PEMBANGUNAN APLIKASI KLASIFIKASI KODE SURAT BERBASIS ANDROID MENGGUNAKAN ALGORITMA BOYER-MOORE DI KANTOR KECAMATAN CIPARAY

SKRIPSI

Karya Tulis sebagai syarat memperoleh

Gelar Sarjana Komputer dari Fakutas Teknologi Informasi

Universitas Bale Bandung

Disusun oleh:

YOSEP BAHTIAR

NPM. 301170024

PROGRAM STRATA 1

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS BALE BANDUNG

BANDUNG

2021

LEMBAR PERSETUJUAN PEMBIMBING

PEMBANGUNAN APLIKASI KLASIFIKASI KODE SURAT BERBASIS ANDROID MENGGUNAKAN ALGORITMA BOYER-MOORE DI KANTOR KECAMATAN CIPARAY

Disusun oleh :

YOSEP BAHTIAR

NPM. 301170024

Telah diterima dan disetujui untuk memenuhi persyaratan mencapai gelar

SARJANA KOMPUTER

Pada

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BALE BANDUNG

Baleendah, Juli 2021

Disetujui oleh:

Pembimbing Utama	Pembimbing Pendamping
Rustiyana, S.T, M.T	Yusuf Muharam, M.Kom
NIK. 04104808015	NIK. 04104820003

LEMBAR PERSETUJUAN PENGUJI

PEMBANGUNAN APLIKASI KLASIFIKASI KODE SURAT BERBASIS ANDROID MENGGUNAKAN ALGORITMA BOYER-MOORE DI KANTOR KECAMATAN CIPARAY

Disusun oleh :

YOSEP BAHTIAR

NPM. 301170024

Telah diterima dan disetujui untuk memenuhi persyaratan mencapai gelar

SARJANA KOMPUTER

Pada

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BALE BANDUNG

Baleendah, Juli 2021

Disetujui oleh:

Penguji 1	Penguji 2
Yaya Suharya, S.Kom, M.T	Khilda Nistrina, S.Pd, M.Sc
NIK. 01043170007	NIK. 04104820004

LEMBAR PENGESAHAN PROGRAM STUDI

PEMBANGUNAN APLIKASI KLASIFIKASI KODE SURAT BERBASIS ANDROID MENGGUNAKAN ALGORITMA BOYER-MOORE DI KANTOR KECAMATAN CIPARAY

Disusun oleh :

YOSEP BAHTIAR

NPM. 301170024

SKRIPSI ini telah diterima dan disetujui untuk memenuhi persyaratan mencapai gelar

SARJANA KOMPUTER

Pada

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS BALE BANDUNG

Baleendah, Juli 2021

Mengetahui,	Mengesahkan,
Dekan,	Ketua Program Studi
Yudi Herdiana, S.T, M.T	Yusuf Muharam, M.Kom
NIK. 04104808008	NIK. 04104820003

LEMBAR PERNYATAAN KEASLIAN

Saya yang bertanda tangan dibawah ini:

Nama : Yosep Bahtiar

NPM : 301170024

Judul Skripsi:

PEMBANGUNAN APLIKASI KLASIFIKASI KODE SURAT BERBASIS ANDROID MENGGUNAKAN ALGORITMA BOYER-MOORE DI KANTOR KECAMATAN CIPARAY

Menyatakan dengan sebenarnya bahwa penulisan skripsi ini berdasarkan hasil penelitian, pemikiran, dan pemaparan asli dari saya sendiri, baik untuk naskah laporan maupun kegiatan *programming* yang tercantum sebagai bagian dari skripsi ini. Jika terdapat karya orang lain, saya mencantumkan sumber yang jelas.

Pernyataan ini saya buat dengan sesungguhnya dan apabila dikemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karena karya tulis ini dan sanksi lain sesuai dengan peraturan yang berlaku di FAKULTAS TEKNOLOGI INFORMASI UNIVERSITAS BALE BANDUNG.

Demikian surat pernyataan ini saya buat dalam keadaan sadar tanpa paksaan dari pihak manapun.

Balcendah, Juli 2021

Yang membuat pernyataan

Yosep Bahtiar

NPM. 301170024

ABSTRAK

Arsiparis merupakan sebutan untuk orang yang memiliki kompetensi dalam bidang kearsipan, biasanya profesi arsiparis dapat ditemui di lingkungan kantor pemerintahan. Salah satu tugas arsiparis adalah melakukan pengelolaan arsip masuk atau keluar. Dalam kegiatan pengelolaan tersebut arsiparis akan melakukan pencocokan kode pada arsip/surat menggunakan daftar klasifikasi kode arsip yang memiliki lebih dari 2000 indeks secara manual, dengan indeks sebanyak itu, tentunya pencocokan kode yang dilakukan oleh arsiparis tidak efisien.

Penggunaan aplikasi berbasis Android dapat membuat pencarian dan pencocokan kode arsip lebih cepat dan efisien. Agar kinerja pencarian aplikasi lebih cepat dibutuhkan algoritma khusus yang mencocokan *string*. Algoritma pencocokan *string* yang masih populer saat ini adalah algoritma Boyer-Moore, algoritma ini dianggap memiliki hasil paling baik, karena dalam praktiknya, algoritma tersebut bergerak mencocokkan *string* dari arah kanan ke kiri. Penelitian ini dilakukan dengan membangun aplikasi bernama Klasifikasi Kode Surat berbasis Android yang dapat mencari kode arsip dengan mengimplementasikan algoritma Boyer-Moore sebagai algoritma yang melakukan pencocokan *string*. Metode penelitian yang digunakan adalah metode *Waterfall*, metode ini memiliki pendekatan sistematis dan berurutan pada proses pengembangan perangkat lunaknya.

Setelah pembangunan aplikasi selesai, aplikasi diuji dengan menggunakan metode *black-box testing* dan pengujian penerimaan oleh pengguna. *Black-box testing* dilakukan agar aplikasi tidak memiliki kesalahan pada antarmuka dan kinerja, sedangkan pengujian penerimaan dilakukan agar tingkat kelayakan aplikasi dapat dinilai secara langsung oleh pengguna. Diketahui bahwa berdasarkan hasil *black-box testing* dan pengujian penilaian yang dilakukan pengguna, aplikasi dapat berjalan secara lancar dan dinilai efektif serta layak untuk digunakan dalam mencari klasifikasi kode surat.

Kata kunci: algoritma Boyer-Moore, algoritma pencarian *string*, Android, Eclipse, Klasifikasi Kode Surat, SQLite.

ABSTRACT

Archivist is a term for people who have competence in the field of archives, usually the archivist profession can be found in government offices. One of the archivist's duties is to manage incoming or outgoing records. In these management activities, archivists will perform code matching on archives/letters using a list of archive code classifications that have more than 2000 indexes manually, with that many indexes, of course the code matching performed by archivists is not efficient.

The use of Android-based applications can make searching and matching archive code faster and more efficient. For faster application search performance, a special algorithm that matches strings is needed. The string matching algorithm that is still popular today is the Boyer-Moore algorithm, this algorithm is considered to have the best results, because in practice, the algorithm moves to match strings from right to left. This research was conducted by building an application called Android-based Letter Code Classification that can search for archive codes by implementing the Boyer-Moore algorithm as an algorithm that performs string matching. The research method used is the Waterfall method, this method has a systematic and sequential approach to the software development process.

After the application development is complete, the application is tested using the black-box testing method and user acceptance testing. Black-box testing is carried out so that the application does not have errors in the interface and performance, while acceptance testing is carried out so that the feasibility level of the application can be assessed directly by the user. It is known that based on the results of black-box testing and assessment tests carried out by users, the application can run smoothly and is considered effective and feasible to use in searching for letter code classifications.

Keywords: *Boyer-Moore algorithm, string search algorithm, Android, Eclipse, Letter Code Classification, SQLite.*

KATA PENGANTAR

Puji dan syukur peneliti panjatkan atas kehadirat Allah SWT yang mana berkat limpahan Rahmat dan Karunia-Nya, peneliti dapat menyelesaikan Laporan Skripsi yang berjudul "PEMBANGUNAN APLIKASI KLASIFIKASI KODE SURAT BERBASIS ANDROID MENGGUNAKAN ALGORITMA BOYER-MOORE DI KANTOR KECAMATAN CIPARAY" ini dengan baik dan tepat pada waktunya.

Terselesaikannya laporan skripsi ini tidak lepas dari bantuan dan bimbingan dari berbagai pihak, baik secara langsung maupun secara tidak langsung. Oleh karena itu, pada kesempatan ini peneliti ingin menyampaikan ucapan terimakasih yang sebesar-besarnya kepada:

1. Allah SWT yang telah memberikan rahmat dan karunia-Nya dalam proses pengerjaan skripsi ini.
2. Yudi Herdiana, S.T, M.T selaku dekan Fakultas Teknologi Informasi Universitas Bale Bandung.
3. Yusuf Muharam, M.Kom selaku ketua Program Studi Teknik Informatika dan Dosen Pembimbing 2 yang telah memberikan pengarahan dalam membuat laporan skripsi.
4. Rustiyana, S.T, M.T selaku Dosen Pembimbing 1 yang telah memberikan pengarahan dalam membuat laporan skripsi.
5. Keluarga yang sangat, amat, saya cintai yang telah memberikan dukungan dan do'a untuk kelancaran dalam proses pembuatan laporan skripsi ini.
6. Dan juga teman-teman yang saling membantu dalam senang maupun susah serta memberikan dukungan satu sama lain.

Bandung, Juli 2021

Yosep Bahtiar

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Arsip (atau sering disebut dengan surat), merupakan naskah-naskah yang dibuat dan diterima oleh lembaga negara dan badan pemerintahan dalam bentuk corak apapun, baik dalam keadaan tunggal maupun berkelompok dalam rangka pelaksanaan kegiatan pemerintahan. Pada sebuah instansi pemerintahan, pengarsipan seringkali diterjemahkan sebagai suatu kegiatan manajemen naskah-naskah yang dianggap penting atau bersejarah sesuai dengan prosedur yang ada. Salah satunya pada instansi pemerintah tingkat kecamatan yaitu Kantor Kecamatan Ciparay. Kantor Kecamatan Ciparay seperti halnya kantor kecamatan lain di Indonesia, mempunyai tugas melaksanakan sebagian kewenangan pemerintah kabupaten di wilayah kerjanya, yang mencakup bidang pemerintahan, ekonomi, pembangunan, kesejahteraan rakyat dan pembinaan kehidupan masyarakat serta urusan pelayanan umum lainnya yang dilimpahkan oleh pemerintah kabupaten.

Dalam proses pelimpahan wewenangnya pemerintah kabupaten berkomunikasi dengan kantor kecamatan biasanya menggunakan arsip yang ditujukan ke camat atau bidang tertentu yang mewakili tugas khusus atau disebut juga SKPD (satuan kerja perangkat daerah) di kantor kecamatan berdasarkan SOP yang berlaku. Arsip yang dikirimkan terlebih dahulu akan diterima dan dikelola oleh Arsiparis, yang merupakan sebutan untuk orang yang memiliki kompetensi dalam bidang kearsipan. Tugas pokok seorang arsiparis adalah pengelolaan arsip dinamis dan arsip statis, pembinaan kearsipan, serta pengelolaan dan penyajian arsip menjadi informasi, tetapi jika tidak ada yang memegang jabatan arsiparis, sekretariat lain di kantor kecamatan akan merangkap sebagai arsiparis.

Ketika melakukan pengelolaan arsip masuk atau keluar, arsiparis akan mencocokan kode pada arsip dengan yang ada pada daftar klasifikasi kode arsip agar arsip masuk atau keluar dapat didata, klasifikasi kode arsip itu sendiri memiliki

jumlah lebih dari 2000 indeks, akibatnya kegiatan pencocokan atau pencarian yang dilakukan arsiparis akan tidak efisien. Dengan dibangunnya sebuah aplikasi yang dapat mencari klasifikasi kode arsip, maka pekerjaan arsiparis bisa terbantu karena pengoperasiannya hanya tinggal menginputkan klasifikasi kode yang ingin dicari tanpa terlebih dahulu melihat kepada buku klasifikasi kode arsip. Namun dikarenakan indeks data klasifikasi kode arsip yang berjumlah sangat banyak, proses pencarian aplikasi akan berjalan dengan lambat, belum lagi aplikasi tersebut haruslah dibangun pada sistem operasi yang mendukung mobilitas agar dapat dibawa dan dibuka dimana saja.

Untuk menjawab permasalahan terkait mobilitas aplikasi, sistem operasi yang mendukung mobilitas dan yang banyak digunakan oleh masyarakat saat ini adalah Android OS pada *smartphone*. Sedangkan untuk mempercepat dan mempermudah suatu proses pencarian, dibutuhkan suatu algoritma yang dapat memaksimalkan proses pencarian tersebut. Algoritma pencarian yang dianggap memiliki hasil paling baik dalam praktiknya, yaitu algoritma yang bergerak mencocokkan *string* dari arah kanan ke kiri. Algoritma Boyer-Moore merupakan salah satu contoh algoritma yang menggunakan pencocokan *string* dari arah kanan ke kiri. Berdasarkan pemaparan sebelumnya, aplikasi klasifikasi kode arsip harus mengimplementasikan algoritma Boyer-Moore dan berbasis Android, maka judul yang diambil oleh peneliti adalah "PEMBANGUNAN APLIKASI KLASIFIKASI KODE SURAT BERBASIS ANDROID MENGGUNAKAN ALGORITMA BOYER-MOORE DI KANTOR KECAMATAN CIPARAY".

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, diperoleh rumusan masalah diantaranya:

1. Apakah aplikasi Klasifikasi Kode Surat dapat dirancang menggunakan basis Android?
2. Apakah algoritma Boyer-Moore dapat diimplementasikan pada aplikasi Klasifikasi Kode Surat berbasis Android?

3. Apakah penggunaan aplikasi Klasifikasi Kode Surat efektif dan layak digunakan?

1.3 Batasan Masalah

Dalam pelaksanaan penelitian pembangunan aplikasi Klasifikasi Kode Surat ini, peneliti menetapkan beberapa batasan masalah, diantaranya:

1. Metode pencocokan *string* (*string matching*) yang digunakan dalam aplikasi adalah algoritma Boyer-Moore untuk mencari kode surat berdasarkan klasifikasi.
2. Aplikasi hanya dapat diakses dan dipasang pada perangkat bersistem operasi Android versi 4.3 keatas yaitu Kitkat, Lollipop, dst.
3. Aplikasi dibangun menggunakan ADT Eclipse Juno versi rilis tanggal 2 Juli 2014.
4. Aplikasi bersifat *offline* dengan basis data menggunakan SQLite.
5. Basis data aplikasi menggunakan klasifikasi kode yang bersumber dari Lampiran Keputusan Menteri Dalam Negeri Republik Indonesia no. 78 tahun 2012 tentang Tata Kearsipan di Lingkungan Kementerian dalam Negeri dan Pemerintah Daerah, serta Peraturan Menteri Dalam Negeri Republik Indonesia no. 135 tahun 2017 tentang Perubahannya.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dikemukakan di atas, tujuan dari penelitian ini yaitu:

1. Merancang aplikasi yang dapat mempermudah pencarian klasifikasi kode surat dinas resmi berbasis Android.
2. Mengimplementasikan algoritma Boyer-Moore pada aplikasi Klasifikasi Kode Surat berbasis Android.
3. Mengetahui efektifitas penggunaan dan tingkat kelayakan aplikasi Klasifikasi Kode Surat.

1.5 Metodologi Penelitian

Metode penelitian yang dilakukan dalam penelitian ini adalah menggunakan metode pengembangan perangkat lunak *Waterfall*. Menurut (A.S. & Shalahuddin, 2019), metode *waterfall* sering juga disebut dengan model sekuensial linier atau alur hidup yang terurut dari tahap analisis, desain, pengkodean, dan pengujian.

1.5.1 Metode Pengumpulan Data

Metode pengumpulan data yang dilakukan pada penelitian ini meliputi:

1. Observasi, peneliti melakukan pengamatan dan analisis secara langsung ke Kantor Kecamatan Ciparay untuk memperoleh informasi dan gambaran yang jelas terhadap objek penelitian.
2. Studi pustaka, peneliti mencari dan mempelajari pustaka berbentuk jurnal, buku, serta artikel di internet yang relevan terkait dengan objek penelitian.
3. Kuesioner, peneliti melakukan pengumpulan data menggunakan lembar pertanyaan isian yang diisi oleh subjek penelitian terkait dengan kebutuhan aplikasi dan penilaian terhadap aplikasi.

1.5.2 Metode Perancangan

Metode perancangan yang digunakan dalam membangun aplikasi ini adalah model *waterfall*. Berikut adalah tahap-tahap perancangan pada metode *waterfall*:

1. Analisis
2. Desain
3. Pengkodean
4. Pengujian

1.6 Sistematika Penulisan

Penggambaran secara umum dan singkat mengenai bab-bab yang ada dalam penelitian ini adalah sebagai berikut:

BAB I : PENDAHULUAN

Bab ini membahas tentang latar belakang, rumusan masalah, batasan masalah, tujuan dari penelitian, metode penelitian, dan sistematika penulisan.

BAB II : TINJAUAN PUSTAKA

Bab ini membahas tentang rangkuman informasi dari penelitian-penelitian yang telah dilakukan, lalu menentukan pustaka yang relevan dengan masalah yang menjadi objek kajian untuk memperluas basis informasi dalam melakukan penelitian.

BAB III : METODOLOGI

Bab ini membahas tentang kerangka berfikir terhadap penelitian yang dilakukan dan mendeskripsikannya.

BAB IV : ANALISI DAN PERANCANGAN

Bab ini menerangkan tentang analisis yang digunakan sebagai dasar implementasi algoritma Boyer-Moore pada aplikasi pencarian kode klasifikasi. Pemodelan terhadap sistem yang dibangun juga dilakukan untuk menggambarkan muatan dan aliran informasinya.

BAB V : IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi implementasi, pengujian, serta hasil pengamatan terhadap pembangunan aplikasi Klasifkasi Surat yang mengimplementasikan algoritma Boyer-Moore.

BAB IV : KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran terhadap uraian yang telah diberikan pada bab-bab sebelumnya.

BAB II

TINJAUAN PUSTAKA

2.1 Landasan Teori

Landasan teori berisi ringkasan berdasarkan jurnal terkait penelitian yang diambil, berikut ini adalah jurnal-jurnal yang digunakan dalam penelitian:

Menurut (Budi et al., 2016) yang melakukan penelitian tentang "Analisis Pemilihan Penerapan Proyek Metodologi Pengembangan Rekayasa Perangkat Lunak" pada jurnal Teknika vol. V, no. 1, hal. 24–31, tahun 2016. Masalah yang ditemukan adalah tidak adanya metodologi yang benar-benar sesuai dengan semua jenis organsasi, oleh itu solusinya adalah dibutuhkan pendekatan lebih lanjut untuk memilih metodologi mana yang paling sesuai untuk dapat diterapkan pada organisasi tertentu. Metode yang digunakan dalam penelitian ini adalah studi pustaka, analisis, dan menjelaskan metodologi pengembangan perangkat lunak yang meliputi: Linear Sequential Model atau Waterfall, Parallel Model, Iterative Model, Prototyping Model, RAD (Rapid Application Development) Model, Spiral Model, V- Shaped Model dan Agile Development untuk membuat perbandingan yang menunjukan kelebihan dan kelemahan masing-masing. Simpulan pada jurnal tersebut adalah pemilihan metodologi yang tepat sesuai dengan kebutuhan dapat didasarkan pada kriteria penilaian yang terdiri dari kejelasan persyaratan pengguna, keakraban dengan teknologi, sistem kompleksitas, sistem keandalan, jadwal waktu singkat dan visibility jadwal hingga mereferensi beberapa pendapat dari penelitian atau jurnal ilmiah karena berdasarkan hasil penelitian tersebut, tidak ada metodologi yang akan cocok terhadap semua jenis organisasi dan pengembangan perangkat lunak.

Pada penelitian yang dilakukan oleh (Erlinda & Masriadi, 2020) tentang "Perancangan Aplikasi Mobile Kamus Istilah Komputer untuk Mahasiswa Baru Bidang Ilmu Komputer Berbasis Android" pada jurnal Teknologi dan Open Source vol. III, no. 1, hal. 30–43, tahun 2020. Masalah pada penelitian tersebut adalah

kamus komputer yang saat ini beredar masih berbentuk buku, maka solusinya adalah perancangan kamus komputer dalam bentuk aplikasi *mobile* berbasis Android yang saat ini merupakan sistem operasi yang banyak digunakan masyarakat. Metode penelitian yang digunakan adalah metode *waterfall*. Simpulan pada penelitian ini adalah kamus komputer yang dibangun dapat berjalan pada sistem *offline* Android yang mana akan menghemat paket data, sehingga *user* atau mahasiswa dapat mengakses kamus kapan dan dimana saja.

Dalam penelitian yang dilakukan oleh (Sinaga et al., 2016) tentang "Aplikasi Mobile Pencarian Kata pada Arti Ayat Al-Qur'an berbasis Android menggunakan Algoritma *String Matching*" pada jurnal INFOTEK vol. II, no. 2, hal. 68–72, tahun 2016. Masalah yang dibahas dalam penelitian tersebut adalah bahwa sangatlah sulit mencari suatu topik pembahasan dalam Al-Qur'an, mengingat Al-Qur'an mempunyai 114 surah, 30 juz dan 6236 ayat. Sehingga solusinya adalah dengan dibuatnya aplikasi mobile pencarian kata pada arti ayat Al-Qur'an. Metode pengembangan perangkat lunak dimulai dari perancangan dan diakhiri dengan implementasi. Simpulan pada penelitian tersebut adalah algoritma *string matching* Knuth-Morris-Pratt dapat diterapkan dalam aplikasi pencarian ayat Al-Qur'an sehingga dapat memudahkan pengguna untuk mencari arti ayat dalam Al-Qur'an.

Dalam penelitian yang dilakukan oleh (Muntahanah & Darnita, 2019) tentang "Aplikasi Pengarsipan Dokumen Menggunakan Metode *String Matching* (Studi Kasus Perpustakaan SMP Negeri 5 Seluma)" pada jurnal Informatika UPGRIS vol. V, no. 1, hal. 9–16, tahun 2019. Masalah yang pada penelitian tersebut adalah kinerja dan pelayanan di SMP Negeri 5 Seluma yang belum memaksimalkan atau menerapkan penggunaan teknologi informasi. Solusinya adalah dibangunnya aplikasi pengarsipan (perpustakaan) yang menerapkan algoritma *string matching* dalam proses pencarian bukunya. Metode pengumpulan data yang dilakukan adalah wawancara, observasi, dan studi pustaka. Sedangkan metode pengembangan perangkat lunak tidak didefinisikan. Simpulan pada penelitian tersebut adalah pemberian informasi data buku pada sistem perpustakaan SMP Negeri 5 Seluma dengan metode *string matching* memudahkan siswa dan siswi untuk mencari buku yang akan dibaca atau dipinjam.

Pada penelitian yang dilakukan oleh (Astuti, 2017) tentang "Analisis *String Matching* Pada Judul Skripsi dengan Algoritma Knuth-Morris-Pratt (KMP)" pada jurnal Ilmiah ILKOM vol. IX, no. 2, hal. 167–172, tahun 2017. Masalah pada penelitian tersebut adalah banyaknya mahasiswa yang kesulitan untuk memulai mengerjakan tugas akhir karena belum mendapatkan judul penelitian, dan juga pengajuannya yang sering ditolak disebabkan oleh adanya kesamaan judul. Solusinya adalah analisis terhadap implementasi algoritma *string matching* Knuth-Morris-Pratt untuk mencari judul yang kemungkinan memiliki kata yang sama. Metode pada penelitian ini adalah *linear model*. Simpulan pada penelitian tersebut adalah dengan diterapkannya algoritma Knuth-Morris-Pratt, mahasiswa dapat mengetahui judul penelitian yang sudah ada dan belum ada pada sistem yang dibangun.

Menurut (Rusito, 2019) yang melakukan penelitian tentang "Aplikasi Pencarian dengan Menggunakan Algoritma Knuth Morris Pratt Pada Berkas Dokumen Shipment" pada jurnal Riset Komputer (JURIKOM) vol. VI, no. 3, hal. 245–254, tahun 2019. Masalah pada penelitian tersebut adalah efisiensi waktu terhadap pencarian berkas dokumen shipment pada suatu perusahaan yang bilamana hilang akan menghambat atau terselip, pengurusan ekspor atau impor tidak bisa diselesaikan tepat waktu. Solusinya adalah pembuatan aplikasi pencarian yang menggunakan algoritma *string matching* Knuth-Morris-Pratt. Metode penelitian yang diterapkan adalah *Research and Development* (RnD). Simpulan pada penelitian tersebut menghasilkan bahwa efektivitas pencarian berkas dokumen shipment terhadap sistem lama yang sebelumnya hanya 45%, kini berubah menjadi 85% pada sistem yang baru dengan menggunakan algoritma Knuth-Morris-Pratt.

Menurut (Daeli & Hondro, 2017) yang melakukan penelitian tentang "Perancangan Aplikasi Pencarian Kata dengan Kombinasi Algoritma Knuth Morris Pratt dan Algoritma Boyer Moore" pada jurnal Majalah Ilmiah INTI vol. XII, no. 2, hal. 271–275, tahun 2017. Masalah pada penelitian tersebut adalah efisiensi pencarian terhadap pencarian *string* pada dokumen di komputer. Solusinya adalah diterapkannya algoritma *string matching* gabungan dari Knuth-Morris-Pratt dan Boyer-Moore (BM). Metode penelitian tidak dijelaskan. Simpulanya adalah aplikasi pencarian kata yang dirancang dengan algoritma Knuth-Morris-Pratt dan

Boyer-Moore mampu menampilkan hasil berdasarkan kata kunci pada *file* berekstensi .txt di komputer.

Menurut (Yogyawan, 2016) yang melakukan penelitian tentang "Implementasi Boyer-Moore Pada Aplikasi Pencarian Rumus Matematika dan Fisika" pada jurnal Ilmiah Teknologi Informasi Terapan (JITTER) vol. III, no. 1, hal. 74–85, tahun 2016. Masalah yang bahas pada penelitian ini adalah sulitnya menghafal dan mencari rumus matematika dan fisika bagi sebagian pelajar SMA. Solusinya adalah mengimplementasikan algoritma pencocokan *string* Boyer-Moore pada aplikasi untuk melakukan pendeteksian kata kunci yang terdapat pada soal matematika dan fisika. Metode penelitian yang digunakan adalah metode *waterfall*. Simpulan penelitian menghasilkan bahwa metode *string matching* dengan algoritma Boyer-Moore dapat diterapkan pada aplikasi pencarian rumus matematika dan fisika tingkat SMA, disertai dengan pengujian algoritma sebanyak 30 kali dimana aplikasi dapat mencocokkan pola kata kunci pada soal serta menampilkan *list* atau kumpulan rumus matematika dan fisika yang cocok dengan soal yang dikerjakan oleh pengguna.

