
12      i++;
13      end;
```

Gambar 3.28 *Source Code* Menghitung *Border Function*

```
1       begin
2       int j = 0;
3       int i = 0;
4       while ((j < pattern.length()) && (i < string.length())) {
5       while ((j >=0 ) && (string.charAt(i)!= pattern.charAt(j)))
6       j = failure[j];
7       i++;
8       j++; }
9       if (j == pattern.length()) {
10      matchPoint = i - pattern.length();
11      return true; }
12      return false;
13      end;
```

Gambar 3.29 *Source Code* Matching

B. Black Box Testing

Tujuan dari uji coba ini adalah untuk mengetahui apakah fungsi-fungsi dari aplikasi pencarian ayat al Qur'an dengan menggunakan metode algoritma *string matching* Knuth-Morris-Pratt berbasis android ini telah berjalan dengan lancar. Teknik tes pada pengujian ini yaitu *functional analysis*, setiap fitur dan fungsi yang disediakan akan diuji hasilnya sesuai dengan *test case*. Desain uji coba pada aplikasi ini adalah sebagai berikut :

1. Desain Uji Coba Pencarian

Proses pencarian dilakukan dengan cara menginputkan kata kunci atau *keyword* tentang ayat yang ingin dicari kemudian sistem akan melakukan pencarian dan menampilkan hasil pencarian berupa ayat yang sesuai dengan kata kunci. Desain pengujian pada fungsi pencarian ayat dapat dilihat pada tabel 3.10.

Tabel 3.10 Desain Uji Coba Percarian

Tes Case	Tujuan	Input	Output Yang Diharapkan
1	Pencarian ayat Al-Qur'an yang memiliki hasil.	Memasukan kata kunci "shalat".	Hasil pencarian yang sesuai dengan kata kunci.
2	Pencarian ayat Al-Qur'an dengan kata yang mirip dan memiliki hasil.	Memasukan kata kunci "sholat" dan "salat"	Hasil pencarian yang sesuai dengan kata kunci.
3	Pencarian ayat Al-Qur'an yang tidak memiliki hasil.	Memasukan kata kunci "bbb".	Tidak ada hasil.
4	Menampilkan pesan jika textbox kosong.	Menekan tombol cari tanpa mengisi textbox.	Aplikasi menampilkan pesan textbox kosong
5	Menampilkan pesan jika karakter yang diinput kurang dari tiga.	Memasukan kata kunci "aa".	Aplikasi menampilkan pesan jumlah karakter yang diinput minimal tiga.

2. Desain Uji Coba Detail Ayat

Pada proses ini akan dilakukan uji coba ketika user memilih salah satu ayat dari hasil pencarian apakah sistem akan menampilkan detail ayat dan

terjemahan yang sesuai dengan ayat yang dipilih oleh user. Desain pengujian pada fungsi melihat detail ayat dapat dilihat pada tabel 3.11.

Tabel 3.11 Desain Uji Coba Detail Ayat

Tes Case	Tujuan	Input	Output Yang Diharapkan
6	Menampilkan ayat dan terjemahan yang sesuai dengan pilihan user.	Pilih salah satu ayat dari hasil pencarian.	Sistem menampilkan ayat dan terjemahan yang sesuai dengan pilihan user.
7	Menampilkan ayat dan terjemahan sebelumnya sesuai list hasil pencarian.	Pilih *button prev*	Sistem menampilkan ayat sebelumnya seseuai list hasil pencarian.
8	Menampilkan ayat dan terjemahan selanjutnya sesuai list hasil pencarian.	Pilih *button next*	Sistem menampilkan ayat selanjutnya seseuai list hasil pencarian.
9	Menampilkan pesan jika tidak ada ayat selanjutnya.	Pilih *button next* dengan kondisi tidak ada ayat selanjutnya.	Aplikasi menampilkan pesan tidak ada ayat selanjutnya.
10	Menampilkan pesan jika tidak ada ayat sebelumnya	Pilih *button next* dengan kondisi tidak ada ayat sebelumnya.	Aplikasi menampilkan pesan tidak ada ayat sebelumnya.

3.6.2 System Testing

A. Compatibility Testing

Pada uji coba kompabilitas ini bertujuan untuk mengetahui apakah aplikasi pencarian ayat Al-Qur'an ini dapat berjalan dengan baik pada

berbagai macam versi android. Versi android yang akan digunakan untuk uji coba ini yaitu versi 2.3 (*Gingerbread*) sampai dengan 4.4 (*KitKat*).

Tabel 3.12 Daftar Versi Android

No	Versi Android
1	Android 2.3 (*Gingerbread*)
2	Android 4.0 (*Ice Cream Sandwich*)
3	Android 4.1 (*Jelly Bean*)
4	Android 4.4 (*KitKat*)

B. Performance Testing

Uji coba ini bertujuan untuk mengetahui kinerja dari aplikasi pencarian ayat Al-Qur'an dengan menggunakan algoritma Knuth-Morris-Pratt. Pengujian ini akan mengukur tingkat kecepatan dan keakuratan ayat yang dihasilkan dari pencarian. Pada uji coba ini akan diberikan 20 kata kunci yang telah ditetapkan, hasil pencarian dari masing-masing kata kunci tersebut akan dihitung rata-rata lama proses pencarian dari tiap kata kunci yang ditentukan dan *precision*.

Tabel 3.13 Daftar Kata Kunci

No	Kata Kunci
1	Qisas
2	Shalat
3	Wasiat
4	Masjidil Haram
5	Puasa

No	Kata Kunci
6	Thalaq
7	Iddah
8	Utang Piutang
9	Putus Asa
10	Judi
11	Mahar
12	Riba
13	Nadzar
14	Zabur
15	Kiblat
16	Riya
17	Nikah
18	Sedekah
19	Halal
20	Haram