Menurut (Rifqo & Andilala, 2020) yang melakukan penelitian tentang "Implementasi Algoritme Boyer-Moore pada Aplikasi Kamus Istilah Komputer Berbasis Android" pada jurnal Pseudocode, vol. VII, no. 1, hal. 69–77, tahun 2020. Masalah pada penelitian tersebut adalah kamus istilah komputer yang saat ini beredar masih berbentuk buku, maka solusinya adalah perancangan kamus istilah komputer dalam bentuk aplikasi *mobile* berbasis Android yang saat ini merupakan sistem operasi yang banyak digunakan masyarakat dengan mengimplementasikan algoritma Boyer-Moore sebagai algoritma yang menangani pencarian *string*-nya. Metode penelitian yang digunakan adalah metode studi pustaka, dan model pengembangan perangkat lunaknya digunakan model *incremental*. Simpulan penelitian tersebut menghasilkan bahwa algoritma Boyer-Moore dapat diterapkan dalam perancangan aplikasi kamus istilah komputer sehingga dapat memudahkan pengguna untuk mencari kata atau istilah komputer yang ingin diterjemahkan.

Menurut (Lestari et al., 2016) yang melakukan penelitian tentang "Perancangan Aplikasi Kamus Istilah Medis Berbasis Android dengan Algoritma

Boyer-Moore" pada jurnal INFOTEK vol. II, no. 3, hal. 1–6, tahun 2016. Masalah yang dibahas pada penelitian ini adalah banyaknya istilah medis yang sulit dipahami bagi mahasiswa kedokteran, solusinya adalah merancang aplikasi istilah medis berbasis Android yang menggunakan algoritma Boyer-Moore. Metode penelitian dan SDLC tidak dijelaskan. Simpulan penelitian tersebut menghasilkan bahwa algoritma Boyer-Moore dapat diterapkan dalam perancangan aplikasi kamus istilah medis sehingga dapat memudahkan pengguna untuk mencari kata yang ingin diterjemahkan.

Menurut (Sari, 2019) yang melakukan penelitian tentang "Perancangan Aplikasi Kamus Bahasa Indonesia ke Bahasa Arab dengan Algoritma Boyer Moore berbasis Android" pada jurnal Pelita Informatika vol. VIII, no. 2, hal. 189–192, tahun 2019. Masalah pada penelitian tersebut adalah tebalnya (identik) kamus bahasa arab yang dapat mempersulit dan memperlambat penggunaannya. Solusinya adalah dirancangnya aplikasi kamus bahasa indonesia ke bahasa arab dengan algoritma pencocokan *string* Boyer-Moore berbasis Android. Metode penelitian dan SDLC tidak dijelaskan. Simpulan penelitian tersebut menghasilkan bahwa penerapan algoritma pencocokan *string* Boyer-Moore dapat digunakan untuk mencari kata-kata bahasa Arab dengan melakukan pencocokan terhadap kalimat yang di-*input*-kan menggunakan bahasa Indonesia.

Jurnal terakhir yang dijadikan landasan teori dalam penelitian ini ditulis oleh (Parlika et al., 2020) yang melakukan penelitian tentang "Studi Literatur Kekurangan dan Kelebihan Pengujian *Black-Box*" pada jurnal Teknomatika vol. X, no. 2, hal. 131–140, tahun 2020. Masalah pada penelitian tersebut adalah layak tidaknya pengujian *black-box* digunakan. Solusinya adalah studi literatur pada 30 literatur yang dipilih untuk mengetahui lebih dalam kelebihan dan kekurangannya. Metode yang digunakan adalah metode studi literatur. Simpulan dari penelitian tersebut menyatakan bahwa ada 9 kekurangan dan 10 kelebihan *black-box* yang bersumber dari 7 literatur, maka terbukti bahwa pengujian *black-box* memiliki lebih banyak kelebihan daripada kekurangannya yang menyebabkan pengujian *black-box* sangat layak dan sangat diperlukan untuk menguji luaran suatu produk perangkat lunak.

Dari landasan teori yang telah dijabarkan, dibuat tabel 2.1 ikhtisar tentang perbandingan penelitian pada jurnal-jurnal tersebut di bawah ini.

Tabel 2.1 Ikhtisar Perbandingan Penelitian

No	Nama Jurnal	Pengarang	Kekurangan	Kelebihan	Hasil
1	Analisis Pemilihan Penerapan Proyek Metodologi Pengembangan Rekayasa Perangkat Lunak	Darmawan Setiya Budi, Taghfirul Azhima Yoga Siswa, dan Heri Abijono	Tidak semua metodologi perangkat lunak dibahas dan dianalisis (hanya 7 SDLC).	Pembahasan metodologi perangkat lunak terbilang lengkap (ada pengertian SDLC, gambaran sistemnya, kelebihan, dan kekurangannya).	Pemilihan metodologi tergantung kepada kebutuhan tiap organisasi (tidak ada metodologi yang cocok terhadap semua organisasi).
2	Perancangan Aplikasi Mobile Kamus Istilah Komputer untuk Mahasiswa Baru Bidang Ilmu Komputer Berbasis Android	Erlinda dan Masriadi	Algoritma *string matching* yang digunakan merupakan algoritma *built-in* Android, sehingga proses pencarian bisa lebih lama jika data pada basis data banyak.	Kamus dapat berjalan *offline* sehingga menghemat paket data, dan berbasis Android yang mana bersifat *mobile*.	Kamus komputer yang dibangun dapat berjalan dan dapat diakses.

3	Aplikasi Mobile Pencarian Kata pada Arti Ayat Al-Qur'an berbasis Android menggunakan Algoritma *String Matching*	Jailamm Igaph Sinaga, Mesran, dan Efori Buulolo	Tidak disebutkan bahwa aplikasi dapat berjalan secara *offline*.	Aplikasi menggunakan algoritma *string matching* KMP, berbasis Android yang mana bersifat *mobile*.	Algoritma *string matching* dapat diterapkan pada aplikasi *mobile* pencarian ayat Al-Qur'an.
4	Aplikasi Pengarsipan Dokumen Menggunakan Metode *String Matching* (Studi Kasus Perpustakaan SMP Negeri 5 Seluma)	Muntahanah dan Yulia Darnita	Sistem berjalan secara *online*, sehingga dibutuhkan paket data untuk mengaksesnya.	Sistem menggunakan algoritma *string matching* KMP, berbasis Web sehingga dapat diakses secara *online*.	Aplikasi yang dibangun dinilai dapat memudahkan para siswa SMPN 5 Seluma dalam mengakses informasi data buku.
5	Analisis *String Matching* Pada Judul Skripsi dengan Algoritma Knuth-Morris-Pratt (KMP)	Wistiani Astuti	Sistem berjalan secara *online*, sehingga dibutuhkan paket data untuk mengaksesnya.	Sistem menggunakan algoritma *string matching* KMP, berbasis Web sehingga dapat diakses secara *online*.	Mahasiswa dapat mengetahui judul penelitian menggunakan algoritma KMP.

6	Aplikasi Pencarian dengan Menggunakan Algoritma Knuth Morris Pratt Pada Berkas Dokumen Shipment	Nurul Khasanah Rusito	Sistem berjalan secara *online*, sehingga dibutuhkan paket data untuk mengaksesnya.	Sistem menggunakan algoritma *string matching* KMP, berbasis Web sehingga dapat diakses secara *online*.	Dengan diterapkannya algoritma KMP, efektifitas pencarian dokumen berubah menjadi 85%.
7	Perancangan Aplikasi Pencarian Kata dengan Kombinasi Algoritma Knuth Morris Pratt dan Algoritma Boyer Moore	Makrina Meti Yana Daeli dan Rivalri Kristianto Hondro	Aplikasi hanya sebatas mencari *string* pada dokumen berekstensi .txt, serta aplikasi tidak mendukung mobilitas.	Aplikasi menggunakan kombinasi antara algoritma string matching KMP dan BM, berbasis Desktop *offline* tanpa perlu paket data.	Aplikasi mampu menampilkan hasil pencarian pada dokumen berekstensi .txt.
8	Implementasi Boyer-Moore Pada Aplikasi Pencarian Rumus Matematika dan Fisika	Halim Agung Yogyawan	Setelah pattern dimasukan, terlebih dahulu tombol cari harus ditekan agar pencarian dapat berjalan.	Algoritma *string matching* pada aplikasi menggunakan algoritma BM, berbasis Android *offline* tanpa perlu paket data.	Aplikasi dapat mencocokkan pola kata kunci pada soal serta menampilkan rumus.

9	Implementasi Algoritme Boyer-Moore pada Aplikasi Kamus Istilah Komputer Berbasis Android	Muhammad Husni Rifqo dan Andilala	Setelah pattern dimasukan, terlebih dahulu tombol cari harus ditekan agar pencarian dapat berjalan.	Aplikasi menggunakan algoritma BM, berbasis Android *offline* tanpa perlu paket data.	Algoritma BM dapat diterapkan dalam aplikasi kamus komputer.
10	Perancangan Aplikasi Kamus Istilah Medis Berbasis Android dengan Algoritma Boyer-Moore	Citra Puji Lestari, Nelly Astuti Hasibuan, dan Guidio Leonarde Ginting	Tidak bisa mengupdate basis data lewat aplikasi secara *online*.	Aplikasi menggunakan algoritma BM, berbasis Android *offline* tanpa perlu paket data. Pencarian *string* tinggal menginput *pattern* yang ingin dicari, tanpa perlu menekan suatu tombol.	Algoritma BM dapat diterapkan dalam aplikasi kamus komputer.
11	Perancangan Aplikasi Kamus Bahasa Indonesia Kebahasa Arab dengan Algoritma Boyer-Moore Berbasis Android	Linda Sari	Setelah pattern dimasukan, terlebih dahulu tombol cari harus ditekan agar pencarian dapat berjalan.	Aplikasi menggunakan algoritma BM, berbasis Android *offline* tanpa perlu paket data.	Algoritma BM dapat digunakan dalam aplikasi kamus Bahasa Indonesia ke Bahasa Arab.

12	Studi Literatur Kekurangan dan Kelebihan Pengujian *Black-Box*	Rizky Parlika, Tasya Ardhian N., Shavira Maya Ningrum, dan Berlianda Adha Haque	–	Pembahasan materi tentang studi pengujian *black-box* lengkap. Literatur yang diamati berjumlah 30.	Pengujian *black-box* memiliki lebih banyak kelebihannya dibandingkan kekurangan-nya.
13	Pembangunan Aplikasi Klasifikasi Kode Surat berbasis Android menggunakan Algoritma Boyer Moore di Kantor Kecamatan Ciparay	Yosep Bahtiar	Tidak bisa mengupdate basis data lewat aplikasi secara *online*.	Aplikasi menggunakan algoritma BM, *offline* berbasis Android, dan *pattern* yang ingin dicari tinggal di-*input* tanpa perlu menekan suatu tombol.	Algoritma BM dapat diterapkan serta mempercepat dalam pencarian klasifikasi kode surat.

Berdasarkan tabel ikhtisar perbandingan penelitian diatas, manfaat yang didapatkan oleh peneliti dari jurnal-jurnal yang dipilih tersebut adalah peneliti bisa mendapatkan informasi tentang kelebihan, kekurangan, dan hasil tiap penelitian yang berhubungan dengan algoritma pencocokan *string*. Dengan informasi tersebut, peneliti jadi mengetahui hal yang akan dilakukan kedepannya pada penelitian yang dilakukan dan fitur yang akan ditambahkan pada aplikasi, seperti fitur aplikasi pada penelitian yang dilakukan oleh (Lestari et al., 2016) dimana *user* yang hanya tinggal mengetikan *pattern* atau *string* yang ingin dicari tanpa perlu menekan suatu tombol.

2.2 Dasar Teori

Dasar teori berisikan beberapa teori yang digunakan untuk membangun aplikasi Klasifikasi Kode Surat, berikut merupakan dasar-dasar teori yang digunakan.

2.2.1 Arsip dan Surat

Menurut (Tasyhar, 2013), Secara etimologi istilah arsip berasal dari bahasa yunani "*Arche*" yang berarti "permulaan", kemudian dari kata menjadi "*arche*" berkembang menjadi kata "*Ta Archia*" yang berarti catatan, selanjutnya berubah lagi menjadi "*Archeon*" yang berarti "Gedung Pemerintahan", dan kemudian dalam bahasa latinnya berbunyi "*Archivium*". Arsip didefinisikan sebagai rekaman informasi dari aktivitas dan kegiatan suatu organisasi. Rekaman informasi arsip dapat digunakan untuk perencanaan, pelaksanaan serta pengawasan kegiatan suatu organisasi.

Berdasarkan fungsinya, arsip dapat dibagi menjadi 2 jenis, diantaranya:

1) Arsip dinamis, yakni arsip yang masih dipergunakan secara langsung dalam perencanaan, pelaksanaan, dan atau penyelenggaraan administrasi perkantoran.
2) Arsip statis, yaitu arsip yang tidak dipergunakan lagi secara langsung dalam perencanaan, pelaksanaan, atau penyelenggaraan administrasi perkantoran, atau sudah tidak dipakai lagi dalam kegiatan perkantoran sehari-hari.

Menurut (Hasugian, 2003), Istilah warkat sering digunakan untuk mendefinisikan arsip. Warkat berasal dari bahasa Arab yang berarti surat, merupakan setiap informasi tertulis, tercetak atau bergambar (surat-surat, catatan-catatan, perhitungan-perhitungan, grafis-grafis, atau gambar-gambar) yang masih memiliki kegunaan sebagai bahan informasi dan ingatan bagi organisasi.

Dari beberapa pengertian di atas dapat disimpulkan bahwa arsip dan surat dianggap memiliki pengertian yang sama dan keduanya dapat diartikan sebagai dokumen tertulis yang mempunyai arti dan nilai historis, disimpan, serta dijaga akan keberadaannya ditempat khusus sebagai bahan rujukan/referensi.

2.2.2 Klasifikasi Kearsipan

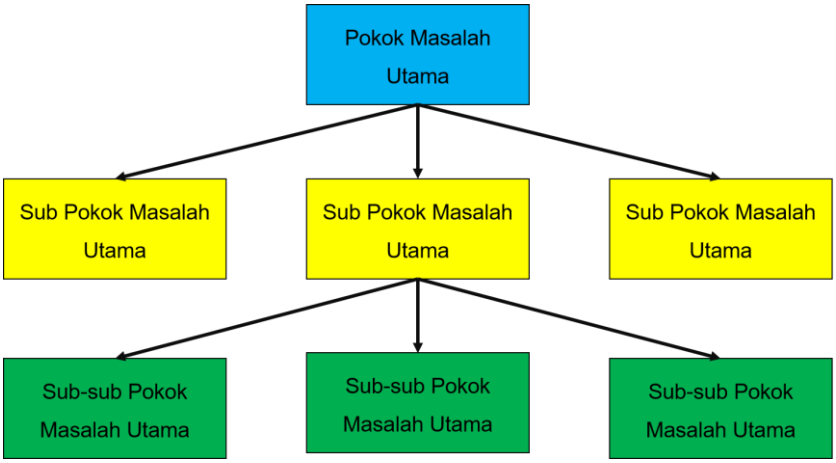
Menurut (Tasyhar, 2013), Kearsipan berarti penyimpanan secara tetap dan teratur terhadap warkat-warkat penting mengenai kemajuan organisasi. Penyimpanan arsip dilakukan dengan menggunakan cara tertentu secara sistematis dengan maksud untuk membantu dan mempermudah dalam penyimpanan dan penemuan kembali arsip tersebut. Metode penyimpanan yang sistematis tersebut sering disebut dengan sistem penyimpanan arsip (*filing system*). Sistem penyimpanan arsip terdiri atas 5 sistem, yaitu:

1) Sistem *filing* abjad adalah sistem penerimaan, penyusunan, penyimpanan, penggunaan, pemeliharaan, dan penemuan kembali surat/warkat dengan mengunakan petunjuk abjad nama orang atau nama organisasi menurut tata urutan abjad.
2) Sistem tanggal, adalah sistem penyimpanan dan penemuan kembali arsip berdasarkan tanggal, bulan, atau tahun.
3) Sistem subyek (pokok masalah), yang dimaksud dengan pengelolaan arsip sistem pokok masalah adalah tata cara penyimpanan dan penemuan kembali arsip (arsip surat masuk maupun arsip surat keluar) berdasarkan subyek atau pokok masalah/perihal dari arsip itu.
4) Sistem wilayah, adalah sistem penyimpanan dokumen, berkas dan/atau arsip yang dijadikan pedoman untuk menyimpan dan menemukan kembali arsip dengan berdasarkan wilayah dari pengirim surat atau wilayah yang kita kirimi surat.

5) Sistem nomor, yang dimaksud sistem penyimpanan arsip berdasarkan nomor adalah sistem penyimpanan dan penemuan kembali arsip dengan menggunakan kode angka/nomor.

Berdasarkan lampiran (Permendagri, 2012) no. 78, Klasifikasi kearsipan merupakan klasifikasi yang disusun berdasarkan masalah (sistem subyek), mencerminkan fungsi dan kegiatan pelaksanaan tugas dari semua satuan organisasi dalam lingkungan Kementerian Dalam Negeri, dan Pemerintah Daerah yaitu menyelenggarakan sebagian tugas umum pemerintahan dan pembangunan dibidang pemerintahan umum dan otonomi daerah, ideologi, politik, pembangunan desa, dan agraria.

Pola klasifikasinya disusun secara berjenjang dengan mempergunakan prinsip perkembangan dari umum kepada khusus dalam hubungan masalah, didahului oleh 3 perincian dasar, masing-masing perincian pertama, perincian kedua dan perincian ketiga sebagai pola dasar yang berfungsi sebagai jembatan penolong dalam menemukan kode masalah yang tercantum dalam pola klasifikasi.



Gambar 2.1 Pola Kode Klasifikasi Arsip (Tasyhar, 2013)

Kode masalah sendiri merupakan alat untuk mengenali masalah yang dikandung dalam arsip, dan disamping itu juga sebagai alat penentu, dimana letak arsip itu dalam urutan hubungan masalahnya pada susunan seluruh arsip dalam simpanan. Berikut merupakan ke-10 pokok kode masalah yang telah menampung seluruh kegiatan pelaksanaan tugas

Kementerian Dalam Negeri termasuk instansi-instansi dalam lingkungannya:

- 000 Umum
- 100 Pemerintahan
- 200 Politik
- 300 Keamanan dan Ketertiban
- 400 Kesejahteraan
- 500 Perekonomian
- 600 Pekerjaan Umum dan Ketenagaan
- 700 Pengawasan
- 800 Kepegawaian
- 900 Keuangan

2.2.3 Algoritma *String Matching*

Menurut (Sinaga et al., 2016), Algoritma *string matching* merupakan algoritma pencocokan *string* yang bersifat mencari sebuah *string* yang terdiri dari beberapa karakter (yang biasa disebut *pattern*) dalam jumlah besar teks. Menurut (Effendi et al., 2013), Algoritma *string matching* memiliki prinsip kerja seperti berikut ini:

1. Memindai teks dengan bantuan sebuah *window* yang ukurannya sama dengan panjang *pattern*.
2. Menempatkan *window* pada awal teks.
3. Membandingkan karakter pada *window* dengan karakter dari *pattern*. Setelah pencocokan (baik hasilnya cocok atau tidak cocok) dilakukan pergeseran ke kanan pada *window*. Prosedur ini dilakukan berulang-ulang sampai *window* berada pada akhir teks. Mekanisme ini disebut mekanisme *sliding window*.

a. Komponen Algoritma *Sring Matching*

Menurut (Effendi et al., 2013), Algoritma *string matching* mempunyai tiga komponen utama, yaitu:

1. *Pattern*, yaitu deretan karakter yang akan dicocokkan dengan teks, dinyatakan dengan $x[0 \ldots m-1]$, panjang *pattern* dinyatakan dengan m.
2. Teks, yaitu tempat pencocokan *pattern* dilakukan. Dinyatakan dengan y$[0 \ldots n-1]$, panjang teks dinyatakan dengan n.
3. Alfabet, berisi semua simbol yang digunakan bahasa pada teks dan *pattern*, dinyatakan dengan Σ dengan ukuran dinyatakan $ASIZE$.

b. Cara Kerja *String Matching*

Menurut (Knuth et al., 1977), Cara yang jelas untuk mencari *pattern* yang cocok dengan teks adalah dengan mencoba mencari di setiap posisi awal dari teks dan mengabaikan pencarian secepat mungkin jika karakter yang salah ditemukan. Proses pertama adalah menyelaraskan bagian paling kiri dari *pattern* dengan teks. Kemudian dibandingkan karakter yang sesuai dari teks dan *pattern*. Setelah seluruhnya cocok maupun tidak cocok dari *pattern*, *window* digeser ke kanan sampai posisi $(n-m+1)$ pada teks. Menurut (Verma & Singh, 2011), efisiensi dari algoritma terletak pada dua tahap:

1. Tahap praproses, tahap ini mengumpulkan informasi penuh tentang *pattern* dan menggunakan informasi ini pada tahap pencarian.
2. Tahap pencarian, *pattern* dibandingkan dengan *window* dari kanan ke kiri atau kiri ke kanan sampai kecocokan atau ketidakcocokan terjadi.

Dengan sebuah nilai karakter $(m < n)$ yang akan dicari dalam teks. Dalam algoritma pencocokan *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.

c. Klasifikasi Algoritma *String Matching*

Menurut (Charras & Lecroq, 2004), Algoritma *string matching* dapat diklasifikasikan menjadi tiga bagian menurut arah pencariannya, yaitu:

1. *From left to right* (dari kiri ke kanan *pattern*), yang merupakan arah untuk membaca. Algoritma yang termasuk dalam kategori ini adalah algoritma Brute Force, algoritma Knuth-Morris-Pratt, dsb.
2. *From right to left* (dari arah kanan ke kiri *pattern*), arah yang biasanya menghasilkan hasil terbaik secara partikal. Contoh algoritma ini adalah algoritma Boyer-Moore, algoritma Zhu Takaoka, algoritma Horspool, dsb.
3. *In a specific order* (dari arah yang ditentukan secara spesifik oleh algoritma tersebut), arah ini menghasilkan hasil terbaik secara teoritis. Algoritma yang termasuk kategori ini adalah algoritma Colussi dan algoritma Crochemore-Perrin.

d. Teknik Algoritma *String Matching*

Menurut (Singla & Garg, 2012), ada dua teknik utama dalam algoritma *string matching*, yaitu:

1. Exact *string matching*. Merupakan pencocokan *string* secara tepat dengan susunan karakter dalam *string* yang dicocokkan memiliki jumlah maupun urutan karakter dalam *string* yang sama. Bagian algoritma ini bermanfaat jika pengguna ingin mencari *string* dalam dokumen yang sama persis dengan *string* masukan. Beberapa algoritma exact *string matching* antara lain: algoritma Brute Force, algoritma Knuth-Morris-Pratt, algoritma Boyer-Moore, dsb.

2. *Approximate string matching* (*Fuzzy string matching*). *Fuzzy string matching* merupakan pencocokan *string* secara samar, maksudnya pencocokan *string* dimana *string* yang dicocokkan memiliki kemiripan memiliki susunan karakter yang berbeda (mungkin jumlah atau urutannya), tetapi *string* tersebut memiliki kemiripan baik kemiripan tekstual/penulisan (*approximate string matching*) atau kemiripan ucapan (*phonetic string matching*).

2.2.4 Algoritma Boyer-Moore

Menurut (Effendi et al., 2013), Algoritma Boyer-Moore adalah algoritma *string matching* yang paling efisien dibandingkan algoritma *string matching* lainnya. Sebelum melakukan pencarian *string*, algoritma melakukan proses terlebih dahulu pada *pattern*, bukan pada *string* pada teks tempat pencarian. Algoritma ini melakukan pencocokan karakter yang dimulai dari kanan ke kiri.

Menurut (Sari, 2019), Algoritma Boyer-Moore adalah salah satu algoritma untuk mencari suatu *string* di dalam teks yang dibuat oleh R.M Boyer dan J.S Moore. Algoritma Boyer-Moore melakukan perbandingan dimulai dari kanan ke kiri, tetapi pergeseran *window* tetap dari kiri ke kanan. Jika terjadi kecocokkan maka dilakukan perbandingan karakter teks dan karakter pola yang sebelumnya, yaitu dengan sama-sama mengurangi indeks teks dan pola masing-masing sebanyak satu. Dengan menggunakan algoritma ini, secara rata-rata proses pencarian akan menjadi lebih cepat jika dibandingkan dengan algoritma lainnya.

Menurut (Charras & Lecroq, 2004), Algoritma Boyer-Moore menggunakan dua buah tabel untuk mengolah informasi saat terjadi kegagalan pencocokan *pattern*. Tabel pertama disebut *bad character shift* juga sering disebut *occurrence heuristic* (OH). Tabel kedua disebut dengan istilah *good suffix shift* juga disebut *match heuristic* (MH).

a. Kelebihan dan Kelemahan Boyer-Moore

Menurut (Halim, 2016), Boyer-Moore membandingkan karakter dari kanan ke kiri dan memiliki loncatan karakter yang besar sehingga mempercepat pencarian *string* karena dengan hanya memeriksa sedikit karakter, dapat langsung diketahui bahwa *string* yang dicari tidak ditemukan dan dapat digeser ke posisi berikutnya. Kelemahan dari algoritma ini adalah ketika semua karakter memiliki kesamaan atau cocok hanya pada karakter terakhir atau karakter paling kiri yang berbeda maka pencarian ini akan memerlukan waktu yang sedikit lama.

b. Tahap Pencarian Boyer-Moore

- Buat tabel pergeseran *pattern* yang dicari (P) dengan pendekatan *Match Heuristic* (MH) dan *Occurence Heuristic* (OH), untuk menentukan jumlah pergeseran yang akan dilakukan jika mendapat karakter tidak cocok pada proses pencocokan dengan teks (T).
- Jika dalam proses pembandingan terjadi ketidakcocokan antara pasangan karakter pada *pattern* dan karakter teks, pergeseran dilakukan dengan memilih salah satu nilai pergeseran dari dua tabel, dan memiliki nilai pergeseran paling besar dari tabel *Match Heuristic* dan *Occurence Heuristic*.
- Dua kemungkinan penyelesaian dalam melakukan pergeseran *pattern*, Jika karakter yang tidak cocok, tidak ada pada *pattern* maka pegeseran adalah sebanyak jumlah karakter pada *pattern*. dan jika karakter yang tidak cocok, ada pada *pattern*, maka banyaknya pergeseran bergantung dari nilai pada tabel *Match Heuristic* dan *Occurence Heuristic*.
    - Jika karakter pada teks yang sedang dibandingkan cocok dengan karakter pada *pattern*, maka posisi karakter pada *pattern* dan teks diturunkan sebanyak 1 posisi, kemudian lanjutkan dengan pencocokan pada posisi tersebut dan seterusnya. Jika kemudian terjadi ketidakcocokan karakter *pattern* dan teks, maka pilih

nilai pergeseran terbesar dari tabel *match heuristic* dan nilai tabel *occurence heuristic*.

- Jika semua karakter telah cocok, artinya *pattern* telah ditemukan di dalam teks.

c. Formulasi Boyer-Moore

Misalnya ada sebuah usaha pencocokan yang terjadi pada $teks[i..i+n-1]$, dan anggap ketidakcocokan pertama terjadi di antara $teks[i+j]$ dan $pattern[j]$, dengan $0 < j < n$. Berarti, $teks[i+j+1..i+n-1] = pattern[j+1..n-1]$ dan $a = teks[i+j]$ tidak sama dengan $b = pattern[j]$. Jika u adalah akhiran dari *pattern* sebelum b dan v adalah sebuah awalan dari *pattern*, maka penggeseran-penggeseran yang mungkin adalah:

- Penggeseran *good-suffix* yang terdiri dari menyejajarkan potongan $teks[i+j+1..i+n-1] = pattern[j+1..n-1]$ dengan kemunculannya paling kanan di *pattern* yang didahului oleh karakter yang berbeda dengan $pattern[j]$. Jika tidak ada potongan seperti itu, maka algoritma akan menyejajarkan akhiran v dari $teks[i+j+1..i+n-1]$ dengan awalan dari *pattern* yang sama.
- Penggeseran *bad-character* yang terdiri dari menyejajarkan $teks[i+j]$ dengan kemunculan paling kanan karakter tersebut di *pattern*. Bila karakter tersebut tidak ada di *pattern*, maka *pattern* akan disejajarkan dengan $teks[i+n+1]$.

Secara sistematis, langkah-langkah yang dilakukan algoritma Boyer-Moore pada saat mencocokkan string adalah:

- Algoritma Boyer-Moore mulai mencocokkan *pattern* pada awal teks.
- Dari kanan ke kiri, algoritme ini akan mencocokkan karakter per karakter *pattern* dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
    - Karakter di *pattern* dan di teks yang dibandingkan tidak cocok.

    - Semua karakter di *pattern* cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
- Algoritma kemudian menggeser *pattern* dengan memaksimalkan nilai penggeseran *good-suffix* dan penggeseran *bad-character*, lalu langkah 2 diulang sampai *pattern* berada di ujung teks (Wikipedia, n.d.-a).

d. Pseudocode Boyer-Moore

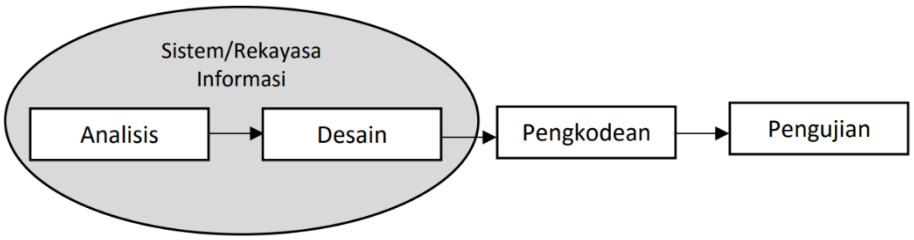
```
procedure BoyerMooreSearch(
       input m, n : integer,
       input P : array[0..n-1] of char,
       input T : array[0..m-1] of char,
       output ketemu : array[0..m-1] of boolean
       )

Deklarasi:
i, j, shift, bmBcShift, bmGsShift: integer
BmBc : array[0..255] of interger
BmGs : array[0..n-1] of interger

Algoritma:
preBmBc(n, P, BmBc)
preBmGs(n, P, BmGs)
i:=0
while (i <= m-n) do
       j:=n-1
       while (j >=0 n and T[i+j] = P[j]) do
              j:=j-1
       endwhile
       if(j < 0) then
              ketemu[i]:=true;
       endif
       bmBcShift:= BmBc[chartoint(T[i+j])]-n+j+1
       bmGsShift:= BmGs[j]
       shift:= max(bmBcShift, bmGsShift)
       i:= i+shift
```

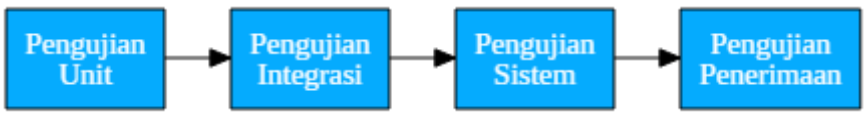
2.2.5 *Software Develoment Life Cycle* (SDLC) *Waterfall*

Menurut (A.S. & Shalahuddin, 2019), Model SDLC *waterfall* (air terjun) sering juga disebut model sekuensial linier (*sequential linear*) atau alur hidup klasik. Model *waterfall* menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut, dimulai dari analisis, desain, pengkodean, dan pengujian. Berikut adalah gambar model *waterfall*:



Gambar 2.2 SDLC Model *Waterfall* (A.S. & Shalahuddin, 2019)

1. Analisis, yaitu proses pengumpulan kebutuhan yang dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak yang sesuai dengan kebutuhan *user*.
2. Desain, merupakan tahap yang menstranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya.
3. Pengkodean, tahap yang menghasilkan sebuah perangkat lunak yang sesuai dengan desain yang telah dibuat pada tahap desain.
4. Pengujian, tahap ini adalah set aktifitas yang direncanakan dan sistematis untuk menguji atau mengevaluasi kebenaran yang diinginkan. Aktifitas pengujian terdiri dari satu set atau sekumpulan langkah dimana dapat menempatkan desain kasus uji spesifik dan metode pengujian. Tahapan pengujian secara keseluruhan adalah sebagai berikut:



Gambar 2.3 Tahapan Pengujian (A.S. & Shalahuddin, 2019)

Pengujian diawali dari pengujian unit, unit disini dapat berupa kumpulan fungsi atau prosedur (*package*), setelah itu dilanjutkan ke pengujian integrasi yang menguji penggabungan dari dua atau lebih unit. Lalu pengujian sistem dilakukan dimana unit-unit proses yang telah terintegarsi diuji dengan antarmuka yang sudah dibuat sehingga pengujian sistem dilakukan untuk menguji sistem secara keseluruhan. Pengujian untuk memvalidasi perangkat lunak memiliki beberapa pendekatan, diantaranya:

- *Black-box testing* (pengujian kotak hitam), yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.
- *White-box testing* (pungujian kotak putih), yaitu menguji perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi-fungsi, masukan, dan keluaran sesuai dengan spesifikasi kebutuhan. Pengujian ini dilakukan dengan memeriksa lojik dari kode program.

Pengujian penerimaan perangkat lunak dilakukan oleh pelanggan untuk mengetahui kepuasan pelanggan atau *user* terhadap perangkat lunak yang sudah dibuat. Jika pelanggan sudah puas dengan perangkat lunak, maka perangkat lunak dapat diserahkan kepada pelanggan.

2.2.6 *Unified Modeling Language* (UML)

Menurut (Dharwiyanti & Wahono, 2003), *Unified Modelling Language* (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model

untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun.

Menurut (A.S. & Shalahuddin, 2019), *Unified Modeling Language* adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan requirement, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek.

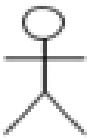
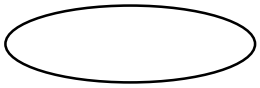
Menurut (Pressman, 2001), *Unified Modeling Language* (UML) *allows a software engineer to express an analysis model using a modeling notation that is governed by a set of syntactic, semantic, and pragmatic rules.*", yang artinya UML memungkinkan seorang pengembang perangkat lunak untuk mengekspresikan sebuah model analisis menggunakan sebuah notasi pemodelan yang diatur oleh sebuah set dari sintaks, semantik, dan aturan pragmatis.

Berikut adalah 4 macam diagram UML yang digunakan dalam penelitian ini, yaitu *use case diagram, class diagram, activity diagram*, dan *sequence diagram*.

a. *Use Case Diagram*

Menurut (A.S. & Shalahuddin, 2019), Use case merupakan pemodelan untuk kelakuan (*behavior*) perangkat lunak yang akan dibuat. Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan perangkat lunak yang akan dibuat. Secara kasar, use case digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Tabel 2.2 Simbol *Use Case Diagram*

Simbol	Deskripsi
	Aktor, adalah segala hal diluar sistem (bisa manusia, sistem, atau perangkat) yang akan menggunakan sistem tersebut untuk melakukan sesuatu.
	*Use Case*, merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.
	Asosiasi, mengidentifikasikan interaksi antara setiap aktor tertentu dengan setiap *use case* tertentu. Digambarkan sebagai garis antara aktor terhadap *use case* yang bersangkutan.
	Generalisasi, menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik.
	Include, menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada digaris dengan panah).
	Extend, menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disisipkan kedalam elemen yang ada di garis dengan panah.

b. *Class Diagram*

Menurut (Dharwiyanti & Wahono, 2003), *Class* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok:

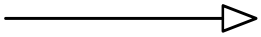
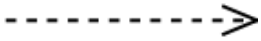
1. Nama (dan stereotype)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut:

- *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
- *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
- *Public*, dapat dipanggil oleh siapa saja.

Tabel 2.3 Simbol *Class Diagram*

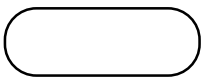
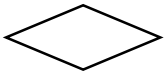
Simbol	Deskripsi
nama_kelas +atribut +operasi()	*Class* pada struktur sistem.
———————	Asosiasi, relasi antarkelas dengan makna umum, biasanya juga disertai dengan *multiplicity*.

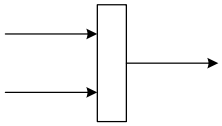
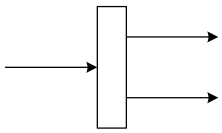
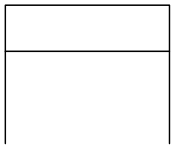
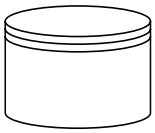
	Asosiasi berarah, relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, biasanya juga disertai dengan *multiplicity*.
	Generalisasi, relasi antarkelas dengan makna umum-khusus.
	*Dependency*, relasi antarkelas dengan makna kebergantungan antarkelas.
	*Aggregation*, relasi antarkelas dengan makna semua bagian (*whole-part*).

c. *Activity Diagram*

Menurut (A.S. & Shalahuddin, 2019), *Activity diagram* menggambarkan aliran kerja atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.

Tabel 2.4 Simbol *Activity Diagram*

Simbol	Deskripsi
	*Start Point*, Menunjukkan dimana aliran kerja dimulai.
	*End Point*, Menunjukkan dimana aliran kerja berakhir.
	*Activities*, Menunjukkan kegiatan dalam aliran kerja.
	*Dicision*, Menunjukkan dimana sebuah keputusan perlu diambil dalam aliran kerja.

	*Join*, Menunjukkan percabangan pada aliran kerja.
	*Fork*, Menunjukkan penggabungan dalam aliran kerja.
	*Swimline*, Pemisah bagian yang bertanggung jawab terhadap aktivitas yang terjadi.
	Penyimpanan Data, Merupakan media penyimpanan data yang tersedia

d. *Sequence Diagram*

Menurut (A.S. & Shalahuddin, 2019), *Sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek.

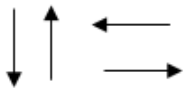
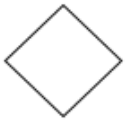
Tabel 2.5 Simbol *Sequence Diagram*

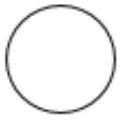
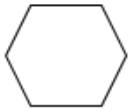
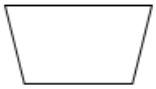
Simbol	Deskripsi
	*Lifeline*, objek entitas yang saling berinteraksi.
	*Activation*, mengindikasikan sebuah objek yang akan melakukan sebuah aksi.
	Message, spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.

2.2.7 *Flowchart*

Menurut (Wilandari, 2018), *Flowchart* adalah suatu bagan dengan simbol-simbol tertentu yang mengambarkan urutan proses secara mendetail dan berhubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program. *Flowchart* sangat penting untuk diterapkan sebelum kita membuat program yang kita buat nanti agar dapat sesuai rencana dan dapat meminimalisir kesalahan-kesalahan pada program. Berikut ini merupakan tabel simbol yang digunakan dalam suatu *flowchart*:

Tabel 2.6 Simbol *Flowchart*

Simbol	Deskripsi
	*Terminal Point*, menunjukkan permulaan atau akhir dari suatu proses.
	*Flow Direction*, simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain.
	*Processing*, digunakan untuk menunjukkan kegiatan yang dilakukan oleh komputer.
	*Decision*, merupakan simbol yang digunakan untuk memilih proses atau keputusan berdasarkan kondisi yang ada.
	*Input-Output*, menunjukkan proses input-output yang terjadi tanpa bergantung dari jenis peralatannya.
	*Predefined Process*, merupakan simbol yang digunakan untuk menunjukkan pelaksanaan suatu bagian prosedur.

	*Connector (On-page)* Simbol ini fungsinya adalah untuk menyederhanakan hubungan antar simbol yang letaknya berjauhan atau rumit bila dihubungkan dengan garis dalam satu halaman.
	*Connector (Off-page)*, sama seperti on-page connector, hanya saja simbol ini digunakan untuk menghubungkan simbol dalam halaman berbeda.
	*Preparation*, merupakan simbol yang digunakan untuk mempersiapkan penyimpanan di dalam storage.
	*Manual Operation*, digunakan untuk menunjukkan kegiatan/ proses yang tidak dilakukan oleh komputer.
	*Document*, artinya input berasal dari dokumen dalam bentuk kertas, atau output yang perlu dicetak di atas kertas.

2.2.8 StarUML

Menurut situs (SMK Taruna Bakti, 2013), StarUML adalah software pemodelan yang mendukung UML (*Unified Modeling Language*). Berdasarkan pada UML version 1.4 dan dilengkapi 11 macam diagram yang berbeda, mendukung notasi UML 2.0 dan juga mendukung pendekatan MDA (*Model Driven Architecture*) dengan dukungan konsep UML. StarUML dapat memaksimalkan produktivitas dan kualitas dari suatu *software project*.

2.2.9 Java

Java pertama kali diluncurkan pada tahun 1995 sebagai bahasa pemrograman umum (*general purpose programming language*) dengan kelebihan dia bisa dijalankan di web browser sebagai applet (Utama, 2002).

Menurut situs (Wikipedia, n.d.-c), Java merupakan sebuah bahasa pemrograman yang populer dikalangan para akademisi dan praktisi komputer. Java pertama kali dikembangkan untuk memenuhi kebutuhan akan sebuah bahasa komputer yang ditulis satu kali dan dapat dijalankan dibanyak sistem komputer yang berbeda tanpa melakukan perubahan kode, Java merupakan bahasa pemrograman berorientasi objek (PBO) atau *Objected-Oriented Programming* (OOP), yang diturunkan dari C++ dengan banyak penyempurnaan.

Menurut (Juhara, 2016), Sebelum membangun aplikasi yang menggunakan bahasa Java seperti Android, JDK (Java *Development Kit*) harus terpasang terlebih dahulu pada komputer. Menurut situs (zonaprogramer, 2016), JDK (Java *Development Kit*) adalah paket fungsi API untuk bahasa pemrograman Java, meliputi Java *Runtime Environment* (JRE) dan Java *Virtual Machine* (JVM). JDK adalah perangkat lunak yang digunakan untuk melakukan proses kompilasi dari kode Java ke *bytecode* yang dapat dimengerti dan dapat dijalankan oleh JRE. JDK wajib terpasang pada komputer yang akan melakukan proses pembuatan aplikasi berbasis Java, namun tidak wajib terpasang di komputer yang akan menjalankan aplikasi yang dibangun dengan Java. Tanpa adanya JDK maka kode-kode Java yang sudah dibuat tidak akan bisa di jadikan aplikasi berbasis Java.

2.2.10 *Extensible Markup Language* (XML)

Menurut (Junaedi, 2003), XML merupakan kependekan dari *eXtensible Markup Language*, yang dikembangkan mulai tahun 1996 dan mendapatkan pengakuan dari W3C pada bulan Februari 1998. Teknologi yang digunakan pada XML sebenarnya bukan teknologi baru, tapi

merupakan turunan dari SGML yang telah dikembangkan pada awal 80-an dan telah banyak digunakan pada dokumentasi teknis proyek-proyek berskala besar.

Seperti halnya HTML, XML juga menggunakan elemen yang ditandai dengan tag pembuka (diawali dengan ‘<’ dan diakhiri dengan ‘>’), tag penutup (diawali dengan ‘</ ‘diakhiri ‘>’) dan atribut elemen (parameter yang dinyatakan dalam tag pembuka misal <form name=”isidata”>).

2.2.11 Android

Menurut situs (Wikipedia, n.d.-b), Android adalah sistem operasi berbasis Linux yang dirancang untuk perangkat bergerak layar sentuh seperti *smartphone* (telepon pintar) dan komputer tablet. Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, yang kemudian dibeli oleh Google sendiri pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya Open Handset Alliance, konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler.

Menurut (Safaat H., 2015), Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, middleware dan aplikasi kunci yang dirilis oleh Google. Saat ini disediakan Android SDK sebagai alat bantu untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java.

Android SDK (*Software Development Kit*) adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk memulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Android SDK biasanya sudah diikutsertakan dalam IDE (*Integrated Development Environtment*), tetapi jika belum, *source* SDK Android dapat dilihat dan diunduh langsung di situs resmi pengembang Android di www.developer.android.com.

Sejak resmi diluncurkan sebagai sistem operasi *open-source*, Android telah mengalami beberapa kali pembaruan. Pembaruan ini dapat dilihat dari versi Android pertama rilis versi beta hingga versi 5.0 yakni versi Lollipop yang dapat dilihat pada tabel dibawah ini.

Tabel 2.7 Versi Android (Versi Beta s/d Lollipop)

Versi	Nama	Rilis	Keterangan
Beta	Android Beta	5 November 2007	Versi pertama Android
1.0	Android 1.0	23 September 2008	
1.1	Android 1.1	9 Februari 2008	
1.5	Cupcake	30 April 2009	Mulai memakai kode nama *dessert*
1.6	Donut	15 September 2009	Peningkatan fitur pencarian dan UI yang lebih user friendly
2.0-2.1	Éclair	26 Oktober 2009 (2.0) 12 Januari 2010 (2.1)	Penambahan fitur untuk pengoptimalan *hardware*
2.2	Froyo	20 Mei 2010	Peningkatan pada kecepatan membuka dan menutup aplikasi
2.3	Gingerbread	6 Desember 2010	Memaksimalkan kemampuan aplikasi
3.0-3.2	Honeycomb	22 Februari 2011 (3.0) 10 Mei 2011 (3.1) 15 Juli 2011 (3.2)	Khusus untuk tablet
4.0	Ice Cream Sandwich	19 Oktober 2011	
4.1-4.3	Jelly Bean	9 Juli 2012 (4.1) 13 November 2012 (4.2) 24 Juli 2013 (4.3)	Pembaruan UI dan fitur pencarian
4.4	Kit Kat	3 September 2013	Peningkatan fitur Google
5.0	Lollipop	25 Juni 2014	Desain ulang UI

2.2.12 Basis Data

Menurut (Fauziah, 2019), *database* (dalam bahasa Indonesia berarti basis data) adalah kumpulan data yang disimpan secara sistematis di dalam komputer yang dapat diolah atau dimanipulasi menggunakan perangkat lunak (program aplikasi) untuk menghasilkan informasi. Pendefinisian basis data meliputi spesifikasi berupa tipe data, struktur data, dan juga batasan-batasan pada data yang kemudian disimpan.

Menurut (Simarmata & Paryudi, 2006), *database* merupakan kumpulan data berisi informasi yang sesuai untuk sebuah perusahaan. Sistem manajemen *database* (DBMS) adalah kumpulan data yang saling berhubungan dan kumpulan program untuk mengakses data. Tujuan utama sistem manajemen basis data adalah menyediakan cara menyimpan dan mengambil informasi basis data secara mudah dan efisien.

2.2.13 SQLite

Android memiliki beberapa teknik untuk melakukan penyimpanan data. Teknik yang umum yang digunakan adalah sebagai berikut:

- *Shared Preferences*, yaitu teknik menyimpan data beberapa nilai (value) dalam bentuk *groups key* yang dikenal dengan *preferences*.
- *Files*, yaitu teknik menyimpan data dalam *file*, berupa menulis dan membaca *file*.
- *Content Providers*, yaitu menyimpan data dalam bentuk *content providers service*.
- SQLite, yaitu menyimpan data dalam bentuk *Database*.

Menurut (Setiawan, 2007), SQLite adalah RDBMS (Sistem Manajemen Basis Data Relasional) alternatif yang bersifat portable (tidak memerlukan proses instalasi), cepat, gratis, dan didukung oleh banyak bahasa pemrograman.

Menurut (Juhara, 2016), SQLite adalah sistem manajemen basis data ringan yang tidak mengadopsi model klien-server, serta cocok diterapkan pada perangkat dengan sumber daya terbatas seperti ponsel. SQLite dirancang sebagai pustaka yang berdiri sendiri dengan ketergantungan pada pustaka lain seminimal mungkin.

SQLite sendiri merupakan gabungan kata dari SQL (*Structured Query Language*) dan Lite, yang artinya bahasa ringan yang digunakan untuk mengelola data pada RDBMS.

2.2.14 Eclipse ADT *Bundle*

Menurut (Safaat H., 2015) Android *Development Tools* (ADT) adalah *plugin* yang didesain untuk Eclipse IDE yang memberikan kita kemudahan dalam mengembangkan aplikasi android dengan menggunakan Eclipse IDE. Sedangkan menurut Juhara (Juhara, 2016), Eclipse ADT adalah *plugin* Eclipse IDE (*Integrated Development Environtment*) yang menambahkan fungsionalitas pengembangan aplikasi Android di Eclipse IDE.

Eclipse ADT *Bundle* merupakan *bundle* program yang berisi *plugin* Eclipse ADT yang terpisah dari program Eclipse IDE-nya itu sendiri, yang mana dapat dipasang tanpa terlebih dahulu memasang Eclipse IDE. Dengan menggunakan ADT untuk Eclipse akan memudahkan kita dalam membuat aplikasi project android, membuat GUI aplikasi, dan menambahkan komponen-komponen yang lainnya, begitu juga kita dapat melakukan *running* aplikasi menggunakan Android SDK melalui Eclipse. Dengan ADT juga kita dapat melakukan pembuatan package android (.apk) yang digunakan untuk distribusi aplikasi android yang kita rancang.

Semakin tinggi *platform* Android yang akan digunakan, maka dianjurkan untuk menggunakan ADT yang lebih terbaru, karena biasanya munculnya *platform* baru diikuti oleh munculnya versi ADT yang terbaru. Namun sayangnya kini ADT Eclipse tidak lagi diakusisi oleh Google

sebagai IDE resmi untuk mengembangkan aplikasi Android dan digantikan oleh Android Studio, sehingga pengembangan ADT Eclipse dihentikan. Meskipun begitu, ADT Eclipse sangat disarankan digunakan untuk mengembangkan aplikasi berbasis Android pada spesifikasi komputer yang rendah karena keringanannya. ADT Eclipse yang digunakan oleh peneliti adalah Eclipse versi Juno yang rilis pada tanggal 2 Juli 2014.

2.2.15 Nox App Player

Berdasarkan situs (Jagad.id, n.d.), Nox App Player adalah Android emulator gratis yang didedikasikan untuk memberikan pengalaman terbaik bagi pengguna dalam memainkan game dan aplikasi Android menggunakan PC atau Mac yang dikembangkan oleh Nox Digital Entertainment Co. Limited yang berasal dari Hongkong.

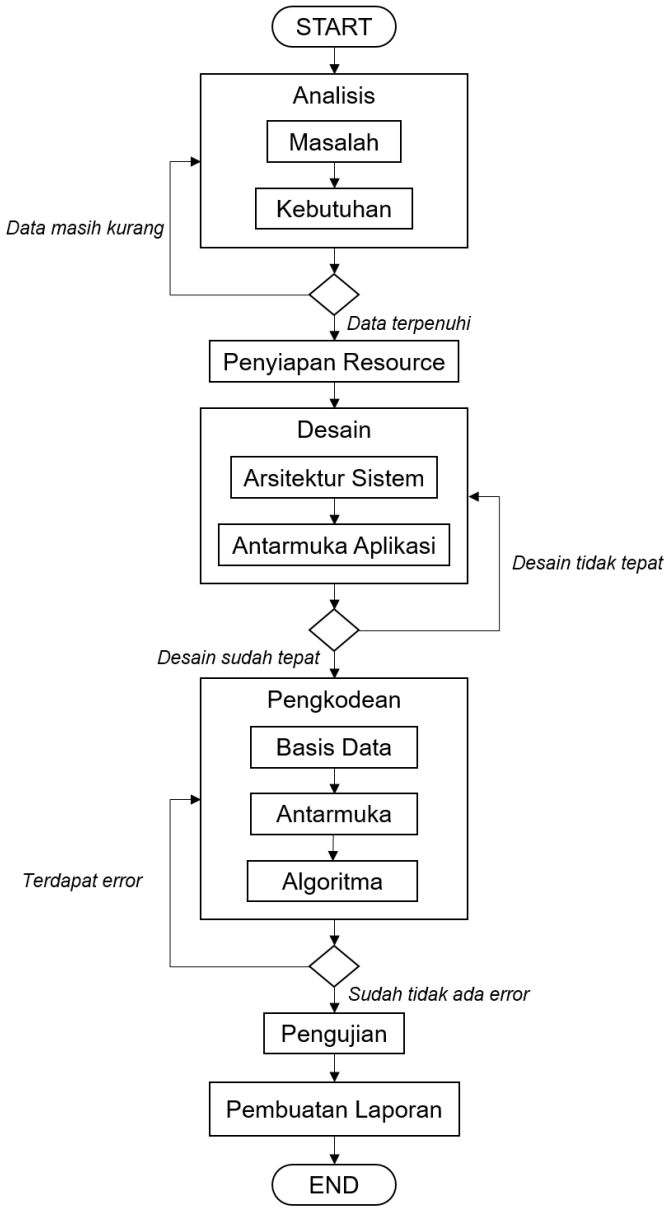
Menurut (Juhara, 2016), Android emulator itu sendiri adalah aplikasi komputer yang meniru cara kerja perangkat Android sesungguhnya, yang mana hampir sebagian fitur-fitur yang ada di perangkat Android dapat ditemukan pada emulator, kecuali beberapa fitur perangkat keras seperti GPS, kamera, atau sensor.

BAB III

METODOLOGI

3.1 Kerangka Pikir

Berikut adalah langkah-langkah yang dilakukan untuk mencapai tujuan dari penelitian, ditunjukan pada Gambar 3.1 dibawah ini:



Gambar 3.1 Kerangka Pikir Model *Waterfall*

3.2 Deskripsi

Berdasarkan dari kerangka pikir sebelumnya, maka diuraikan langkah-langkah dari pengerjaan penelitian ini sebagai berikut:

3.2.1 Analisis

Tahap analisis merupakan tahap pertama pada penelitian yang dilaksanakan di Kantor Kecamatan Ciparay. Berikut adalah penjelasan pada tiap tahapannya:

1. Analisis Masalah

Analisis masalah dilakukan untuk mencari permasalahan yang berada di tempat penelitian serta jawaban atas masalah tersebut, metode pengumpulan data yang digunakan pada tahap ini adalah metode observasi, dan pemberian kuesioner kepada responden. Observasi dilakukan dengan cara mengamati sistem yang berjalan di Kantor Kecamatan Ciparay pada bagian arsiparis, sedangkan kuesioner dilakukan untuk mengetahui kebutuhan terkait keperluan dan spesifikasi aplikasi yang dibutuhkan oleh arsiparis. Berikut adalah hasil dari tahap analisis masalah ini:

a) Arsiparis membutuhkan aplikasi yang dapat mencari klasifikasi kode arsip, karena pencarian kode yang dilakukan oleh arsiparis masih manual sehingga proses pencarian memakan waktu yang tidak sebentar.
b) Aplikasi yang dibutuhkan adalah aplikasi berbasis Android, karena aplikasi pencarian klasifikasi kode arsip ini harus mendukung konsep mobilitas, yaitu mudah diakses dan dapat dibawa kemana-mana.
c) Aplikasi membutuhkan algoritma pencocokan *string*, mengingat bahwa jumlah indeks dalam klasifikasi kode arsip tidak sedikit.
d) Algoritma pencocokan *string* yang perlu dipakai adalah algoritma Boyer-Moore, karena dalam praktiknya algoritma ini dianggap memiliki hasil paling baik.

2. Analisis Kebutuhan

Analisis kebutuhan dilaksanakan ketika tahap sebelumnya yaitu tahap analisis masalah selesai dilaksanakan, tahap ini merupakan tahapan lanjutan dari tahap analisis, yaitu tahap yang dilakukan untuk mengetahui apa saja peralatan yang dibutuhkan terkait penyelesaian masalah yang ada di tempat penelitian, metode pengumpulan data yang digunakan adalah studi pustaka. Studi pustaka dilakukan sebagai referensi untuk mengambil tindakan terhadap penyelesaian masalah dan pembangunan aplikasi dengan cara melihat referensi pada artikel. Studi pustaka tersebut menghasilkan kebutuhan fungsional serta perangkat lunak yang dibutuhkan untuk membangun aplikasi, berikut adalah hasilnya:

Tabel 3.1 Kebutuhan Fungsional

No	Kebutuhan Fungsional	Keterangan
1	Halaman Utama	Halaman ini digunakan untuk menampilkan informasi seputar aplikasi.
2	Pencarian	Halaman pencarian digunakan untuk mencari klasifikasi berdasarkan kode ataupun namanya.
3	Detail Pencarian	Halaman untuk menampilkan detail dari klasifikasi surat yang dicari.
3	Materi Struktur Nomor Surat	Halaman ini digunakan untuk menampilkan materi tentang struktur pada nomor surat dinas resmi.

4	Tentang Pengembang	Halaman yang menampilkan informasi tentang pengembang aplikasi.
5	Menu Navigasi	Menu yang digunakan untuk berpindah antar halaman.

Dibawah ini merupakan perangkat lunak yang dibutuhkan untuk membangun aplikasi Klasifikasi Kode Surat:

a) Eclipse ADT Bundle.
b) Android SDK Manager.
c) DB Browser (SQLite).
d) Java JDK.
e) Nox App Player.
f) StarUML.
g) Balsamiq MockUp.

Sedangkan perangkat keras yang dibutuhkan adalah laptop yang digunakan untuk membangun aplikasi, dan *smartphone* untuk menjalankan aplikasi yang dibangun. Berikut adalah masing-masing spesifikasi minimal perangkat keras tersebut:

a) Laptop
   - Sistem Operasi dan Arsitekturnya: Windows 7, 32-bit
   - Processor: *Dual-Core*, 2 Ghz
   - RAM: 1.5 GB
   - Harddisk (kapasitas kosong): 4 GB

b) *Smartphone*
   - Sistem Operasi: Android KitKat 4.4 (API level 19)
   - Processor: *Single-Core*, 1.2 Ghz
   - RAM: 500 MB

3.2.2 Penyiapan *Resource*

Penyiapan *resource* dilakukan ketika hasil pada tahap analisis kebutuhan telah diketahui. Tahap ini dilakukan dengan cara melakukan instalasi perangkat lunak pada alat perangkat keras yang digunakan dalam penelitian agar tahap desain dan pembangunan aplikasi Klasifikasi Kode Surat dapat dilakukan.

3.2.3 Desain

Tahap desain dilakukan untuk memodelkan/merancang kebutuhan yang telah dianalisis sebelumnya, dengan menggunakan peralatan yang telah disiapkan pada tahap penyiapan *resource*. Berikut adalah masing-masing penjelasan pada tiap tahapannya:

1. Arsitektur Sistem

Tahap ini dilakukan dengan membuat desain menggunakan bahasa pemodelan *Unified Modelling Language* (UML). UML yang yang digunakan dalam penelitian ini terdiri dari:

a) *Use case diagram*, untuk mengetahui gambaran umum fungsi apa saja yang ada di dalam sistem aplikasi.
b) *Class diagram*, untuk menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem.
c) *Activity diagram*, untuk memodelkan aktifitas yang ada dalam suatu sistem meliputi gambaran keseluruhan aktifitas yang dilakukan oleh pengguna dengan aplikasi.
d) *Sequence diagram*, menggambarkan kelakukan objek pada use case dengan mendeskripsikan pesan yang dikirimkan dan diterima antar objek. Diagram ini memiliki jumlah sebanyak 5 buah diagram, jumlah ini merupakan gambaran total interkasi yang dapat dilakukan oleh pengguna dengan aplikasi.

Berikut adalah interkasi yang dapat dilakukan pengguna dalam s*equence diagram*:

1) Pengguna melakukan pencarian klasifikasi kode surat berdasarkan kode klasifikasinya
2) Pengguna melakukan pencarian klasifikasi kode surat berdasarkan nama klasifikasinya.
3) Pengguna melakukan akses detail tentang klasifikasi kode surat.
4) Pengguna melakukan akses kepada materi tentang struktur nomor surat.
5) Pengguna melakukan akses kepada halaman yang menampilkan informasi tentang pengembang aplikasi.

Perangkat lunak yang digunakan untuk merancang UML tersebut adalah perangkat lunak StarUML. Desain basis data juga dilakukan pada tahap ini.

2. Antarmuka Aplikasi

Desain antarmuka pada aplikasi dibuat sederhana, karena desain yang sederhana akan membuat penyampaian pesan lebih cepat daripada ilustrasi yang kompleks dan detail, tentunya desain tersebut juga diperhatikan dari sisi interaksi manusia komputer yang akan terjadi dalam aplikasi. Alat yang digunakan dalam tahap ini adalah perangkat lunak Balsamiq Mockup dan Microsoft Paint.

Aplikasi memiliki desain antarmuka sebanyak 7 buah antarmuka, yaitu:

1) Halaman Utama
2) Halaman Pencarian Berdasarkan Kode
3) Halaman Pencarian Berdasarkan Nama
4) Halaman Detail Surat
5) Halaman Menu Navigasi
6) Halaman Struktur Nomor Surat
7) Halaman Tentang Pengembang

3.2.4 Pengkodean

Tahap ini merupakan tahap implementasi desain pada tahap sebelumnya, implementasi dilakukan dengan cara pengkodean pada algoritma yang telah dipilih untuk mengatasi pencocokan *string*, pengkodean fungsi atau prosedur, pengkodean antarmuka aplikasi, dan pengkodean basis data. Berikut adalah penjelasan masing-masing tahapan pengkodeannya:

1. **Basis Data**

Pengkodean basis data dilakukan dengan menggunakan DBMS SQLite di Eclipse ADT dengan mengacu kepada desain yang telah dibuat pada tahap desain arsitektur sistem.

2. **Antarmuka**

Tahap pengkodean antarmuka dilakukan dengan menggunakan tag script XML pada Eclipse ADT, tahap ini dilakukan berdasarkan desain antarmuka aplikasi yang sebelumnya telah dibuat pada tahap desain.

3. **Algoritma**

Tahap ini dilakukan untuk mengimplementasikan algoritma Boyer-Moore, pengkodean fungsi atau prosedur, dan hal lainnya yang berkaitan dengan proses yang terjadi dalam aplikasi Klasifikasi Kode Surat menggunakan bahasa pemrograman Java pada Eclipse ADT.

Secara keseluruhan, tujuan dari tahap pengkodean ini adalah untuk menghasilkan paket *file* aplikasi Klasifikasi Kode Surat (.apk). Pada tahap ini juga dilakukan pengujian unit dan pengujian integrasi untuk mengetahui ada tidaknya *error* pada aplikasi. Jika terdapat *bug*, *error*, atau kekurangan sintaks dan lojik pada kode program yang menyebabkan aplikasi tidak dapat di-*build* dan *running*, maka tahap selanjutnya yaitu tahap pengujian belum bisa dilakukan.

3.2.5 Pengujian

Tahap pengujian ini merupakan tahap pengujian sistem yang dilakukan untuk mengetahui kualitas aplikasi Klasifikasi Kode Surat dengan cara menguji aplikasi dari segi spesifikasi fungsional tanpa menguji desain dan kode program (*black-box testing*). *Black-box testing* dilakukan setelah tahap pengkodean selesai dan aplikasi selesai di-*build*, yang tujuannya untuk mencari:

- Fungsi yang salah atau hilang.
- Kesalahan dalam struktur data atau akses basis data.
- Kesalahan antarmuka dan kinerja.
- Validasi fungsional.
- Kesensitifan sistem terhadap nilai input tertentu.
- Batasan suatu data.

Setelah pengujian *black-box* selesai dilakukan, peneliti melakukan tahap pengujian penerimaan oleh pengguna (arsiparis) untuk mengetahui tingkat kepuasan terhadap aplikasi Klasifikasi Kode Surat. Metode pengumpulan data yang digunakan pada tahap ini adalah kuesioner.

Tahap ini menghasilkan tingkat kelayakan pada aplikasi Klasifikasi Kode Surat, tingkatan kelayakan aplikasi dinilai oleh responden berdasarkan 3 kategori penilaian akhir terhadap aplikasi, yaitu:

- Layak digunakan tanpa revisi
- Layak digunakan dengan revisi sesuai saran pengguna
- Tidak layak

3.2.6 Pembuatan Laporan

Tahap pembuatan laporan merupakan tahap terakhir dalam penelitian yang dilakukan. Tahap ini menghasilkan dokumen Laporan Skripsi dimulai dari Bab I dan diakhiri dengan Bab VI. Sistematika penulisan laporan dapat dilihat pada Subbab 1.6.

BAB IV

ANALISIS DAN PERANCANGAN

4.1 Analisis

Analisis dilakukan untuk mengetahui permasalahan yang dihadapi oleh instansi serta solusinya yang mendefinisikan kebutuhan aplikasi yang dibangun.

4.1.1 Instrumen Penelitian

Instrumen penelitian merupakan alat bantu yang digunakan peneliti untuk mengumpulkan dan mengolah data pada tahap analisis. Instrumen yang digunakan dalam penelitian ini meliputi perangkat keras, perangkat lunak, dan formulir kuisioner.

1. Instrumen Perangkat Keras

Perangkat keras digunakan untuk mengumpulkan data khususnya pada saat pencatatan terkait hasil observasi, serta hasil analisis yang nantinya digabungkan dalam laporan.

Tabel 4.1 Instumen Perangkat Keras

Laptop	
Merek dan Tipe	Lenovo, G400
Sistem Operasi dan Arsitekturnya	Windows 10, 64-bit
Processor	*Dual-Core* Pentium, 2.40 Ghz
Kapasitas RAM dan Harddisk	2 GB, 500 GB
Smartphone	
Merek dan Tipe	Xiaomi, Redmi 3
Sistem Operasi	Android, Lollipop 5.1.1
Kapasitas RAM	2 GB

2. Instrumen Perangkat Lunak

Instrumen ini merupakan perangkat lunak yang menjadi penunjang dalam penelitian dan telah terpasang pada instrumen perangkat keras, berikut ini adalah beberapa perangkat lunak yang digunakan:

1) Google Chrome, merupakan *browser* yang digunakan untuk mengumpulkan data dengan cara observasi dan studi pustaka tentang penelitian yang dilakukan melalui sumber dari internet, seperti artikel dan jurmal.
2) Microsoft Word 2016, aplikasi yang digunakan untuk membuat dokumen kuesioner, dan dokumen lain yang menunjang dalam penelitian ini.
3) Mendeley Desktop, aplikasi yang digunakan sebagai pencatatan sitasi jurnal, buku, artikel, dan dokumen lain yang dikumpulkan dan digunakan dalam penelitian ini.
4) Catatan dan Notepad, aplikasi Catatan terpasang pada *smartphone* sedangkan Notepad yang merupakan aplikasi pada windows, keduanya digunakan untuk mencatat data-data sementara yang terkumpul dengan metode observasi.

3. Formulir Kuesioner

Formulir isian kuesioner digunakan untuk mendapatkan data terkait keperluan terhadap aplikasi Klasifikasi Kode Arsip. Formulir tersebut berbentuk dokumen 1 lembar yang dibagikan secara langsung kepada subjek penelitian atau orang yang memegang jabatan arsiparis di Kantor Kecamatan Ciparay.

Berdasarkan hasil dari kuesioner tersebut, narasumber membutuhkan aplikasi yang dapat mencari klasifikasi kode surat berbasis Android (hasil kuesioner dapat dilihat pada lampiran 1, dengan judul kuesioner keperluan aplikasi).

4.1.2 Analisis Masalah

Analisis yang dilakukan pada tahap ini adalah analisis terkait pengelolaan surat masuk dan keluar yang dilakukan oleh petugas kearsipan. Untuk mengetahui alur pengelolaan surat masuk, peneliti menggunakan metode pengumpulan data dengan cara melakukan observasi, dan studi pustaka.

Berdasarkan kegiatan penanganan surat masuk dan keluar yang dilakukan oleh petugas kearsipan, peneliti selanjutnya melakukan analisis dan didapatkan tahap yang tidak efisien, yaitu tahap pencarian klasifikasi kode surat yang masih dilakukan secara manual dengan cara melihat pada buku atau daftar klasifikasi kode arsip. Setelah masalah inti ditemukan, peneliti lalu melakukan pengembangan terhadap masalah tersebut dengan cara melakukan studi pustaka dan didapatkan hasil tehadap analisis masalah yang lebih lengkap, yaitu sebagai berikut:

1) Petugas kearsipan membutuhkan aplikasi yang dapat mencari klasifikasi kode arsip, karena pencarian kode yang dilakukan oleh arsiparis masih manual sehingga proses pencarian memakan waktu yang tidak sebentar.
2) Aplikasi yang dibutuhkan adalah aplikasi berbasis Android, karena aplikasi pencarian klasifikasi kode arsip ini harus mendukung konsep mobilitas, yaitu mudah diakses dan dapat dibawa kemana-mana.
3) Aplikasi membutuhkan algoritma pencocokan *string*, mengingat bahwa jumlah indeks dalam klasifikasi kode arsip tidak sedikit.
4) Algoritma pencocokan *string* yang perlu dipakai adalah algoritma Boyer-Moore, karena dalam praktiknya algoritma ini dianggap memiliki hasil paling baik.

4.1.3 Analisis Kebutuhan

Setelah data pada tahap analisis masalah telah selesai dikumpulkan, peneliti kemudian melakukan analisis kebutuhan terhadap aplikasi yang akan dibangun berdasarkan kebutuhan yang dapat memenuhi proses pembuatan aplikasi, yang meliputi kebutuhan fungsional, kebutuhan perangkat lunak, dan kebutuhan perangkat keras.

1. Kebutuhan Fungsional

Kebutuhan fungsional merupakan kebutuhan dari segi fungsionalitas yang akan muncul pada aplikasi yang dirancang. Adapun beberapa kebutuhan fungsional pada aplikasi Klasifikasi Kode Surat ini adalah sebagai berikut:

1) Halaman Utama, Halaman ini digunakan untuk menampilkan informasi seputar aplikasi, dan merupakan halaman yang pertama kali dibuka oleh aplikasi (halaman beranda).
2) Pencarian Klasifikasi, Halaman pencarian digunakan untuk mencari klasifikasi kode surat. Pencarian klasifikasi kode surat akan memiliki 2 opsi pilihan, yang pertama adalah pencarian klasifikasi berdasarkan kode surat, dan yang kedua adalah pencarian klasifikasi berdasarkan nama surat.
3) Detail Pencarian, Halaman yang digunakan untuk menampilkan detail dari klasifikasi surat yang dicari.
4) Materi Struktur Nomor Surat, Halaman ini digunakan untuk menampilkan materi tentang struktur penulisan nomor pada surat dinas resmi.
5) Tentang Pengembang, Halaman yang menampilkan informasi tentang pengembang aplikasi.
6) Menu Navigasi, Menu yang digunakan untuk berpindah antar halaman induk, seperti Halaman Utama, halaman Struktur Materi Nomor Surat, dan halaman Tentang Pengembang.

2. Kebutuhan Perangkat Lunak

Dari hasil analisis yang telah dilakukan dengan cara observasi dari internet dan studi pustaka dari jurnal, didapatkan bahwa untuk membangun aplikasi Klasifikasi Kode Surat, dibutuhkan beberapa perangkat lunak sebagai berikut:

a. Perangkat Utama
   1) Eclipse ADT Bundle, merupakan *bundle* pada Eclipse IDE yang digunakan untuk membangun aplikasi berbasis Android. Versi yang digunakan adalah versi rilis tanggal 2 Juli 2014 (Juno).
   2) Android SDK Manager, aplikasi bawaan dari Eclipse ADT Bundle yang digunakan untuk memanajemen Android SDK.
   3) DB Browser (SQLite), merupakan aplikasi yang digunakan untuk membuat dan membuka basis data sqlite. Versi yang digunakan adalah 3.12.1.
   4) Java JDK, merupakan perangkat lunak yang dibutuhkan untuk membangun aplikasi berbasis Android di Eclipse IDE. Versi yang digunakan adalah versi 8 update 102.

b. Perangkat Pendukung
   1) Nox App Player, simulator yang digunakan untuk menjalankan aplikasi Android pada perangkat *desktop*, dan simulator ini digunakan juga untuk keperluan observasi sistem penyimpanan basis data pada Android, yang mana jika dilakukan pada perangkat *smartphone* yang sebenarnya, *smartphone* tersebut harus di *root* terlebih dahulu agar sistem penyimpanan basis data pada direktori sistem Android dapat dilihat. Versi yang digunakan adalah 5.1.0.1 bersistem operasi Android 4.4.2.
   2) StarUML, digunakan untuk memodelkan rancangan UML dan *flowchart*. Versi yang digunakan adalah 2.6.0.
   3) Balsamiq MockUp, digunakan untuk memodelkan rancangan *user interface* aplikasi. Versi yang digunakan adalah 3.5.17.

3. Kebutuhan Perangkat Keras

Kebutuhan perangkat keras didapatkan dengan menggunakan metode observasi dan studi pustaka di internet, dan didapatkan data berupa spesifikasi minimum untuk membangun aplikasi dan menjalankannya.

a. Perangkat Untuk Membangun Aplikasi

Berikut ini adalah spesifikasi minimum yang digunakan untuk membangun aplikasi pencocokan *string* berbasis Android menggunakan Eclipse ADT Bundle.

Tabel 4.2 Spesifikasi Minimum *Hardware* untuk Membangun Aplikasi

Sistem Operasi dan Arsitekturnya	Windows 7, 32-bit
Processor	*Dual-Core*, 2 Ghz
RAM	1.5 GB
Harddisk (kapasitas kosong)	4 GB

Jika dibandingkan dengan spesifikasi perangkat keras yang digunakan peneliti pada tabel 4.1, dapat disimpulkan bahwa spesifikasi perangkat keras yang digunakan peneliti memenuhi spesifikasi minimum perangkat keras untuk membangun aplikasi. Oleh karena itu, instrumen *hardware* yang digunakan peneliti sebelumnya digunakan kembali pada tahap merancang dan membangun aplikasi.

b. Perangkat Untuk Menjalankan Aplikasi

Perangkat keras yang dapat digunakan untuk menjalankan aplikasi Klasifikasi Kode Surat ini yaitu *smartphone*/tablet berbasis Android yang mempunyai spesifikasi minimum sebagai berikut:

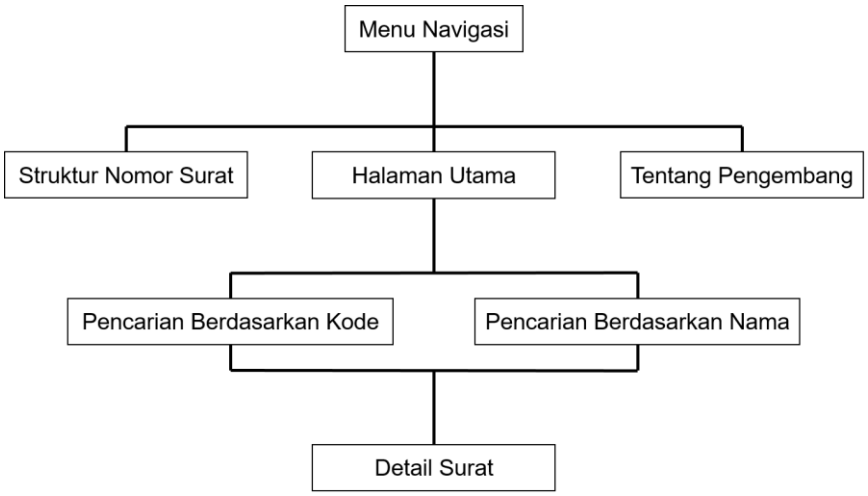
- Sistem Operasi: Android, KitKat 4.4 (API level 19)
- Processor: *Single-Core*, 1.2 Ghz
- RAM: 500 MB

4.2 Perancangan

Perancangan proses aplikasi ini digunakan *Unified Modelling Language* pada perangkat lunak StarUML yang terdiri dari *use case diagram, class diagram, activity diagram, dan sequence diagram*. Sedangkan perancangan *user interface* aplikasi digunakan perangkat lunak Balsamiq Mockups.

4.2.1 Struktur Menu

Struktur menu disusun secara bertingkat agar dapat memudahkan pengguna dalam pengoperasian aplikasi. Berikut adalah struktur menu dari aplikasi yang akan dibangun.



Gambar 4.1 Perancangan Struktur Menu

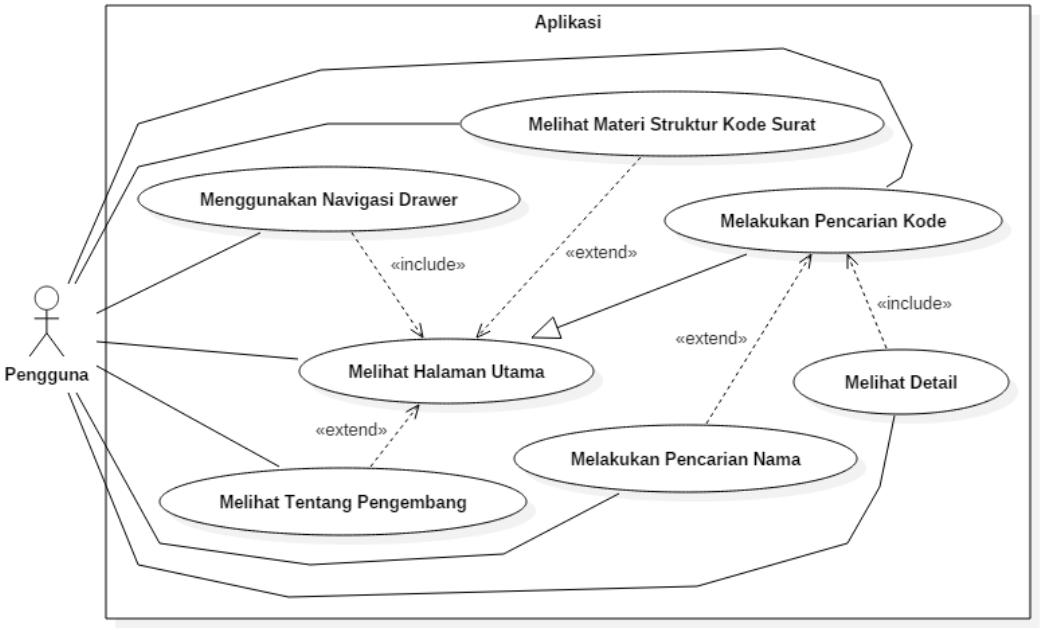
Berikut ini adalah penjelasan dari stuktur menu aplikasi:

1) Menu Navigasi, menu ini dibuat untuk mempermudah perpindahan antar halaman induk, yaitu Halaman Utama, halaman Stuktur Kode Surat, dan halaman Tentang Pengembang.
2) Halaman Utama, halaman ini dibuat untuk menampilkan informasi seputar aplikasi dan pada halaman ini pula terdapat dua tombol untuk berpindah kepada halaman pencarian klasifikasi kode surat, yaitu halaman Pencarian Berdasarkan Kode dan halaman Pencarian Berdasarkan Nama.

3) Stuktur Kode Surat, merupakan halaman yang digunakan untuk menampilkan materi sekilas tentang penggunaan kode surat pada surat dinas resmi di lingkungan pemerintahan.
4) Tentang Pengembang, halaman ini dibuat untuk menampilkan informasi tentang peneliti.
5) Pencarian Berdasarkan Kode, merupakan halaman yang dibuat dengan tujuan sebagai tempat pencarian klasifikasi kode surat berdasarkan kode klasifikasinya.
6) Pencarian Berdasarkan Nama, merupakan halaman yang dibuat dengan tujuan sebagai tempat pencarian klasifikasi kode surat berdasarkan nama klasifikasinya.
7) Detail Surat, halaman ini dibuat untuk menampilkan detail tentang klasifikasi kode surat yang dicari.

4.2.2 *Use Case Diagram*

Use case diagram dibuat untuk mengetahui gambaran fungsi apa saja yang ada di dalam sistem aplikasi. Berikut adalah *use case diagram* pada aplikasi Klasifikasi Kode Surat.



Gambar 4.2 Perancangan *Use Case Diagram*

1. Definisi Aktor

Berikut adalah deskripsi pendefinisian aktor pada aplikasi Klasifikasi Kode Surat:

Tabel 4.3 Definisi Aktor

Aktor	Deskripsi
Pengguna	Merupakan orang yang memegang jabatan sebagai arsiparis, pemegang jabatan lain yang merangkap menjadi arsiparis, maupun masyarakat umum. Pengguna dapat menggunakan aplikasi dan fiturnya secara keseluruhan tanpa batasan terkait hak akses.

2. Definisi *Use Case*

Berikut adalah deskripsi pendefinisian *use case* pada aplikasi Klasifikasi Kode Surat:

Tabel 4.4 Definisi *Use Case*

No	*Use Case*	Deskripsi
1	Melihat Halaman Utama	Merupakan proses menampilkan Halaman Utama, *use case* ini merupakan proses generalisasi proses pencarian dibawahnya. *Use case* ini akan selalu memanggil proses Menggunakan Navigasi Drawer.
2	Menggunakan Navigasi Drawer	Merupakan proses untuk menampilkan halaman Navigasi Drawer.
3	Melihat Stuktur Nomor Surat	Merupakan proses menampilkan halaman Stuktur Nomor Surat. *Use case* ini mewarisi sifat proses Melihat Halaman Utama, maka *use case* ini juga akan selalu memanggil proses Menggunakan Navigasi Drawer.

4	Melihat Tentang Pengembang	Merupakan proses menampilkan halaman Tentang Pengembang. *Use case* ini mewarisi sifat proses Melihat Halaman Utama, maka *use case* ini juga akan selalu memanggil proses Menggunakan Navigasi Drawer.
5	Melakukan Pencarian Kode	Merupakan proses menampilkan halaman Pencarian Kode, yaitu pencarian klasifikasi kode surat berdasarkan kode klasifikasinya. *Use case* ini akan selalu memanggil proses Melihat Detail.
6	Melakukan Pencarian Nama	Merupakan proses menampilkan halaman Pencarian Nama, yaitu pencarian klasifikasi kode surat berdasarkan nama klasifikasinya *Use case* ini mewarisi sifat proses Melakukan Pencarian Kode, maka *use case* ini juga akan selalu memanggil proses Melihat Detail serta generalisasi terhadap proses Melihat Halaman Utama.
7	Melihat Detail	Merupakan proses yang dilakukan untuk menampilkan halaman Detail Surat, yaitu halaman yang menampilkan informasi klasifikasi kode surat yang dicari. *Use case* ini akan digunakan oleh proses Melakukan Pencarian Kode, dan karena proses Melakukan Pencarian Nama mewarisi proses Melakukan Pencarian Kode, maka *use case* Melihat Detail juga akan digunakan oleh proses Melakukan Pencarian Nama.

3. Skenario *Use Case*

Berikut adalah skenario jalannya masing-masing *use case* yang telah didefinisikan sebelumnya:

1) Skenario *Use case*: Melihat Halaman Utama

Tabel 4.5 Skenario *Use Case*: Melihat Halaman Utama

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Membuka aplikasi	
	2. Menampilkan Halaman Utama
Skenario Alternatif	
1. Menekan menu Halaman Utama	
	2. Menampilkan Halaman Utama

2) Skenario *Use case*: Menggunakan Navigasi Drawer

Tabel 4.6 Skenario *Use Case*: Menggunakan Navigasi Drawer

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Menekan tombol Navigasi	
	2. Menampilkan menu Navigasi Drawer
Skenario Alternatif	
1. Mengusap layar dari pojok kiri ke kanan	
	2. Menampilkan Navigasi Drawer

3) Skenario *Use case*: Melihat Stuktur Nomor Surat

Tabel 4.7 Skenario *Use Case*: Melihat Stuktur Nomor Surat

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Menekan menu Stuktur Nomor Surat	
	2. Menampilkan halaman Stuktur Nomor Surat

4) Skenario *Use case*: Melihat Tentang Pengembang

Tabel 4.8 Skenario *Use Case*: Melihat Tentang Pengembang

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Menekan menu Tentang Pengembang	
	2. Menampilkan halaman Tentang Pengembang

5) Skenario *Use case*: Melakukan Pencarian Kode

Tabel 4.9 Skenario *Use Case*: Melakukan Pencarian Kode

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Menekan tombol Pencarian Berdasarkan Kode	
	2. Menampilkan halaman Pencarian Kode

3. Memasukan kode surat yang ingin dicari	
	4. Mencocokan kode dengan yang ada pada basis data
	5. Menampilkan kode yang dicari
Skenario Alternatif	
1. Menekan tombol Pencarian Berdasarkan Kode	
	2. Menampilkan halaman Pencarian Kode
3. Memasukan kode surat yang ingin dicari	
	4. Mencocokan kode dengan yang ada pada basis data
	5. Menampilkan pesan kode tidak ditemukan

6) Skenario *Use case*: Melakukan Pencarian Nama

Tabel 4.10 Skenario *Use Case*: Melakukan Pencarian Nama

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Menekan tombol Pencarian Berdasarkan Nama	
	2. Menampilkan halaman Pencarian Nama

3. Memasukan nama surat yang ingin dicari	
	4. Mencocokan kode dengan yang ada pada basis data
	5. Menampilkan kode yang dicari
Skenario Alternatif	
1. Menekan tombol Pencarian Berdasarkan Nama	
	2. Menampilkan halaman Pencarian Nama
3. Memasukan nama surat yang ingin dicari	
	4. Mencocokan nama dengan yang ada pada basis data
	5. Menampilkan pesan kode tidak ditemukan

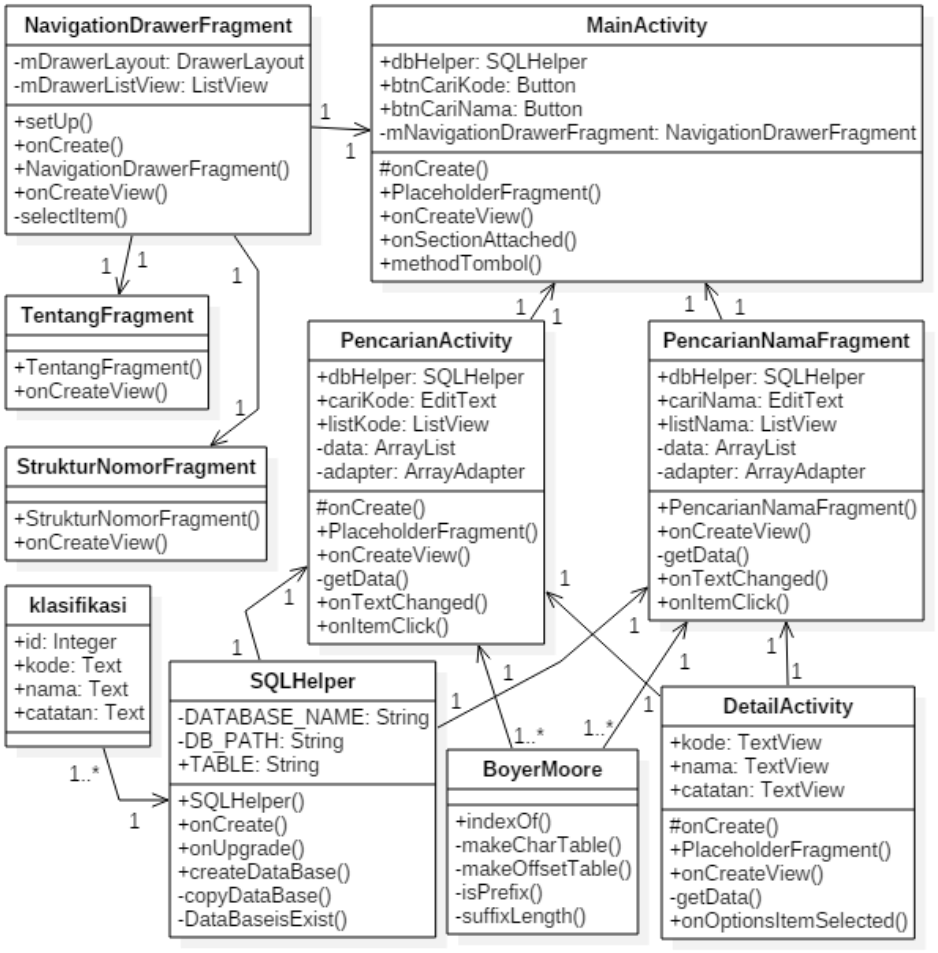
7) Skenario *Use case*: Melihat Detail

Tabel 4.11 Skenario *Use Case*: Melihat Detail

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Menekan data klasifikasi kode surat yang tersedia	
	2. Menampilkan halaman Detail Surat

4.2.3 *Class Diagram*

Class diagram dibuat untuk menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun aplikasi.



Gambar 4.3 Perancangan *Class Diagram*

Berikut adalah penjelasan masing-masing dari perancangan kelas yang dibuat:

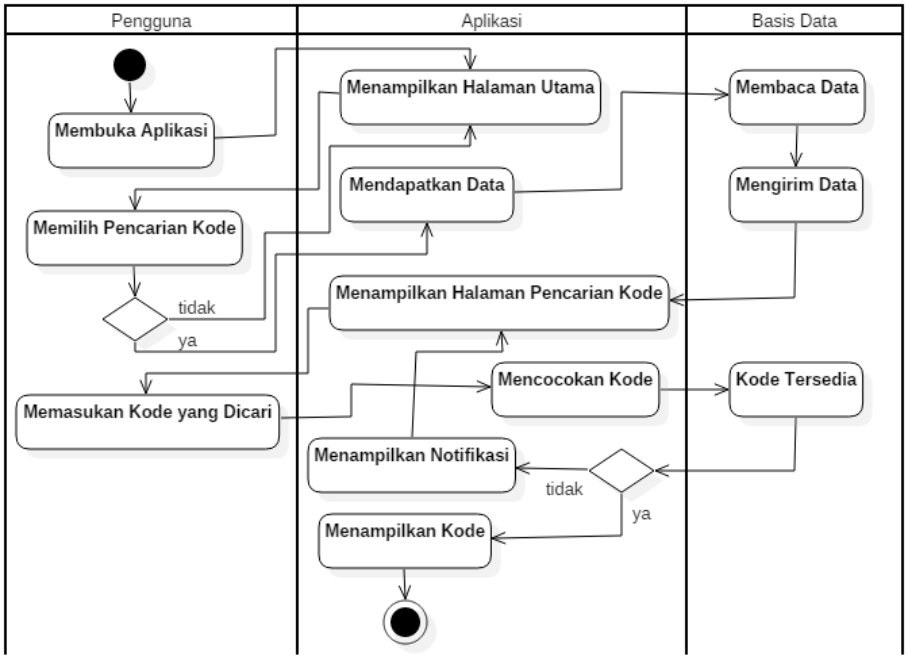
1) NavigationDrawerFragment, pendefinisian kelas dari Menggunakan Navigasi Drawer.
2) MainActivity, pendefinisian kelas dari Melihat Halaman Utama.
3) TentangFragment, pendefinisian kelas dari Melihat Tentang Pengembang.
4) StrukturNomorFragment, pendefinisian kelas dari Melihat Stuktur Nomor Surat.

5) PencarianActivity, pendefinisian kelas Melakukan Pencarian Kode.
6) PencarianNamaFragment, pendefinisian kelas dari Melakukan Pencarian Nama.
7) DetailActivity, pendefinisian kelas dari Melihat Detail.
8) klasifikasi, kelas yang mendeskripsikan basis data.
9) SQLHelper, kelas utilitas untuk koneksi basis data.
10) BoyerMoore, kelas tempat *script* algoritma Boyer-Moore berada.

4.2.4 *Activity Diagram*

Activity diagram adalah model aktifitas yang ada dalam sistem. Pada aplikasi yang dibangun, *activity diagram* meliputi gambaran aktifitas pencarian dan pengkasesan halaman yang dilakukan oleh aplikasi.

1. *Activity Diagram* Pencarian Berdasarkan Kode

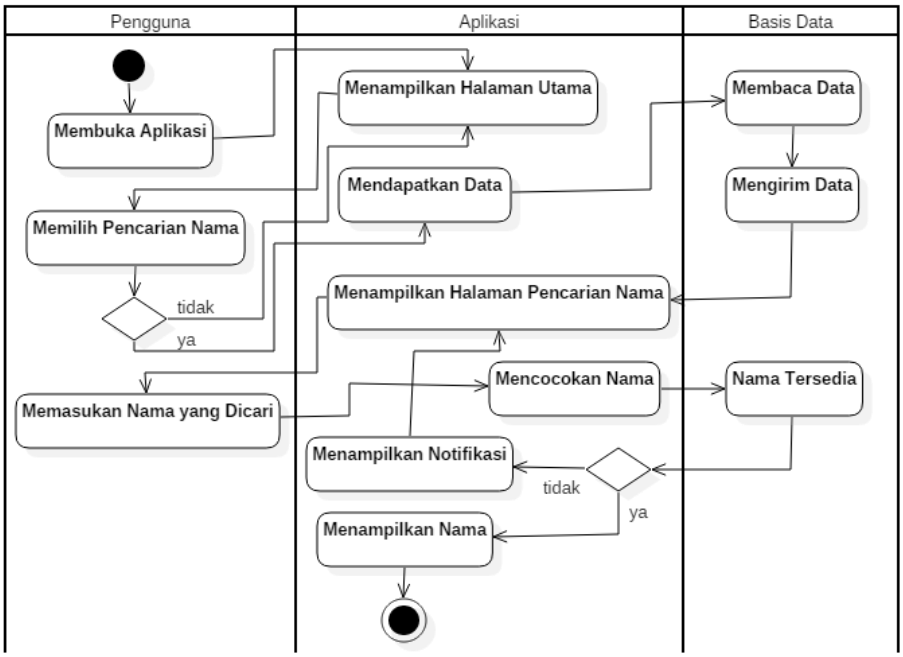


Gambar 4.4 Perancangan *Activity Diagram* Pencarian Berdasarkan Kode

Aktifitas dimulai saat aplikasi dibuka oleh pengguna, aplikasi akan menampilkan Halaman Utama. Pengguna dapat memilih tombol pencarian klasifikasi berdasarkan kode, yang mana jika ditekan akan mengganti

tampilan menjadi halaman Pencarian Berdasarkan Kode. Sebelum mengganti tampilan, aplikasi akan memanggil data pada basis data, setelah data dibaca dan dikirim, barulah tampilan *user interface* akan diperbaharui. Pengguna dapat mencari klasifikasi kode surat pada *Edit Text* yang tersedia. Ketika pengguna memasukan kode/*pattern* yang ingin dicari, aplikasi akan mencocokan *pattern* yang dimasukan dengan data yang ada, Jika kode yang dicari tersedia, aplikasi akan menampilkan kode surat tersebut, jika kode tidak ada, maka notifikasi bahwa data tidak ditemukan akan tampil.

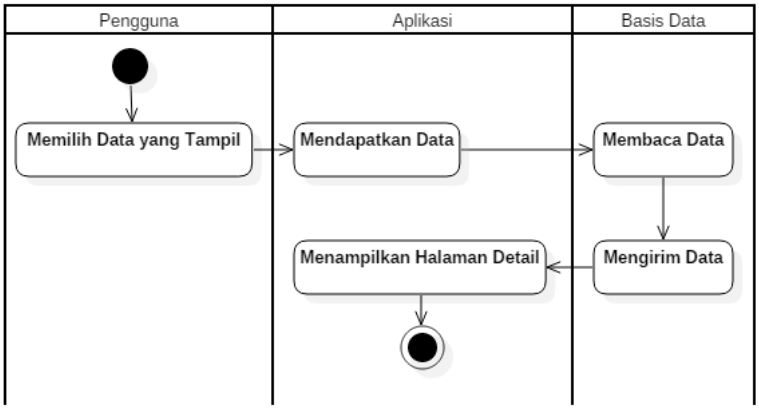
2. *Activity Diagram* Pencarian Berdasarkan Nama



Gambar 4.5 Perancangan *Activity Diagram* Pencarian Berdasarkan Nama

Sama halnya dengan *activity* pencarian berdasarkan kode, *activity* pencarian berdasarkan nama ini dimulai dari penampilan Halaman Utama. Pengguna dapat menekan tombol pencarian berdasarkan nama, dan halaman Pencarian Berdasarkan Nama akan muncul. Pencarian nama yang dilakukan pengguna memiliki sistem yang sama seperti *activity* sebelumnya, seperti pencocokan *pattern* dan penampilan ada tidaknya nama surat yang dicari tersebut.

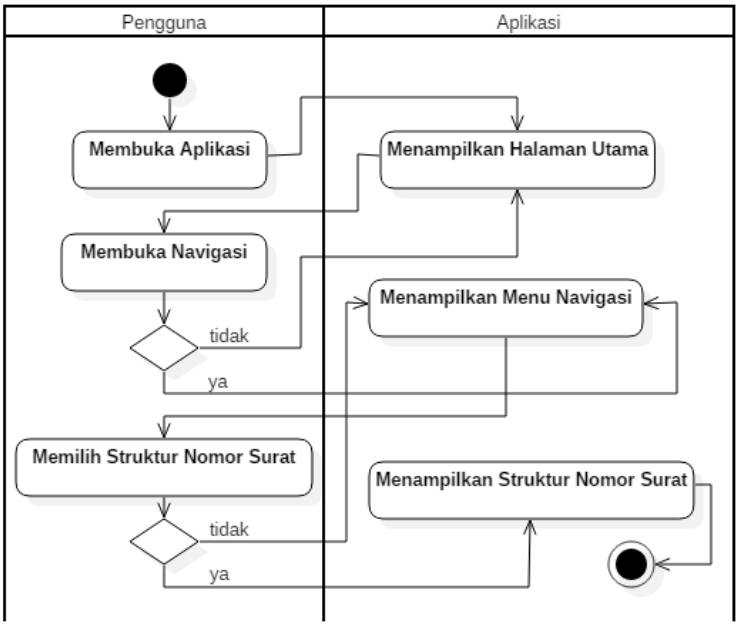
3. *Activity Diagram* Detail Surat



Gambar 4.6 Perancangan *Activity Diagram* Detail Surat

Aktifitas pada penampilan halaman Detail Surat dimulai saat pengguna menekan salah satu kode /nama surat yang tersedia pada *List View* di halaman Pencarian Berdasarkan Kode ataupun halaman Pencarian Berdasarkan Nama. Sebelum menampilkan halaman Detail, aplikasi akan membaca basis data dan mengirimnya, setelah itu barulah UI diperbaharui.

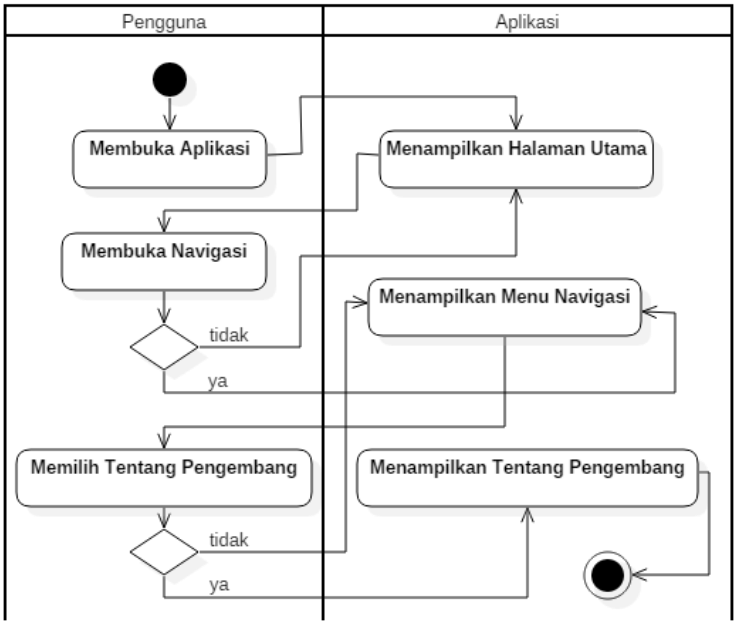
4. *Activity Diagram* Struktur Nomor Surat



Gambar 4.7 Perancangan *Activity Diagram* Struktur Nomor Surat

Aktifitas pada penampilan halaman Struktur Nomor Surat dimulai saat pengguna membuka aplikasi yang pertama kali akan menampilkan Halaman Utama. Pada Halaman Utama pengguna dapat membuka menu Navigasi untuk berpindah ke halaman Struktur Nomor Surat menggunakan tombol yang berada di posisi kiri atas (atau dengan melakukan *swipe* horizontal dari kiri ke tengah layar), setelah itu pengguna dapat memilih list Struktur Nomor Surat untuk menampilkan halaman tersebut.

5. *Activity Diagram* Tentang Pengembang



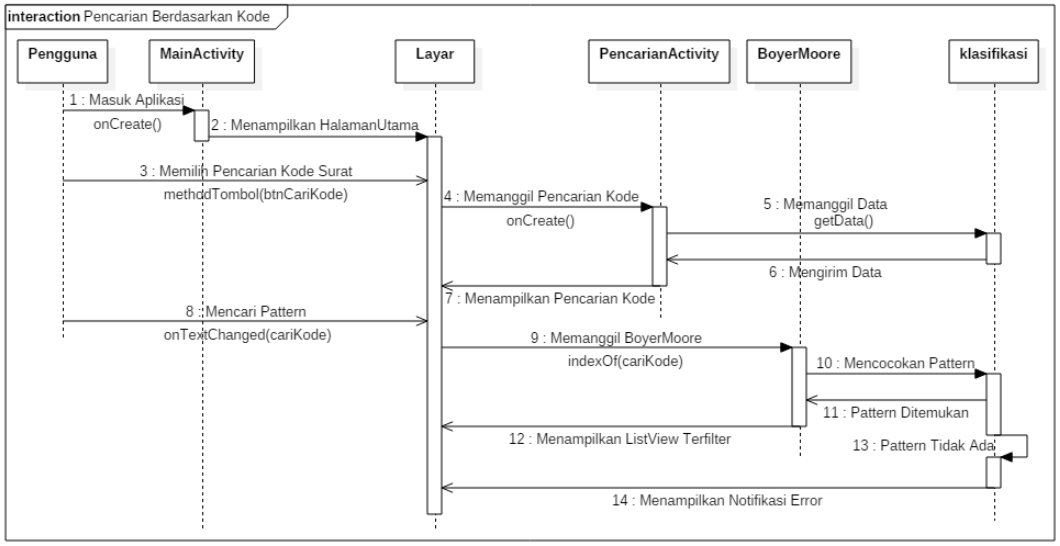
Gambar 4.8 Perancangan *Activity Diagram* Tentang Pengembang

Aktifitas pada penampilan halaman Tentang Pengembang tidak jauh berbeda dengan penampilan halaman Struktur Nomor Surat, yaitu dimulai dari saat pengguna membuka aplikasi yang pertama kali akan menampilkan Halaman Utama. Pada Halaman Utama pengguna dapat membuka menu Navigasi untuk berpindah ke halaman Tentang Pengembang menggunakan tombol yang berada di posisi kiri atas (atau dengan melakukan *swipe* horizontal dari kiri ke tengah layar), setelah itu pengguna dapat memilih list Struktur Nomor Surat untuk menampilkan halaman tersebut.

4.2.5 *Sequence Diagram*

Sequence diagram adalah diagram yang menggambarkan kelakukan objek pada use case dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek.

1. *Sequence Diagram* Pencarian Berdasarkan Kode



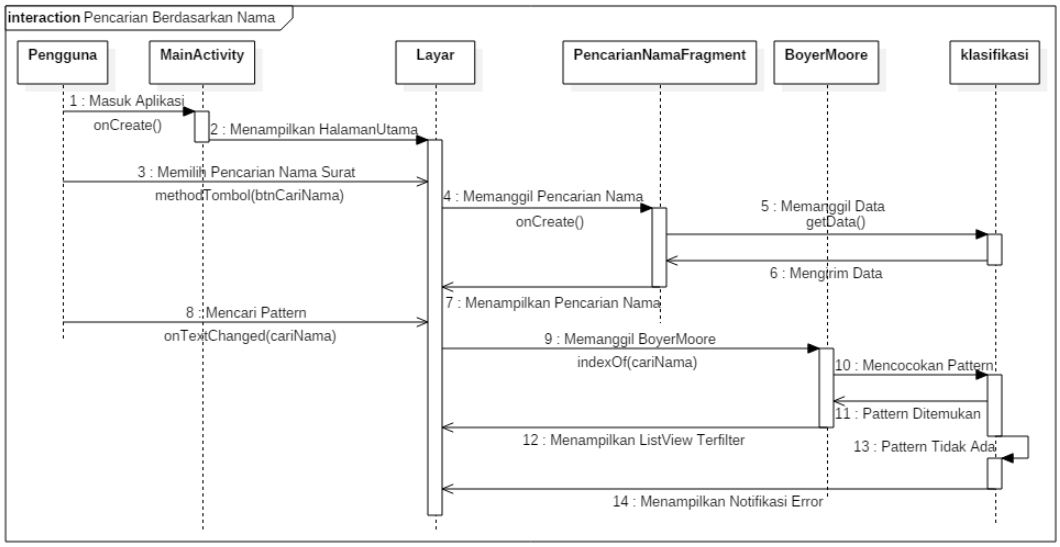
Gambar 4.9 Perancangan *Sequence Diagram* Pencarian Berdasarkan Kode

Berikut ini adalah penjelasan *sequence diagram* Pencarian Berdasarkan Kode:

1) Pengguna masuk ke aplikasi. Ketika aplikasi dibuka sistem akan memanggil kelas MainActivity dan menjalankan prosedur *onCreate*().
2) Aplikasi pertama kali akan menampilkan *layout* Halaman Utama. *Layout* ini didefinisikan dari kelas MainActivity.
3) Pengguna menekan tombol Pencarian Kode Surat. Ketika tombol ditekan, *methodTombol*() akan diproses dengan parameter nama variabel tombol yang ditekan tersebut, yaitu btnCariKode.
4) Aplikasi memanggil kelas PencarianActivity dan menjalankan prosedur *onCreate*().
5) Aplikasi memanggil tabel klasifikasi pada basis data memakai prosedur *getData*().

6) Aplikasi mengirim basis data kepada kelas PencarianActivity untuk ditampilkan di *layout*/halaman Pencarian Berdasarkan Kode.
7) Aplikasi menampilkan halaman Pencarian Berdasarkan Kode.
8) Pengguna menekan *pattern* yang ingin dicari. Ketika pengguna mengetik *pattern* pada keyboard, prosedur *onTextChanged*() akan diproses dengan parameter nama variabel TextView, yaitu cariKode.
9) Aplikasi memanggil prosedur *indexOf*() pada kelas BoyerMoore dengan parameter cariKode dari TextView.
10) Aplikasi mencocokan *pattern* dengan data di basis data.
11) Jika *pattern* ditemukan pada basis data, aplikasi akan mengirim data tersebut.
12) Aplikasi menampilkan data yang cocok dengan *pattern* di *ListView* halaman Pencarian Berdasarkan Kode.
13) Jika *pattern* tidak cocok dengan data yang ada pada basis data maka langkah 14 dilakukan, selain itu abaikan langkah 14.
14) Tampilkan notifikasi data tidak ditemukan/*error*.

2. *Sequence Diagram* Pencarian Berdasarkan Nama



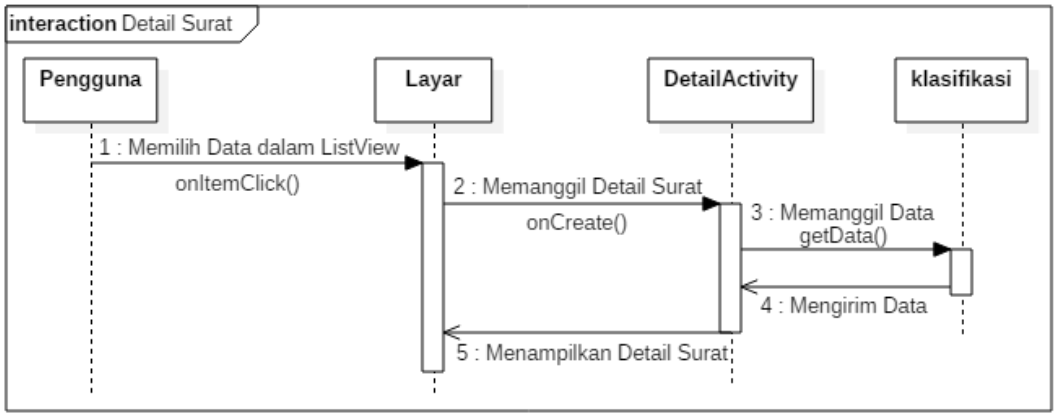
Gambar 4.10 Perancangan *Sequence Diagram* Pencarian Berdasarkan Nama

Berikut ini adalah penjelasan *sequence diagram* Pencarian Berdasarkan Nama:

1) Pengguna masuk ke aplikasi. Ketika aplikasi dibuka sistem akan memanggil kelas MainActivity dan menjalankan prosedur *onCreate*().
2) Aplikasi pertama kali akan menampilkan *layout* Halaman Utama. *Layout* ini didefinisikan dari kelas MainActivity.
3) Pengguna menekan tombol Pencarian Kode Surat. Ketika tombol ditekan, *methodTombol*() akan diproses dengan parameter nama variabel tombol yang ditekan tersebut, yaitu btnCariNama.
4) Aplikasi memanggil kelas PencarianNamaFragment dan menjalankan prosedur *onCreate*().
5) Aplikasi memanggil tabel klasifikasi pada basis data memakai prosedur *getData*().
6) Aplikasi mengirim basis data kepada kelas PencarianNamaFragment untuk ditampilkan di *layout*/halaman Pencarian Berdasarkan Nama.
7) Aplikasi menampilkan halaman Pencarian Berdasarkan Nama pada layar.
8) Pengguna menekan *pattern* yang ingin dicari. Ketika pengguna mengetik *pattern* pada keyboard, prosedur *onTextChanged*() akan diproses dengan parameter nama variabel TextView, yaitu cariNama.
9) Aplikasi memanggil prosedur *indexOf*() pada kelas BoyerMoore dengan parameter cariNama dari TextView.
10) Aplikasi mencocokan *pattern* dengan data yang ada pada tabel di basis data.
11) Jika *pattern* ditemukan pada basis data, aplikasi akan mengirim data tersebut.
12) Aplikasi menampilkan data yang cocok dengan *pattern* di *ListView* halaman Pencarian Berdasarkan Nama.

13) Jika *pattern* tidak cocok dengan data yang ada pada basis data maka langkah 14 dilakukan, selain itu abaikan langkah 14.
14) Tampilkan notifikasi data tidak ditemukan/*error*.

3. *Sequence Diagram* Detail Surat

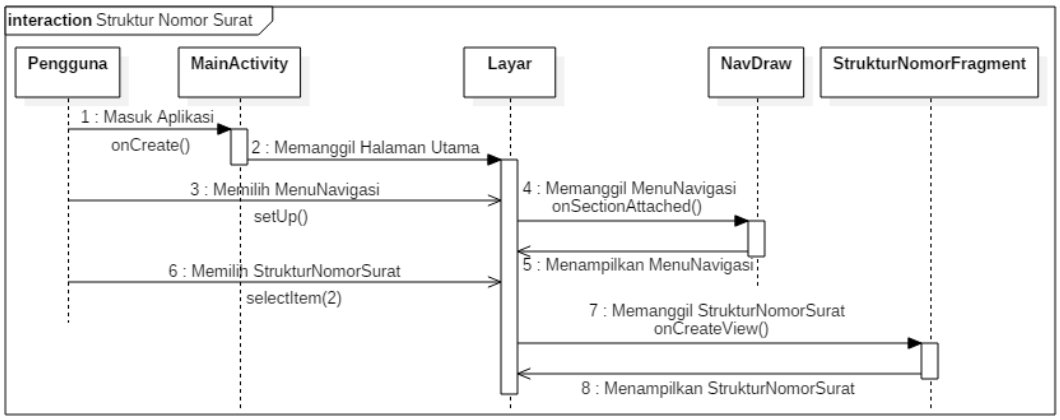


Gambar 4.11 Perancangan *Sequence Diagram* Detail Surat

Berikut ini adalah penjelasan *sequence diagram* halaman Detail Surat:

1) Pengguna menekan salah satu *ListView* klasifikasi kode surat yang ada di halaman Pencarian Berdasarkan Kode atau Nama. Prosedur *onItemClick*() akan diproses oleh aplikasi.
2) Aplikasi memanggil kelas DetailActivity. Kelas ini akan memanggil halaman Detail Surat melalui prosedur *onCreate*(), yaitu prosedur yang digunakan untuk pendefinisian awal atau pembuatan suatu *activity*.
3) Aplikasi memanggil tabel klasifikasi pada basis data memakai prosedur *getData*().
4) Aplikasi mengirim basis data kepada kelas DetailActivity untuk ditampilkan di halaman Detail Surat.
5) Aplikasi menampilkan halaman Detail Surat. Halaman Detail Surat ini akan menampilkan data yang telah dikirim sebelumnya oleh *getData*().

4. *Sequence Diagram* Struktur Nomor Surat



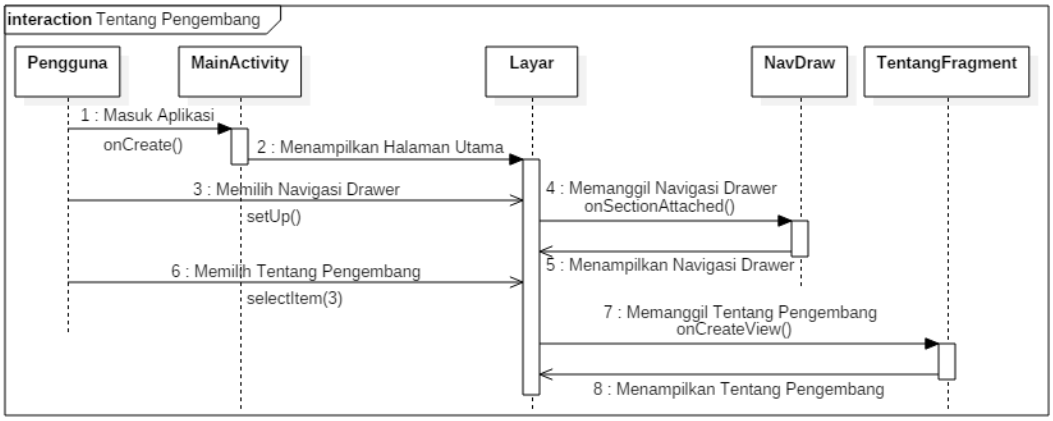
Gambar 4.12 Perancangan *Sequence Diagram* Struktur Nomor Surat

Berikut ini adalah penjelasan *sequence diagram* Struktur Nomor Surat:

1) Pengguna masuk ke aplikasi. Aplikasi akan memanggil kelas MainActivity dan menjalankan prosedur *onCreate*().
2) Aplikasi pertama kali akan menampilkan *layout* Halaman Utama. *Layout* ini didefinisikan dari kelas MainActivity.
3) Pengguna menekan tombol Menu Navigasi. Ketika pengguna menekan tombol Menu Navigasi, prosedur *setUp*() akan dijalankan untuk menyiapkan halaman Menu Navigasi.
4) Aplikasi memanggil kelas NavigationDrawerFragment dan menyiapkan prosedur *onSectionAttached*() untuk menyiapkan seleksi kondisi atau pemilihan halaman yang akan dibuka oleh pengguna.
5) Aplikasi menampilkan halaman Menu Navigasi pada layar.
6) Pengguna menekan tombol Struktur Nomor Surat pada halaman *list* Menu Navigasi. Prosedur *selectItem*() akan dijalankan aplikasi, parameter dalam prosedur tersebut bernilai 2 yang akan menampilkan halaman Struktur Nomor Surat.
7) Aplikasi memanggil kelas StrukturNomorFragment yang merupakan pendefinisian kelas untuk halaman Struktur Nomor Surat.

8) Aplikasi memperbaharui tampilan layar dengan menampilkan halaman Struktur Nomor Surat.

5. *Sequence Diagram* Tentang Pengembang



Gambar 4.13 Perancangan *Sequence Diagram* Tentang Pengembang

Berikut ini adalah penjelasan *sequence diagram* Tentang Pengembang:

1) Pengguna masuk ke aplikasi. Aplikasi akan memanggil kelas MainActivity dan menjalankan prosedur *onCreate*().
2) Aplikasi pertama kali akan menampilkan *layout* Halaman Utama. *Layout* ini didefinisikan dari kelas MainActivity.
3) Pengguna menekan tombol Menu Navigasi. Ketika pengguna menekan tombol Menu Navigasi, prosedur *setUp*() akan dijalankan untuk menyiapkan halaman Menu Navigasi.
4) Aplikasi memanggil kelas NavigationDrawerFragment dan menyiapkan prosedur *onSectionAttached*() untuk menyiapkan kondisi pemilihan halaman yang akan dibuka oleh pengguna.
5) Aplikasi menampilkan halaman Menu Navigasi pada layar.
6) Pengguna menekan tombol Tentang Pengembang pada halaman *list* Menu Navigasi. Prosedur *selectItem*() akan dijalankan aplikasi, parameter dalam prosedur tersebut bernilai 3 yang akan menampilkan halaman Tentang Pengembang.

7) Aplikasi memanggil kelas TentangFragment yang merupakan pendefinisian kelas untuk halaman Tentang Pengembang.
8) Aplikasi memperbaharui tampilan layar dengan menampilkan halaman Tentang Pengembang.

4.2.6 Basis Data

Pembangunan basis data pada aplikasi Klasifikasi Kode Surat ini menggunakan perangkat lunak DB Browser (SQLite). Rancangan basis data Klasifikasi Kode Surat dapat dilihat pada tabel yang bernama tabel klasifikasi di bawah ini:

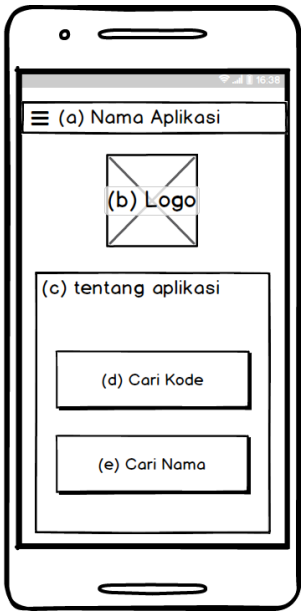
Tabel 4.12 Basis Data klasifikasi

Field	***Type***	**Keterangan**
id_klasifikasi	*INTEGER*	memberikan penomoran indeks secara *auto increment* kepada data klasifikasi kode surat. *Field* ini juga bertindak sebagai *primary key*.
kode_klasifikasi	*TEXT*	menampung data kode surat, tipe *text* dipilih karena data kode surat memiliki data yang bernilai '000'.
nama_klasifikasi	*TEXT*	menampung data nama klasifikasi surat, tipe *text* dipilih karena data nama klasifikasi surat memiliki data *string*.
catatan_klasifikasi	*TEXT*	menampung data catatan klasifikasi, tipe *text* dipilih karena data catatan klasifikasi memiliki data *string*.

4.2.7 *User Interface* (UI)

Perancangan UI dilakukan setelah diagram-diagram yang berkaitan dengan proses pada aplikasi telah dirancang. Didapatkan hasil rancangan bahwa aplikasi yang dibangun akan mempunyai 7 halaman UI, diantaranya:

1. Halaman Utama

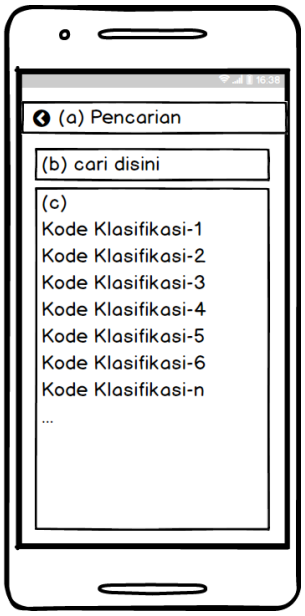


Gambar 4.14 Perancangan UI Halaman Utama

Halaman Utama merupakan UI yang pertama kali dibuka pada aplikasi Klasifikasi Kode Surat, berikut adalah keterangan poin-poinnya:

a) *Action Bar: Menu Navigation Drawer*, merupakan tempat nama aplikasi yang disampingnya ada tombol untuk membuka tampilan Menu Navigasi.
b) *Image View*, *view* atau elemen UI yang digunakan untuk menampilkan gambar/logo aplikasi.
c) Area *Text View*, *view* yang menampilkan teks/*string* sekilas informasi tentang aplikasi.
d) *Button*, tombol yang digunakan untuk membuka halaman Pencarian Berdasarkan Kode.
e) *Button*, tombol yang digunakan untuk membuka halaman Pencarian Berdasarkan Nama.

2. Halaman Pencarian Berdasarkan Kode

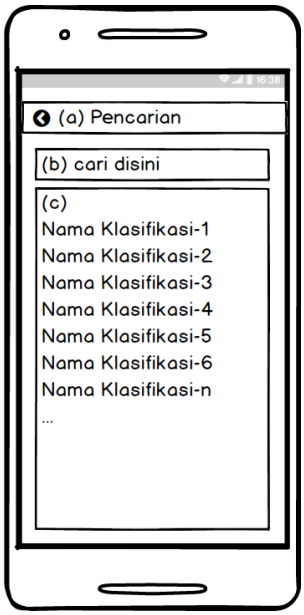


Gambar 4.15 Perancangan UI Pencarian Berdasarkan Kode

Halaman Pencarian Berdasarkan Kode merupakan halaman pencarian klasifikasi kode surat berdasarkan klasifikasi kodenya, berikut adalah keterangan poin-poin pada gambar rancangan di atas:

a) *Action Bar: Back Button*, merupakan tempat yang menampilkan nama *activity* pencarian, dengan tombol kembali ke Halaman Utama yang berada disamping kirinya.
b) *Edit Text*, tempat yang digunakan untuk mencari *pattern* (*string*) klasifikasi kode surat berdasarkan klasifikasi kodenya, yang mana jika diisi akan menampilkan hasil *filter* atau hasil pencarian dari pencocokan *string* menggunakan algoritma Boyer-Moore pada *List View* di bawahnya.
c) *List View*, *view* berbentuk daftar (*list*) yang menampilkan klasifikasi kode surat, tampilan *list* akan berubah jika *Edit Text* dimasukan *pattern* klasifikasi kode surat. *List View* ini akan menampilkan notifikasi hasil pencarian tidak ditemukan jika *pattern* yang sebelumnya dimasukan pada *Edit Text* diatas tidak ditemukan pada basis data aplikasi.

3. Halaman Pencarian Berdasarkan Nama

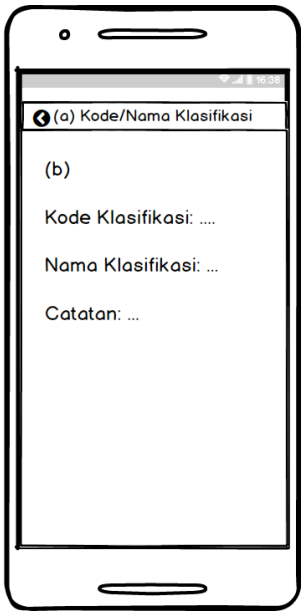


Gambar 4.16 Perancangan UI Pencarian Berdasarkan Nama

Halaman Pencarian Berdasarkan Nama merupakan halaman pencarian klasifikasi kode surat berdasarkan klasifikasi namanya, berikut adalah keterangan poin-poin pada gambar rancangan di atas:

a) *Action Bar: Back Button*, merupakan tempat yang menampilkan nama *activity* pencarian, dengan tombol kembali ke Halaman Utama yang berada disamping kirinya.
b) *Edit Text*, tempat yang digunakan untuk mencari *pattern* (*string*) klasifikasi kode surat berdasarkan klasifikasi namanya, yang mana jika diisi akan menampilkan hasil *filter* atau hasil pencarian dari pencocokan *string* menggunakan algoritma Boyer-Moore pada *List View* di bawahnya.
c) *List View*, *view* berbentuk daftar (*list*) yang menampilkan klasifikasi kode surat, tampilan *list* akan berubah jika *Edit Text* dimasukan *pattern* klasifikasi kode surat. *List View* ini akan menampilkan notifikasi hasil pencarian tidak ditemukan jika *pattern* yang sebelumnya dimasukan pada *Edit Text* diatas tidak ditemukan pada basis data aplikasi.

4. Halaman Detail Surat

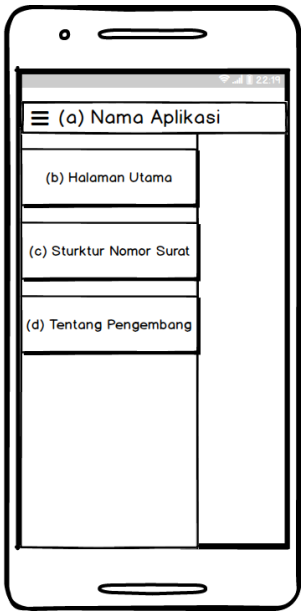


Gambar 4.17 Perancangan UI Detail Surat

Halaman Detail Surat merupakan halaman yang dipanggil dari hasil menekan *List View* klasifikasi kode surat yang ada pada halaman Pencarian Berdasarkan Kode atau halaman Pencarian Berdasarkan Nama. Berikut adalah keterangan poin-poin pada gambar rancangan di atas:

a) *Action Bar: Back Button*, merupakan tempat yang menampilkan nama kode atau nama klasifikasi (tergantung dari *List View* yang ditekan pada *activity* sebelumnya), dengan tombol kembali ke halaman sebelumnya.
b) Area *Text View*, *view* yang menampilkan teks/*string* tentang informasi klasifikasi kode surat yang diambil dari basis data aplikasi, meliputi:
   - Klasifikasi kode surat yang ditampilkan dengan nomor desimal 3 digit.
   - Klasifikasi nama surat.
   - Catatan tenang penggunaan klasifikasi kode surat tersebut (jika ada).

5. Halaman Menu Navigasi

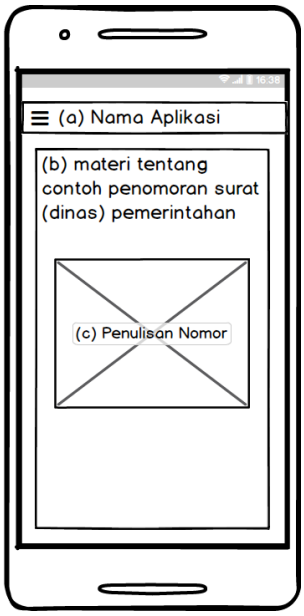


Gambar 4.18 Perancangan UI Menu Navigasi

Halaman Menu Navigasi merupakan halaman atau tampilan UI yang menampilkan *list* pada Halaman Utama, halaman Struktur Nomor Surat, dan halaman Tentang Pengembang. Berikut adalah keterangan poin-poin pada gambar rancangan di atas:

a) *Action Bar: Menu Navigation Drawer*, merupakan tempat nama aplikasi yang disampingnya ada tombol untuk membuka tampilan Menu Navigasi.
b) *List Button*, *list* yang dapat ditekan untuk membuka Halaman Utama. Jika halaman yang diakses sebelumnya adalah Struktur Nomor Surat atau Tentang Pengembang, maka proses kelas yang menangani sistem akan berubah kembali menjadi MainActivity.
c) *List Button*, *list* yang dapat ditekan untuk membuka halaman Struktur Nomor Surat dan proses kelas yang menangani sistem akan berpindah dari MainActivity menjadi StrukturNomorFragment.
d) *List Button*, *list* yang dapat ditekan untuk membuka halaman Tentang Pengembang dan proses kelas yang menangani sistem akan berpindah dari MainActivity menjadi TentangFragment.

6. Halaman Struktur Nomor Surat

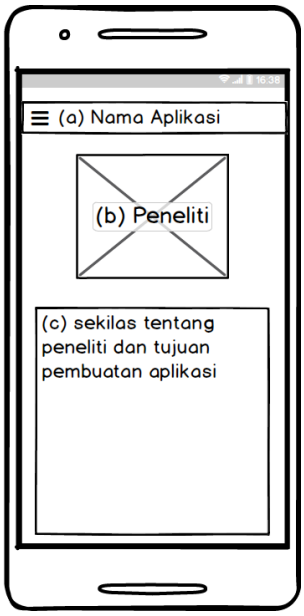


Gambar 4.19 Perancangan UI Struktur Nomor Surat

Halaman Struktur Nomor Surat merupakan halaman yang menampilkan tentang materi singkat dari struktur penomoran arsip/surat dinas resmi di lingkungan Kementerian Dalam Negeri dan Pemerintah Daerah. Berikut adalah keterangan poin-poin pada gambar di atas:

a) *Action Bar: Menu Navigation Drawer*, merupakan tempat nama aplikasi yang disampingnya ada tombol untuk membuka tampilan Menu Navigasi.
b) Area *Text View*, *view* yang menampilkan teks/*string* informasi tentang materi tentang penulisan struktur nomor arsip/surat dinas resmi.
c) *Image View*, *view* atau elemen UI yang digunakan untuk menampilkan gambar materi tentang struktur nomor surat. Gambar yang digunakan pada halaman ini berjumlah lebih dari 1, karena gambar tersebut digunakan sebagai pembanding terhadap contoh penulisan nomor surat dinas resmi yang pernah dibuat oleh Pemerintah Daerah ataupun Kementerian Dalam Negeri.

7. Halaman Tentang Pengembang



Gambar 4.20 Perancangan UI Tentang Pengembang

Halaman Tentang Pengembang merupakan halaman yang menampilkan foto dan informasi tentang peneliti/pengembang aplikasi. Berikut adalah keterangan poin-poin pada gambar rancangan di atas:

a) *Action Bar: Menu Navigation Drawer*, merupakan tempat nama aplikasi yang disampingnya ada tombol untuk membuka tampilan Menu Navigasi.
b) *Image View*, *view* atau elemen UI yang digunakan untuk menampilkan gambar/foto dari pengembang aplikasi.
c) Area *Text View*, *view* yang menampilkan teks/*string* sekilas informasi profil tentang pengembang aplikasi dan tentang tujuan dari pembuatan aplikasi.

BAB V

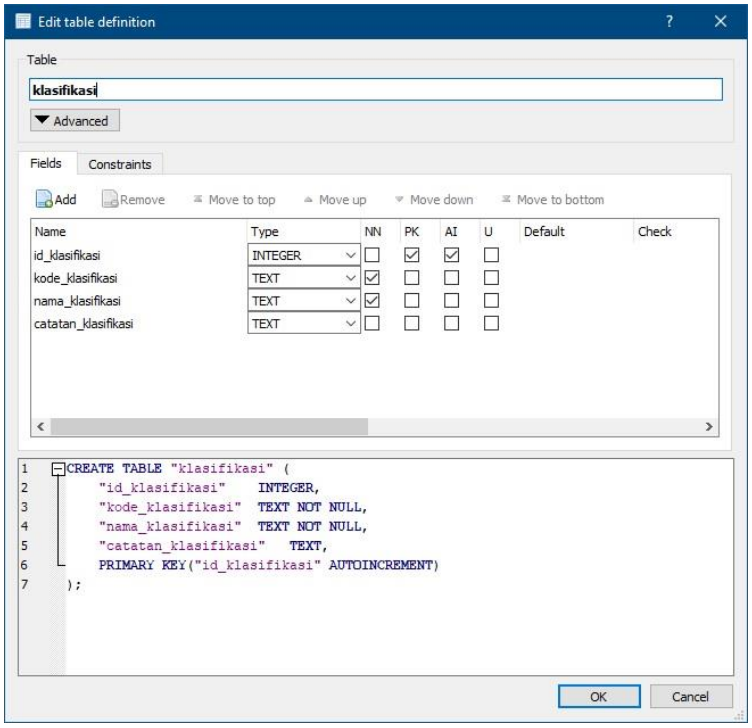
IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi

Tahap Implementasi ini dilakukan sesuai dengan perancangan aplikasi yang dilakukan pada bab sebelumnya. Tahap ini terdiri dari implementasi basis data, tampilan UI aplikasi, serta implementasi algoritma Boyer-Moore.

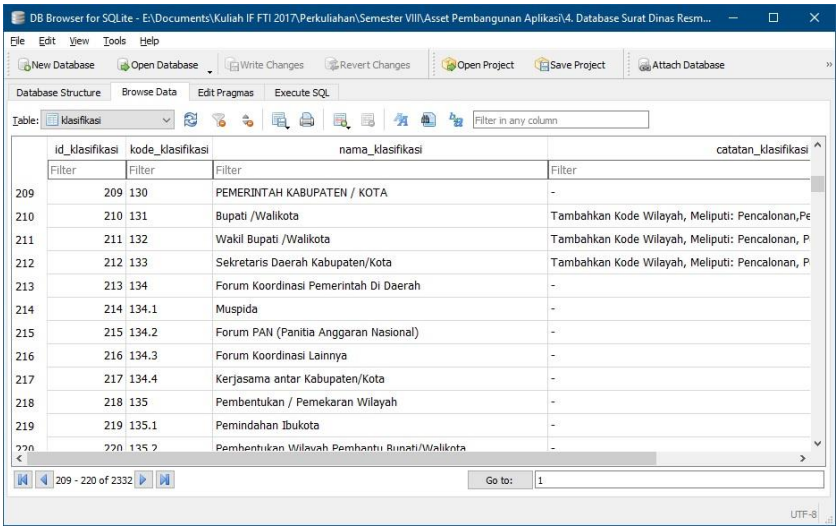
5.1.1 Implementasi Basis Data

Implementasi dimulai dengan tahapan pembuatan basis data pada aplikasi DB Browser (SQLite) berdasarkan rancangan basis data yang telah dibuat sebelumnya. Tabel yang dibuat pada basis data aplikasi ini hanya satu, tabel tersebut bernama tabel klasifikasi yang memiliki 4 *field* yang masing-masing nantinya akan menampung data yang akan dimasukan.



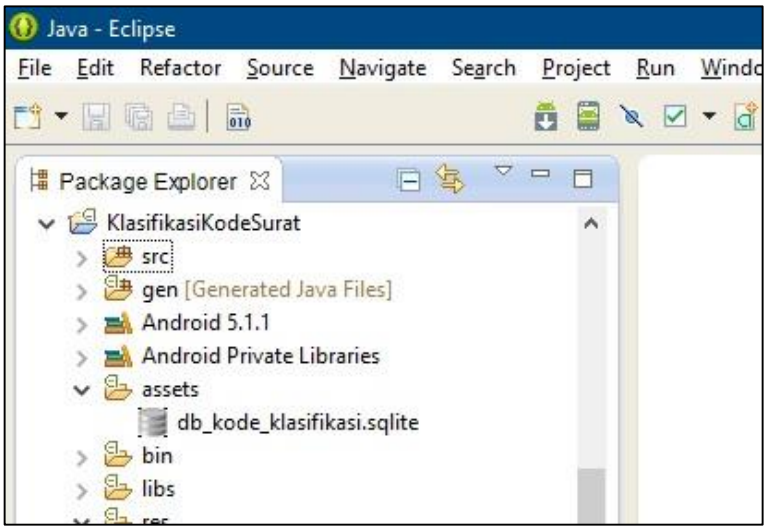
Gambar 5.1 Implementasi Tabel Basis Data pada DB Browser (SQLite)

Tahap selanjutnya dari implementasi basis data ini adalah dengan memasukan seluruh klasifikasi kode arsip yang berjumlah 2332 indeks kepada tabel klasifikasi. Data klasifikasi kode arsip tersebut didapatkan dari Permendagri no. 78 tahun 2012, dan no. 135 tahun 2017.



Gambar 5.2 Implementasi Konten Basis Data pada DB Browser (SQLite)

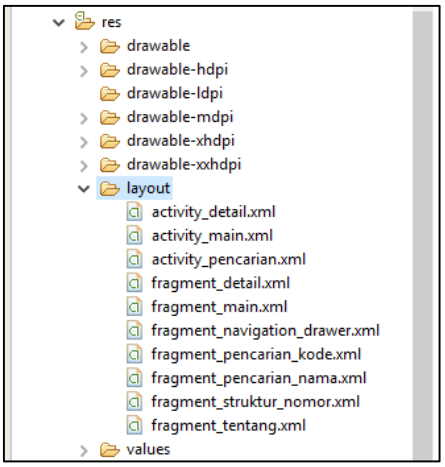
Setelah data klasifikasi kode arsip telah selesai di-*input*-kan seluruhnya, selanjutnya basis data tersebut disimpan dengan *file* db_kode_klasifikasi.sqlite, dan diimplementasikan kepada struktur *project* direktori android yang bernama *assets* pada Eclipse.



Gambar 5.3 Implementasi Basis Data pada Direktori *assets*

5.1.2 Implementasi Antarmuka

Implementasi antarmuka ini adalah tahap pengkodean antarmuka yang telah dirancang pada tahap perancangan *user interface* (UI). Pengkodean antarmuka dilakukan dengan menggunakan tag bahasa xml kepada 7 halaman rancangan UI, 7 halaman tersebut diberi awalan nama *fragment*, seperti gambar dibawah:



Gambar 5.4 Implementasi Antarmuka Halaman pada Eclipse

Terlihat bahwa pada gambar diatas terdapat *file* lain yang memiliki awalan nama activity_*.xml, *file* tersebut merupakan *file* utama dari masing-masing halaman *fragment*, penggunaan *fragment* sendiri adalah untuk penggunaan ulang suatu *activity* pada pemrograman aplikasi di Android agar lebih sederhana, efisien terhadap memori, dan terstruktur. Berikut ini adalah penjelasan dari tugas masing-masing *activity* tersebut:

- activity_main menangani penampilan fragment_main, fragment_navigation_drawer, fragment_struktur_nomor, dan fragment_tentang
- activity_pencarian menangani penampilan fragment_pencarian _kode dan fragment_pencarian nama
- activity_detail menangani penampilan fragment_detail

Untuk *source code* tentang implementasi antarmuka dapat dilihat pada lampiran dengan nama *file* yang berakhiran dengan ekstensi *.xml.

5.1.3 Implementasi Algoritma Boyer-Moore

Tahap selanjutnya dari tahap implementasi ini adalah tahap implementasi algoritma Boyer-Moore. Contoh pencocokan *string* yang terjadi jika diketahui sebuah teks memiliki *string* SAYASUKAMAKAN, dengan *pattern* yang dicari adalah MAKAN, maka pencarian yang dilakukan algoritma Boyer-Moore adalah sebagai berikut:

- Membuat tabel delta dari *pattern* yang dicari (*occurrence heuristic*). Karakter pada *pattern* yang diketahui adalah M-A-K-N, maka dengan menggunakan teknik *looking glass* (melihat dari kanan) pada Boyer-Moore, karakter N akan memiliki $indeks = 0$, $A = 1\, dan\, 3$ (1 merupakan nilai yang muncul paling awal dari pada 3), $K = 2$, dan $M = 4$ seperti pada tabel berikut:

Tabel 5.1 *Occurrence Heuristic (OH)*

Indeks	M	A	K	N
0				■
1		■		
2			■	
3		■		
4	■			

Karakter *Pattern*	M	A	K	N
Kemunculan Paling Awal (OH)	4	1	2	0

- Setelah tabel delta diketahui, proses pencocokan *string* dengan menggunakan algoritma Boyer-Moore adalah sebagai berikut:

Tabel 5.2 Proses Pencocokan *String*

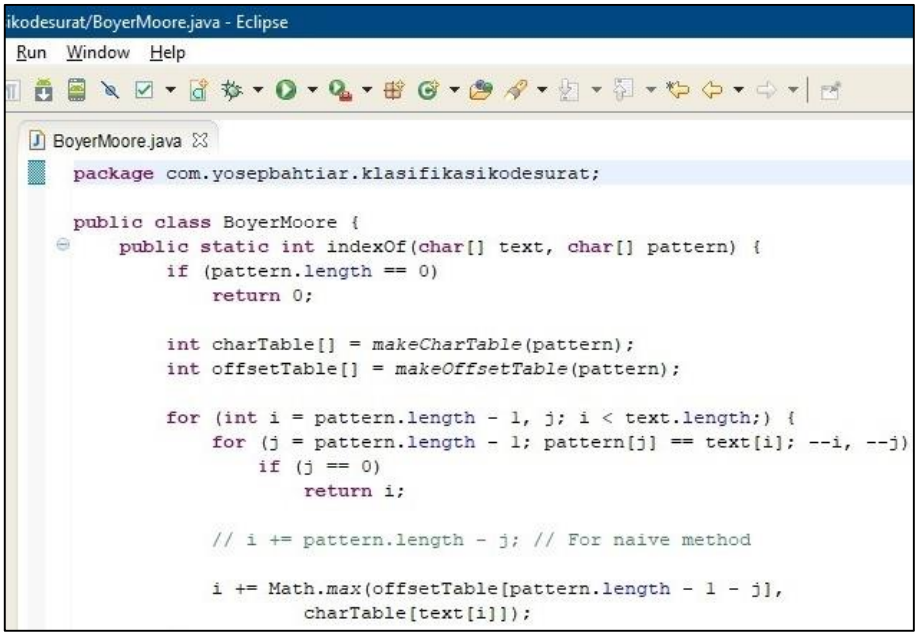
Indeks	0	1	2	3	4	5	6	7	8	9	10	11	12
Teks	S	A	Y	A	**S**	U	K	A	M	**A**	**K**	A	N
Pattern	M	A	K	A	**N**								
						M	A	K	A	**N**			
							M	A	K	A	**N**		
									M	A	K	A	N

- Proses pencocokan *string* yang pertama berdasarkan pada tabel diatas adalah proses mencocokan karakter dari *pattern* paling kanan yang memiliki indeks 4 dengan karakter dari teks yang memiliki indeks sama seperti *pattern,* yaitu karakter N dengan karakter S. Karena karakter $N \neq S$ dan karakter S pada tabel delta tidak ada, maka *pattern* digeser sebanyak jumlah karakter yang dimiliki oleh *pattern*, yaitu 5.
- Pencocokan kedua adalah pencocokan karakter N pada *pattern* dengan karakter A pada teks. Karena karakter $N \neq A$ tetapi karakter A terdapat pada tabel delta dengan nilai $OH(A) = 1$, maka *pattern* digeser sebanyak jumlah OH pada karakter A, yaitu 1.
- Pencocokan ketiga adalah pencocokan karakter N pada *pattern* dengan karakter K pada teks. Karena karakter $N \neq K$ tetapi karakter K terdapat pada tabel delta dengan nilai $OH(K) = 2$, maka *pattern* digeser sebanyak jumlah OH pada karakter A, yaitu 2.
- Pencocokan keempat adalah pencocokan karakter N pada *pattern* dengan karakter N pada teks. Karena karakter $N = N$, maka pencocokan karakter sukses dan dilanjutkan dengan mencocokan karakter pada indeks teks dan *pattern* sebelumnya, yaitu karakter A pada *pattern* dengan karakter A pada teks.
- Setelah dilakukan pencocokan dari kanan ke kiri teks terhadap *pattern* dan ditemukan kecocokan seluruh *pattern* pada teks. Maka algoritma akan berhenti dan mengembalikan mengembalikan nilai 8, yaitu nilai dimana indeks pada teks ditemukan kecocokan yang terakhir kalinya dengan *pattern*.

Diatas merupakan contoh pencocokan *string* menggunakan algoritma Boyer-Moore, dapat disimpulkan bahwa algoritma Boyer-Moore ini dapat dengan cepat menemukan *pattern* yang ingin dicari pada teks, terlihat pada jumlah pergerakan geser yang dilakukan pada *pattern* ketika algoritma menemukan ketidakcocokan pada karakter teks yang dicari. Proses pencocokan tersebut sangat berbeda dengan algoritma konvensional seperti Brute Force dan Knuth-Morris-Pratt yang mencocokan *pattern* dari

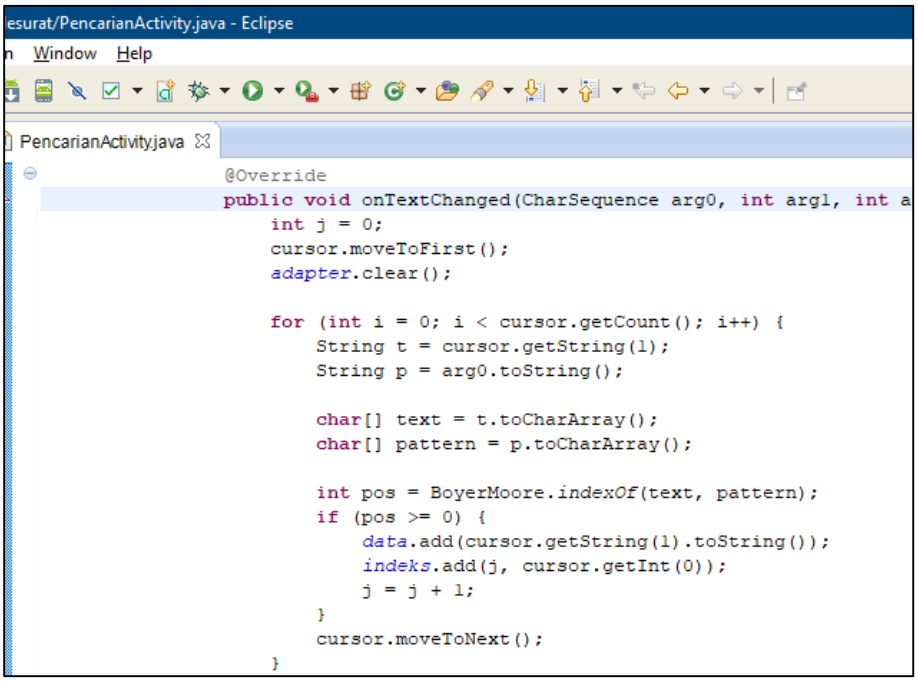
kiri ke kanan dan menggeser *pattern* dengan jumlah geser *pattern*-nya yang relatif kecil.

Algoritma Boyer-Moore tersebut diimplementasikan kepada kelas dengan nama *file* BoyerMoore.java, kelas tersebut akan mengembalikan nilai integer antara $0 \ldots n - 1$, dimana n adalah banyaknya indeks pada teks yang merupakan tempat *pattern* yang akan dicari, atau dalam hal ini teks tersebut dianggap sebagai suatu baris pada suatu kolom *field* kode atau nama surat dalam basis data. Jadi dapat dikatakan bahwa nilai integer yang dikembalikan dari kelas Boyer-Moore tersebut adalah nilai yang menunjukan posisi *pattern* ditemukan pada teks.



Gambar 5.5 Penggalan *source code* kelas BoyerMoore.java

Kelas BoyerMoore digunakan oleh PencarianActivity.java dan PencarianNamaFragment.java dengan cara menampungkan prosedur *indexOf*() milik BoyerMoore pada variabel integer pos dalam prosedur *onTextChanged*(). Prosedur *onTextChanged*() sendiri merupakan prosedur yang membaca aktifitas perubahan *string* pada *edit text* di kelas PencarianActivity dan PencarianNamaFragment. Jadi ketika *user* mengetikan sesuatu pada *edit text*, prosedur tersebut akan berjalan.



Gambar 5.6 Pemanggilan kelas BoyerMoore pada PencarianActivity.java

Dengan menampungkan data suatu baris menggunakan perintah *cursor.getString*() pada variabel *text*, dan variabel arg0 yang merupakan data *edit text* yang berubah sesuai dengan masukan pengguna kepada variabel *pattern*, maka aplikasi akan menampilkan suatu baris yang sama dengan *pattern* tersebut pada *list view* di halaman pencarian yang memiliki nilai integer antara $0 \ldots n-1$. Jika baris pada teks tersebut tidak cocok dengan *pattern* yang dimasukan (tidak memiliki nilai integer $0 \ldots n-1$), maka baris tersebut akan diabaikan dan proses akan diteruskan sampai kondisi variabel i < cursor.getCount() tidak memenuhi syarat pengulangan.

Selain melakukan implementasi algoritma Boyer-Moore, pada tahap ini peneliti juga melakukan implementasi terhadap pembuatan kelas-kelas yang digunakan dalam aplikasi seperti kelas MainActivity, NavigationDrawerFragment, PencarianActivity, PencarianNamaFragment, DetailActivity, SQLHelper, StrukturNomorFragment, dan TentangFragment. Semua *source code* pada kelas-kelas tersebut dapat dilihat pada lampiran dengan nama *file* yang sama, dan berakhiran dengan ekstensi *.java.

5.1.4 Hasil Implementasi

Hasil implementasi merupakan hasil *build* dari keseluruhan tahap implementasi atau pengkodean terhadap basis data, antarmuka, algoritma Boyer-Moore, dan prosedur-prosedur lain. Hasil daripada tahap tersebut adalah aplikasi Klasifikasi Kode Surat.apk yang dapat dipasang dan dijalankan pada sistem operasi Android. Berikut adalah hasil tangkapan layar dari aplikasi yang dijalankan peneliti pada perangkat *smartphone*:

1. Halaman Utama

Halaman ini merupakan tampilan yang akan tampil pertama kali saat aplikasi dibuka oleh pengguna, terlihat pada gambar dibawah bahwa komponen-komponen *user interface* Halaman Utama telah dibangun sesuai dengan rancangan pada tahap perancangan sebelumnya, yaitu memiliki:

- 2 tombol yang masing-masing digunakan untuk berpindah ke halaman Pencarian Berdasarkan Kode Surat dan halaman Pencarian Berdasarkan Nama Surat.
- Sebuah tombol navigasi di pojok kiri atas yang digunakan untuk membuka Menu Navigasi.
- Komponen gambar serta teks yang menampilkan info aplikasi.

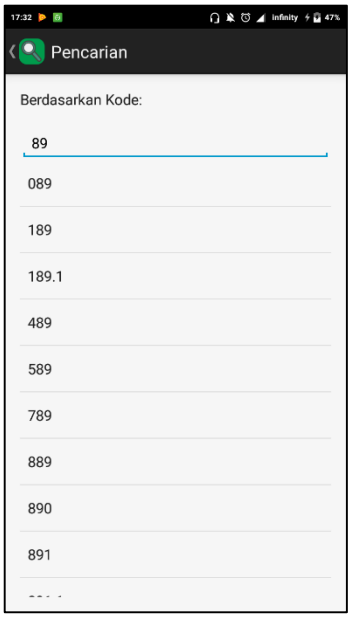
Gambar 5.7 Tampilan Halaman Utama

2. Halaman Pencarian Berdasarkan Kode

Halaman Pencarian Berdasarkan Kode ini merupakan halaman yang digunakan oleh pengguna untuk mencari klasifikasi surat berdasarkan kodenya. Halaman ini telah dibangun sesuai rancangan pada tahap perancangan *user interface*, dengan komponen-komponen yang terdiri dari:

- Tombol *back* di pojok kiri atas yang digunakan untuk kembali ke halaman sebelumnya.
- *Edit text* yang digunakan untuk memasukan *pattern*.
- *List view* yang menampilkan kode yang cocok dengan *pattern*.

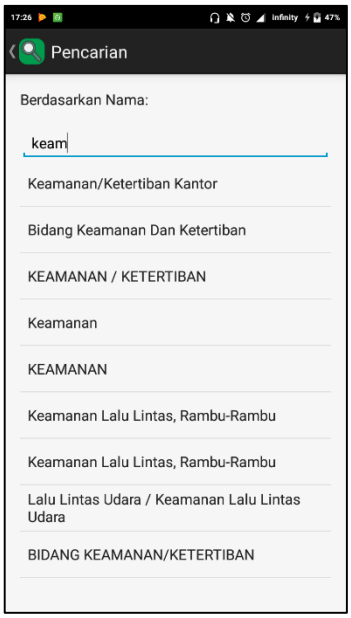
Terlihat juga pada gambar dibawah bahwa pencarian *string* menggunakan algoritma Boyer-Moore telah berhasil diimplementasikan.



Gambar 5.8 Tampilan Pencarian Berdasarkan Kode

3. Halaman Pencarian Berdasarkan Nama

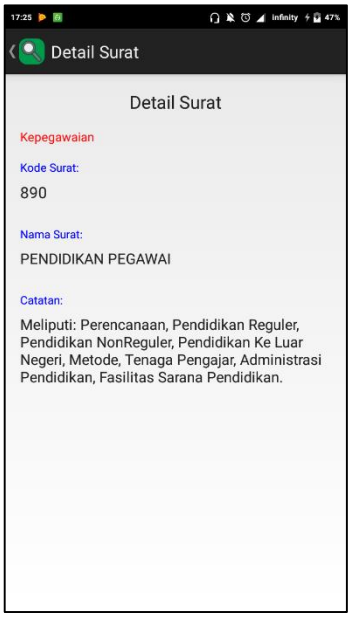
Seperti pada halaman Pencarian Berdasarkan Kode, halaman ini menggunakan komponen-komponen yang sama seperti tombol *back, edit text*, dan *list view*. Halaman ini juga telah berhasil algoritma Boyer-Moore.



Gambar 5.9 Tampilan Pencarian Berdasarkan Nama

4. Halaman Detail Surat

Halaman Detail Surat ini merupakan halaman yang tampil setelah sebuah *list view* ditekan pada halaman pencarian, baik halaman Pencarian Berdasarkan Kode maupun halaman Pencarian Berdasarkan Nama.

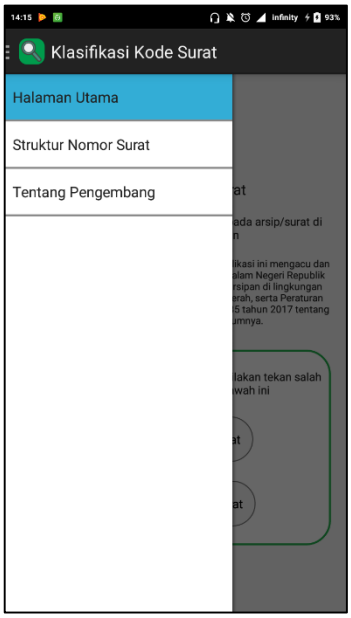


Gambar 5.10 Tampilan Detail Surat

5. Halaman Menu Navigasi

Halaman Menu Navigasi ini merupakan halaman yang digunakan untuk memudahkan perpindahan halaman antara Halaman Utama, halaman Struktur Nomor Surat, dan halaman Tentang Pengembang. Halaman ini telah dibangun sesuai rancangan pada tahap perancangan *user interface*, dengan komponen-komponen yang terdiri dari:

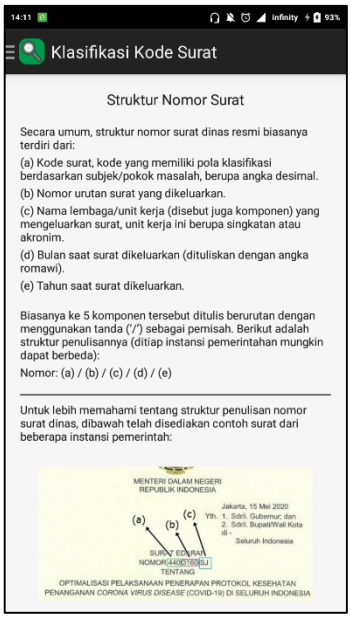
- Tombol navigasi di pojok kiri atas yang digunakan untuk menutup Menu Navigasi.
- *List view* yang menampilkan *list* halaman yang tersedia untuk dilakukan perpindahan.



Gambar 5.11 Tampilan Menu Navigasi

6. Halaman Struktur Nomor Surat

Halaman ini merupakan halaman yang digunakan untuk menampilkan materi tentang struktur penomoran pada surat dinas di lingkungan pemerintah. Halaman ini telah dibangun sesuai dengan rancangan pada tahap perancangan, yaitu menampilkan komponen *user interface* seperti gambar serta teks.



Gambar 5.12 Tampilan Struktur Nomor Surat

7. Halaman Tentang Pengembang

Halaman Tentang Pengembang ini mempunyai komponen *user interface* gambar dan teks. Informasi yang ditampilkan adalah informasi tentang peneliti dan tujuan tentang tentang pembuatan aplikasi.

Gambar 5.13 Tampilan Tentang Pengembang

5.2 Pengujian

Setelah aplikasi selesai dipasang dan dijalankan pada perangkat keras yang digunakan peneliti, selanjutnya adalah pelaksanaan tahap pengujian. Tahap pengujian ini dilakukan dengan metode *black-box testing* dan pengujian kelayakan terhadap aplikasi yang telah dibangun.

1. Pengujian *Black-Box*

Pengujian *black-box* digunakan untuk mengetahui unjuk kerja dari aplikasi Klasifikasi Kode Surat dengan cara menguji fungsionalitas dari aplikasi tersebut tanpa menguji kode programnya. Pengujian dilakukan dengan cara menjalankan setiap fungsi masukan pada aplikasi di beberapa *smartphone* Android dengan versi Android dan spesikasi perangkat yang berbeda-beda. Berikut hasil dari uji fungsionalitas aplikasi yang dibangun:

Tabel 5.3 Pengujian *Black-Box*

No	Item Uji	Skenario Uji	Hasil yang diharapkan	Hasil
1	Halaman Utama	Menekan tombol Menu Navigasi	Ketika tombol ditekan, diharapkan halaman Menu Navigasi akan muncul dari sisi kiri halaman.	+Berhasil
		Menekan tombol Pencarian Berdasarkan Kode Klasifikasi	Ketika tombol ditekan, diharapkan halaman Pencarian Kode Klasifikasi terbuka.	+Berhasil
		Menekan tombol Pencarian Berdasarkan Nama Klasifikasi	Ketika tombol ditekan, diharapkan halaman Pencarian Nama Klasifikasi terbuka.	+Berhasil

2	Halaman Pencarian Berdasarkan Kode Surat	Menekan tombol *Up/ Back Button*	Ketika tombol ditekan, diharapkan halaman akan berpindah menuju halaman sebelumnya.	+Berhasil
		Mencari klasifikasi berdasarkan kode surat pada *view Edit Text*	Ketika kode/*pattern* yang ingin dicari dimasukan, *List View* dibawahnya akan menampilkan kode yang cocok.	+Berhasil
		Menampilkan notifikasi data tidak tersedia	Ketika kode/*pattern* yang dicari tidak tersedia, aplikasi akan menampilkan notifikasi bahwa data tidak tersedia.	+Berhasil
		Menekan data yang tampil pada *List View*	Ketika data yang tampil pada *List View* ditekan, diharapkan halaman Detail Surat akan terbuka.	+Berhasil
3	Halaman Pencarian Berdasarkan Nama Surat	Menekan tombol *Up/ Back Button*	Ketika tombol ditekan, diharapkan halaman akan berpindah menuju halaman sebelumnya.	+Berhasil
		Mencari klasifikasi berdasarkan nama surat pada *view Edit Text*	Ketika nama/*pattern* yang ingin dicari dimasukan, *List View* dibawahnya akan menampilkan kode yang cocok.	+Berhasil

		Menampilkan notifikasi data tidak tersedia	Ketika nama/*pattern* yang dicari tidak tersedia, aplikasi akan menampilkan notifikasi bahwa data tidak tersedia.	+Berhasil
		Menekan data yang tampil pada *List View*	Ketika data yang tampil pada *List View* ditekan, diharapkan halaman Detail Surat akan terbuka.	+Berhasil
4	Halaman Detail Surat	Menekan tombol *Up/ Back Button*	Ketika tombol ditekan, diharapkan halaman akan berpindah menuju halaman sebelumnya.	+Berhasil
5	Halaman Menu Navigasi	Menekan tombol Halaman Utama	Ketika tombol ditekan, diharapkan halaman	+Berhasil
		Menekan tombol Halaman Struktur Nomor Surat	Ketika tombol ditekan, diharapkan halaman	+Berhasil
		Menekan tombol Tentang Pengembang	Ketika tombol ditekan, diharapkan halaman	+Berhasil
6	Halaman Struktur Nomor Surat	Menekan tombol Menu Navigasi	Ketika tombol ditekan, diharapkan halaman Menu Navigasi akan muncul dari sisi kiri halaman.	+Berhasil

7	Halaman Tentang Pengembang	Menekan tombol Menu Navigasi	Ketika tombol ditekan, diharapkan halaman Menu Navigasi akan muncul dari sisi kiri halaman.	+Berhasil

2. Pengujian Penerimaan Pengguna

Pengujian penerimaan ini dilakukan untuk mengetahui efektifitas dan kelayakan berdasarkan penilaian oleh pengguna (subjek penelitian) terhadap aplikasi yang dibangun dengan menggunakan metode pengumpulan data kuesioner. Lebih lengkap kuesioner dapat dilihat pada lampiran 2.

Berdasarkan kuesioner penilaian pengguna terhadap aplikasi tersebut, dapat disimpulkan bahwa aplikasi Klasifikasi Kode Surat yang dibangun peneliti layak digunakan oleh petugas kearsipan untuk mencari kode klasifikasi surat.

BAB VI

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil analisis, perancangan, pengujian dengan menggunakan metode *black-box testing* dan pengujian efektifitas dan kelayakan aplikasi menggunakan kuesioner pada penelitian ini, didapatkan hasil bahwa:

1. Aplikasi dapat mengimplementasikan pencocokan *string* dengan menggunakan algoritma Boyer-Moore.
2. Pencarian pada aplikasi dengan menggunakan algoritma Boyer-Moore dapat menampilkan keseluruhan indeks yang mengandung *pattern* yang dicari, dan tentunya hasil pencarian tersebut ditampilkan dengan sangat cepat oleh aplikasi.
3. Aplikasi Klasifikasi Kode Surat layak digunakan untuk mencari klasifikasi kode arsip/surat di lingkungan pemerintahan daerah.

6.2 Saran

Dalam proses pembangunan aplikasi Klasifikasi Kode Surat berbasis Android menggunakan algoritma Boyer-Moore di Kantor Kecamatan Ciparay ini banyak sekali memiliki kekurangan dan jauh dari sempurna, sehingga peneliti berharap agar peneliti selanjutnya dapat mengembangkan lagi aplikasi ini, seperti:

1. Basis data aplikasi yang bisa diperbaharui secara *online* agar menyesuaikan dengan ketentuan tata kearsipan di lingkungan pemerintahan.
2. Penampilan data terkait tingkatan klasifikasi kode surat/arsip pada halaman Pencarian Berdasarkan Kode dan Nama, supaya klasifikasi kode pokok masalah utama dan kode sub pokok masalah dapat dibedakan dengan mudah.

DAFTAR PUSTAKA

A.S., R., & Shalahuddin, M. (2019). *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek* (Revisi). Informatika.

Astuti, W. (2017). Analisis String Matching Pada Judul Skripsi Dengan Algoritma Knuth-Morris-Pratt (Kmp). *Jurnal Ilmiah ILKOM*, *IX*(2), 167–172. http://jurnal.fikom.umi.ac.id/index.php/ILKOM/article/view/136/90

Budi, D. S., Siswa, T. A. Y., & Abijono, H. (2016). Analisis Pemilihan Penerapan Proyek Metodologi Pengembangan Rekayasa Perangkat Lunak. *Jurnal Teknika*, *V*(1), 24–31. http://ejournal.ikado.ac.id/index.php/teknika/article/view/48/38

Charras, C., & Lecroq, T. (2004). *Handbook of Exact String Matching Algorithms*. 238. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.133.4896&rep=rep1&type=pdf

Daeli, M. M. Y., & Hondro, R. K. (2017). Perancangan Aplikasi Pencarian Kata dengan Kombinasi Algoritma Knuth Morris Pratt dan Algoritma Boyer Moore. *Majalah Ilmiah INTI*, *XII*(2), 271–275. https://ejurnal.stmik-budidarma.ac.id/index.php/inti/article/view/380/362

Dharwiyanti, S., & Wahono, R. S. (2003). *Pengantar Unified Modeling Language (UML)*. IlmuKomputer.Com. https://www.coursehero.com/file/79951527/yanti-umldoc/

Effendi, D., Hartono, T., & Kurnaedi, A. (2013). Penerapan String Matching Menggunakan Algoritma Boyer-Moore Pada Translator Bahasa Pascal Ke C. *Majalah Ilmiah UNIKOM*, *11*(2), 262–275. https://jurnal.unikom.ac.id/jurnal/penerapan-string-matching.3v

Erlinda, & Masriadi. (2020). Perancangan Aplikasi Mobile Kamus Istilah Komputer untuk Mahasiswa Baru Bidang Ilmu Komputer Berbasis Andorid.

Jurnal Teknologi dan Open Source, *III*(1), 30–43. https://ejournal.uniks.ac.id/index.php/JTOS/article/view/551/390

Fauziah, M. (2019). *Implementasi Elasticsearch untuk Pencarian dan Menetukan Skor Similarity pada Proposal Skripsi di Fakultas Teknologi Informasi Universitas Bale Bandung*. Universitas Bale Bandung.

Halim, I. G. (2016). *Perbandingan Algoritma Boyer Moore dan Brute Force dalam Pembuatan Kamus Bahasa Mandarin - Indonesia - Inggris Platform Android* [Universitas Sumatera Utara]. https://123dok.com/document/1y9p89vq-perbandingan-algoritma-pembuatan-mandarin-indonesia-inggris-platform-android.html?utm_source=search_v3

Hasugian, J. (2003). *Pengantar Kearsipan*. USU Digital Library. https://www.yumpu.com/id/document/read/4155955/pengantar-kearsipan-oleh-drs-jonner-hasugian-msi

Jagad.id. (n.d.). *Download Nox Player : Pengertian, Fitur, Kelebihan dan Kekurangan*. Diambil 2 Maret 2021, dari https://jagad.id/download/nox-player/

Juhara, Z. P. (2016). *Panduan Lengkap Pemrograman ANDROID* (P. S. Wibowo (ed.); 1 ed.). ANDI OFFSET.

Junaedi, M. (2003). *Pengantar XML*. IlmuKomputer.Com. https://www.academia.edu/7666829/Pengantar_XML

Knuth, D. E., Morris (Jr), J. H., & Pratt, V. R. (1977). Fastest pattern matching in strings. *Journal of Algorithms*, *16*(2), 163–189. https://doi.org/10.1006/jagm.1994.1008

Lestari, C. P., Hasibuan, N. A., & Ginting, G. L. (2016). Perancangan Aplikasi Kamus Istilah Medis Berbasis Android Dengan Algoritma Boyer-Moore. *Jurnal INFOTEK*, *II*(3), 1–6. https://ejurnal.stmik-budidarma.ac.id/index.php/komik/article/view/1661/1337

Muntahanah, & Darnita, Y. (2019). Aplikasi Pengarsipan Dokumen Menggunakan Metode String Matching (Studi Kasus Perpustakaan SMP Negeri 5 Seluma).

Jurnal Informatika UPGRIS, *V*(1), 9–16. http://journal.upgris.ac.id/index.php/JIU/article/view/2895/2540

Parlika, R., Nisaa, T. A., Ningrum, S. M., & Haque, B. A. (2020). Studi Literatur Kekurangan dan Kelebihan Pengujian Black Box. *Jurnal Teknomatika*, *X*(2), 131–140. http://ojs.palcomtech.ac.id/index.php/teknomatika/article/view/490/364

Permendagri. (2012). *Lampiran Keputusan Tentang Tata Kearsipan di Lingkungan Kementerian dalam Negeri dan Pemerintah Daerah.* (No. 78). https://123dok.com/document/qo36m20q-lampiran-permendagri-th-kode-surat.html

Pressman, R. S. (2001). *Software Engineering: A Practitioner's Approach* (5 ed.). McGraw-Hil.

Rifqo, M. H., & Andilala. (2020). Implementasi Algoritme Boyer-Moore pada Aplikasi Kamus Istilah Komputer Berbasis Android. *Jurnal Pseudocode*, *VII*(1), 69–77. https://ejournal.unib.ac.id/index.php/pseudocode/article/download/10478/5283

Rusito, N. K. (2019). Aplikasi Pencarian dengan Menggunakan Algoritma Knuth Morris Pratt Pada Berkas Dokumen Shipment. *Jurnal Riset Komputer (JURIKOM)*, *VI*(3), 245–254. http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom%7CPage245

Safaat H., N. (2015). *ANDROID: Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android*. Informatika.

Sari, L. (2019). Perancangan Aplikasi Kamus Bahasa Indonesia ke Bahasa Arab dengan Algoritma Boyer Moore berbasis Android. *Jurnal Pelita Informatika*, *VIII*(2), 189–192. https://ejurnal.stmik-budidarma.ac.id/index.php/pelita/article/view/1820

Setiawan, D. (2007). Mengenal SQLite Command Line. In *IlmuKomputer.Com* (Vol. 3). IlmuKomputer.Com. https://ilmukomputer.org/wp-

content/uploads/2013/09/di2k-SQLite-Command-Line.pdf

Simarmata, J., & Paryudi, I. (2006). *Basis Data* (O. H. Sudiyarto (ed.); 1 ed.). ANDI OFFSET.

Sinaga, J. I., Mesran, & Buulolo, E. (2016). Aplikasi Mobile Pencarian Kata pada Arti Ayat Al-Qur'an berbasis Android menggunakan Algoritma String Matching. *Jurnal INFOTEK*, *II*(2), 68–72. https://ejurnal.amikstiekomsu.ac.id/index.php/infotek/article/view/110/99

Singla, N., & Garg, D. (2012). *String Matching Algorithms and their Applicability in various Applications*. *6*, 218–222. http://dl.icdst.org/pdfs/files/7e7fae2284b8ff50bbbc170e4b1f5e42.pdf

SMK Taruna Bakti. (2013). *Pengenalan StarUML*. https://multimediasmktarunabhakti.wordpress.com/2013/05/22/pengenalan-staruml/

Tasyhar, M. (2013). *Kearsipan 1 Bahan Ajar Kurikulum 2013 Sekolah Menengah Kejuruan Program Keahlian Administrasi Perkantoran* (W. Sucipto & S. Prihatin (ed.)). https://bsd.pendidikan.id/data/2013/kelas_10smk/Kelas_10_SMK_Kearsipan_1.pdf

Utama, G. (2002). *Berfikir Objek: Cara Efektif Menguasai Java*. IlmuKomputer.Com. https://www.academia.edu/12193030/Berfikir_Objek_Cara_Efektif_Menguasai_Java

Verma, H. N., & Singh, R. (2011). *A fast string matching algorithm*. *2*(6), 1877–1883. https://pdffox.com/a-fast-pattern-matching-algorithm-pdf-free.html

Wikipedia. (n.d.-a). *Algoritme Boyer-Moore*. Diambil 25 Juli 2021, dari https://id.wikipedia.org/wiki/Algoritme_Boyer-Moore

Wikipedia. (n.d.-b). *Android (sistem operasi)*. Diambil 18 Maret 2021, dari https://id.wikipedia.org/wiki/Android_(sistem_operasi)

Wikipedia. (n.d.-c). *Java (programming language)*. Diambil 18 Maret 2021, dari https://en.wikipedia.org/wiki/Java_(programming_language)

Wilandari, V. (2018). *Flowchart, Data Flow Diagram (DFD) dan Unified Modelling Language (UML)*. https://vegiwilandari.wordpress.com/2018/11/08/flowchartdata-flow-diagram-dfd-dan-unified-modelling-language-uml/

Yogyawan, H. A. (2016). Implementasi Boyer-Moore Pada Aplikasi Pencarian Rumus Matematika dan Fisika. *Jurnal Ilmiah Teknologi Informasi Terapan (JITTER)*, *III*(1), 74–85. http://journal.widyatama.ac.id/index.php/jitter/article/view/124/114

zonaprogramer. (2016). *Pengertian JDK*. https://zonaprogramer.wordpress.com/2016/05/17/189/

LAMPIRAN

Lampiran 1 Kuesioner: Keperluan Aplikasi

Kuesioner Keperluan Aplikasi

Jawablah pertanyaan dibawah ini dengan cara memberikan tanda √ pada salah satu opsi jawaban yang tersedia

1. Apakah aktivitas dan pekerjaan arsiparis sibuk?
   ☐ Tidak ☑ Terkadang ☐ Selalu
2. Apakah arsip/surat masuk maupun keluar pernah datang bertubi-tubi?
   ☐ Tidak ☑ Terkadang ☐ Selalu
3. Apakah pencarian kode klasifikasi arsip masih manual?
   ☐ Tidak ☑ Ya
4. Apakah ada aplikasi yang membantu pencarian kode klasifikasi arsip?
   ☑ Tidak ☐ Ya
5. Apakah anda membutuhkan aplikasi yang mencari kode klasifikasi arsip?
   ☐ Tidak ☑ Ya
6. Jika dibuatkan aplikasi kode klasifikasi, apakah anda bersedia untuk mengikuti uji coba?
   ☐ Tidak ☑ Ya
7. Apakah setuju jika dibuatkan aplikasi yang mencari kode klasifikasi arsip?
   ☐ Tidak ☑ Ya
8. Apakah setuju jika aplikasi pencarian arsip dibangun pada perangkat elektronik yang sering digunakan?
   ☐ Tidak ☑ Ya
9. Perangkat elektronik apa yang sering digunakan sehari-hari?
   ☐ Komputer ☑ *Smartphone*

Bandung, 27 April 2021

Narasumber

PEMERINTAH KABUPATEN BANDUNG KEC. CIPARAY

(MAESAROH)

Lampiran 2 Kuesioner: Efektifitas dan Kelayakan Aplikasi

Kuesioner Efektifitas dan Kelayakan Penggunaan Aplikasi

Jawablah pertanyaan dibawah ini dengan cara memberikan tanda √ pada salah satu opsi jawaban yang tersedia

1. Apakah tampilan pada aplikasi Klasifikasi Kode Surat bagus?
   ☐ Tidak ☑ Ya
2. Apakah proses pencarian kode surat pada aplikasi cepat?
   ☐ Tidak ☑ Ya
3. Apakah hasil pencarian pada aplikasi terdapat kesalahan atau *error*?
   ☑ Tidak ☐ Ya
4. Apakah aplikasi Klasifikasi Kode Surat membantu dalam pekerjaan anda?
   ☐ Tidak ☑ Ya
5. Apakah aplikasi Kode Klasifikasi Surat layak digunakan?
   ☑ Layak (tanpa saran revisi)
   ☐ Layak, dengan saran _____
   ☐ Tidak layak, karena _____

Bandung, 30 Juni 2021

Narasumber

PEMERINTAH KABUPATEN KEC. CIPARAY BANDUNG

(MAESAROH S.Pd)

Lampiran 3 Listing Program: db_kode_klasifikasi.sqlite

```
CREATE TABLE "klasifikasi" (
	"id_klasifikasi"	INTEGER,
	"kode_klasifikasi"	TEXT NOT NULL,
	"nama_klasifikasi"	TEXT NOT NULL,
	"catatan_klasifikasi"	TEXT,
	PRIMARY KEY("id_klasifikasi" AUTOINCREMENT)
	);
```

Lampiran 4 Listing Program: BoyerMoore.java

```
package com.yosepbahtiar.klasifikasikodesurat;

public class BoyerMoore {
public static int indexOf(char[] text, char[] pattern) {
	if (pattern.length == 0)
	return 0;
	int charTable[] = makeCharTable(pattern);
	int offsetTable[] = makeOffsetTable(pattern);
	for (int i = pattern.length - 1, j; i < text.length;) {
	for (j = pattern.length - 1; pattern[j] == text[i]; --i, --j)
		if (j == 0) return i;
		i += Math.max(offsetTable[pattern.length - 1 - j],
		charTable[text[i]]);
	}
	return -1;
}

private static int[] makeCharTable(char[] pattern) {
	final int ALPHABET_SIZE = 256;
	int[] table = new int[ALPHABET_SIZE];
	for (int i = 0; i < table.length; ++i)
		table[i] = pattern.length;
	for (int i = 0; i < pattern.length - 1; ++i)
		table[pattern[i]] = pattern.length - 1 - i;
	return table;
```

```
}

private static int[] makeOffsetTable(char[] pattern) {
	int[] table = new int[pattern.length];
	int lastPrefixPosition = pattern.length;
	for (int i = pattern.length - 1; i >= 0; --i) {
		if (isPrefix(pattern, i + 1)) lastPrefixPosition = i + 1;
		table[pattern.length - 1 - i] = lastPrefixPosition - i
				+ pattern.length - 1;
	}
	for (int i = 0; i < pattern.length - 1; ++i) {
		int slen = suffixLength(pattern, i);
		table[slen] = pattern.length - 1 - i + slen;
	}
	return table;
}

private static boolean isPrefix(char[] pattern, int p) {
	for (int i = p, j = 0; i < pattern.length; ++i, ++j)
		if (pattern[i] != pattern[j]) return false;
	return true;
}

private static int suffixLength(char[] pattern, int p) {
	int len = 0;
	for (int i = p, j = pattern.length - 1; i >= 0
			&& pattern[i] == pattern[j]; --i, --j)
		len += 1;
	return len;
}
}
```

Lampiran 5 Listing Program: DetailActivity.java

```
package com.yosepbahtiar.klasifikasikodesurat;

import android.app.Activity;
```

```
import android.app.ActionBar;
import android.app.Fragment;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

public class DetailActivity extends Activity {
boolean induk;

@Override
protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.activity_detail);
	if (savedInstanceState == null) {
		getFragmentManager().beginTransaction()
		.add(R.id.container, new PlaceholderFragment()).commit();
	}
	Intent intent = getIntent();
	induk = intent.getBooleanExtra("sebelumnya", true);
	ActionBar actionBar = getActionBar();
	actionBar.setDisplayHomeAsUpEnabled(true);
}

public static class PlaceholderFragment extends Fragment {
	TextView txtTingkat;
	TextView txtKode;
	TextView txtNama;
	TextView txtCatatan;

public PlaceholderFragment() {
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
```

```
        Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_detail,
        container, false);
        txtTingkat = (TextView) rootView.findViewById(R.id.tvTingkat);
        txtKode = (TextView) rootView.findViewById(R.id.tvKode);
        txtNama = (TextView) rootView.findViewById(R.id.tvNama);
        txtCatatan = (TextView) rootView.findViewById(R.id.tvCatatan);
        txtTingkat.setText(getActivity().getIntent().getStringExtra(
                "varTingkat"));
        txtKode.setText(getActivity().getIntent()
                .getStringExtra("varKode"));
        txtNama.setText(getActivity().getIntent()
                .getStringExtra("varNama"));
        txtCatatan.setText(getActivity().getIntent().getStringExtra(
                "varCatatan"));
        return rootView;
        }
}

public boolean onOptionsItemSelected(MenuItem item) {
        if (induk == true) {
        Intent myIntent = new Intent(getApplicationContext(),
        PencarianActivity.class);
        startActivityForResult(myIntent, 0);
        } else {
        Intent myIntent = new Intent(getApplicationContext(),
                PencarianActivity.class);
        myIntent.putExtra("tombol_diklik", true);
        startActivityForResult(myIntent, 0);
}
return true;
}
}
```

Lampiran 6 Listing Program: MainActivity.java

```
package com.yosepbahtiar.klasifikasikodesurat;

import android.app.Activity;
import android.app.ActionBar;
import android.app.Fragment;
import android.app.FragmentManager;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup;
import android.support.v4.widget.DrawerLayout;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity implements
NavigationDrawerFragment.NavigationDrawerCallbacks {
private NavigationDrawerFragment mNavigationDrawerFragment;
private CharSequence mTitle;
protected Cursor cursor;
SQLHelper dbHelper;
Button btnCariKode, btnCariNama;

@Override
protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
       btnCariKode = (Button) findViewById(R.id.btn_cari_kode);
       btnCariNama = (Button) findViewById(R.id.btn_cari_nama);
       mNavigationDrawerFragment = (NavigationDrawerFragment)
              getFragmentManager()
       .findFragmentById(R.id.navigation_drawer);
       mTitle = getTitle();
       mNavigationDrawerFragment.setUp(R.id.navigation_drawer,
```

```
                (DrawerLayout) findViewById(R.id.drawer_layout));
        dbHelper = new SQLHelper(this);
        try {
                dbHelper.createDataBase();
        } catch (Exception ioe) {
                Toast.makeText(getApplicationContext(),
        "Basis Data gagal di-Import :(", Toast.LENGTH_LONG).show();
        }
}

public void methodTombol(View v) {
        Intent intent;
        switch (v.getId()) {
        case R.id.btn_cari_kode:
                intent = new Intent(MainActivity.this,
                        PencarianActivity.class);
                startActivity(intent);
        break;
        case R.id.btn_cari_nama:
                intent = new Intent(MainActivity.this,
                        PencarianActivity.class);
                intent.putExtra("tombol_diklik", true);
                startActivity(intent);
        break;
        default:
        break;
        }
}

@Override
public void onNavigationDrawerItemSelected(int position) {
        FragmentManager fragmentManager = getFragmentManager();
        fragmentManager.beginTransaction().replace(R.id.container,
                PlaceholderFragment.newInstance(position + 1)).commit();
}

public void onSectionAttached(int number) {
        mTitle = getString(R.string.app_name);
```

```
        FragmentManager fragmentManager = getFragmentManager();
        switch (number) {
        case 1:
                break;
        case 2:
                fragmentManager.beginTransaction()
                .replace(R.id.container, new StrukturNomorFragment())
                        .commit();
                break;
        case 3:
                fragmentManager.beginTransaction()
                .replace(R.id.container, new TentangFragment()).commit();
                break;
        }
}

public void restoreActionBar() {
        ActionBar actionBar = getActionBar();
        actionBar.setDisplayShowTitleEnabled(true);
        actionBar.setTitle(mTitle);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
        if (!mNavigationDrawerFragment.isDrawerOpen()) {
                restoreActionBar();
                return true;
        }
        return super.onCreateOptionsMenu(menu);
}

public static class PlaceholderFragment extends Fragment {
        private static final String ARG_SECTION_NUMBER =
                "section_number";
        public static PlaceholderFragment newInstance(int
                sectionNumber) {
        PlaceholderFragment fragment = new PlaceholderFragment();
        Bundle args = new Bundle();
```

```
        args.putInt(ARG_SECTION_NUMBER, sectionNumber);
        fragment.setArguments(args);
        return fragment;
}

public PlaceholderFragment() {
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
              Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_main,
              container, false);
        return rootView;
}

@Override
public void onAttach(Activity activity) {
        super.onAttach(activity);
        ((MainActivity)activity).onSectionAttached(getArguments().getInt(
              ARG_SECTION_NUMBER));
        }
}
}
```

Lampiran 7 Listing Program: NavigationDrawerFragment

```
package com.yosepbahtiar.klasifikasikodesurat;
import android.app.Activity;
import android.app.ActionBar;
import android.app.Fragment;
import android.support.v4.app.ActionBarDrawerToggle;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.content.res.Configuration;
import android.os.Bundle;
import android.view.LayoutInflater;
```

```
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class NavigationDrawerFragment extends Fragment {
private static final String STATE_SELECTED_POSITION
       ="selected_navigation_drawer_position";
private NavigationDrawerCallbacks mCallbacks;
private ActionBarDrawerToggle mDrawerToggle;
private DrawerLayout mDrawerLayout;
private ListView mDrawerListView;
private View mFragmentContainerView;
private int mCurrentSelectedPosition = 0;

public NavigationDrawerFragment() {
}

@Override
public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       if (savedInstanceState != null) {
              mCurrentSelectedPosition = savedInstanceState
                     .getInt(STATE_SELECTED_POSITION);
       }
       selectItem(mCurrentSelectedPosition);
}

@Override
public void onActivityCreated(Bundle savedInstanceState) {
       super.onActivityCreated(savedInstanceState);
       setHasOptionsMenu(true);
}
```

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
       Bundle savedInstanceState) {
       mDrawerListView = (ListView) inflater.inflate(
       R.layout.fragment_navigation_drawer, container, false);
       mDrawerListView.setOnItemClickListener(new AdapterView
              .OnItemClickListener() {
       @Override
       public void onItemClick(AdapterView<?> parent, View view,
              int position, long id) {
       selectItem(position);
                     }
              });
       mDrawerListView.setAdapter(new ArrayAdapter<String>
               (getActionBar().getThemedContext(),
       android.R.layout.simple_list_item_activated_1,
       android.R.id.text1, new String[] {
       getString(R.string.title_section1),
       getString(R.string.title_section2),
       getString(R.string.title_section3), }));
       mDrawerListView.setItemChecked(mCurrentSelectedPosition, true);
       return mDrawerListView;
}

public boolean isDrawerOpen() {
       return mDrawerLayout != null && mDrawerLayout
              .isDrawerOpen(mFragmentContainerView);
}

public void setUp(int fragmentId, DrawerLayout drawerLayout) {
       mFragmentContainerView = getActivity()
              .findViewById(fragmentId);
       mDrawerLayout = drawerLayout;

       mDrawerLayout.setDrawerShadow(R.drawable.drawer_shadow,
              GravityCompat.START);
       ActionBar actionBar = getActionBar();
       actionBar.setDisplayHomeAsUpEnabled(true);
```

```
        actionBar.setHomeButtonEnabled(true);
        mDrawerToggle = new ActionBarDrawerToggle(getActivity(),
        mDrawerLayout,
        R.drawable.ic_drawer,
        R.string.navigation_drawer_open,
        R.string.navigation_drawer_close) {
                @Override
                public void onDrawerClosed(View drawerView) {
                        super.onDrawerClosed(drawerView);
                        if (!isAdded()) {
                                return;
                        }
                        getActivity().invalidateOptionsMenu(); // calls
                }
                @Override
                public void onDrawerOpened(View drawerView) {
                        super.onDrawerOpened(drawerView);
                        if (!isAdded()) {
                                return;
                        }
                getActivity().invalidateOptionsMenu(); // calls
                }
        };

        mDrawerLayout.post(new Runnable() {
                @Override
                public void run() {
                        mDrawerToggle.syncState();
                }
        });
        mDrawerLayout.setDrawerListener(mDrawerToggle);
}

private void selectItem(int position) {
        mCurrentSelectedPosition = position;
        if (mDrawerListView != null) {
                mDrawerListView.setItemChecked(position, true);
        }
```

```
        if (mDrawerLayout != null) {
                mDrawerLayout.closeDrawer(mFragmentContainerView);
        }
        if (mCallbacks != null) {
                mCallbacks.onNavigationDrawerItemSelected(position);
        }
}

@Override
public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
                mCallbacks = (NavigationDrawerCallbacks) activity;
        } catch (ClassCastException e) {
                throw new ClassCastException(
                "Activity must implement NavigationDrawerCallbacks.");
        }
}

@Override
public void onDetach() {
        super.onDetach();
        mCallbacks = null;
}

@Override
public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putInt(STATE_SELECTED_POSITION,
                mCurrentSelectedPosition);
}

@Override
public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
                mDrawerToggle.onConfigurationChanged(newConfig);
}
```

```
@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
	if (mDrawerLayout != null && isDrawerOpen()) {
		showGlobalContextActionBar();
	}
	super.onCreateOptionsMenu(menu, inflater);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
	if (mDrawerToggle.onOptionsItemSelected(item)) {
		return true;
	}
	return super.onOptionsItemSelected(item);
}

private void showGlobalContextActionBar() {
	ActionBar actionBar = getActionBar();
	actionBar.setDisplayShowTitleEnabled(true);
	actionBar.setTitle(R.string.app_name);
}

private ActionBar getActionBar() {
	return getActivity().getActionBar();
}
public static interface NavigationDrawerCallbacks {
	void onNavigationDrawerItemSelected(int position);
}
}
```

Lampiran 8 Listing Program: PencarianActivity.java

```
package com.yosepbahtiar.klasifikasikodesurat;

import java.util.ArrayList;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
```

```
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Toast;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class PencarianActivity extends Activity {
private static boolean diklik;
private static ArrayList<Object> data;
private static ArrayAdapter<Object> adapter;
static ArrayList<Integer> indeks;
static SQLHelper dbHelper;
static ListView listNama;
static EditText editNama;
private static Context mContext;

@Override
protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.activity_pencarian);
	Intent intent = getIntent();
	diklik = intent.getBooleanExtra("tombol_diklik", false);
	if (savedInstanceState == null) {
		getFragmentManager().beginTransaction()
	.add(R.id.container2, new PlaceholderFragment()).commit();
	}
```

```
	if (diklik == true) {
		FragmentManager fragmentManager = getFragmentManager();
		fragmentManager.beginTransaction()
		.replace(R.id.container2, new PencarianNamaFragment())
		.commit();
	}
}

public static class PlaceholderFragment extends Fragment {
public PlaceholderFragment() {
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
		Bundle savedInstanceState) {
	View rootView;
	rootView = inflater.inflate(R.layout.fragment_pencarian_kode,
		container, false);
	mContext = getActivity();
	dbHelper = new SQLHelper(mContext);
	data = new ArrayList<>();
	indeks = new ArrayList<>();
	listNama = (ListView) rootView.findViewById(R.id.LVKode);
	editNama = (EditText) rootView.findViewById(R.id.etCariKode);
	getData();
	return rootView;
}

public void getOutput(int p, Cursor cursor) {
	String tingkat = TingkatKlasifikasi.cekTingkat(p);
	Intent i = new Intent(getActivity(), DetailActivity.class);
	i.putExtra("varTingkat", tingkat);
	i.putExtra("varKode", cursor.getString(1).toString());
	i.putExtra("varNama", cursor.getString(2).toString());
	i.putExtra("varCatatan", cursor.getString(3).toString());
	startActivity(i);
}
```

```
private void getData() {
	SQLiteDatabase db = dbHelper.getReadableDatabase();
	final Cursor cursor = db.rawQuery("SELECT *FROM klasifikasi",
	null);
	if (cursor.getCount() == 0) {
	} else {
	while (cursor.moveToNext()) {
		data.add(cursor.getString(1).toString());
		}
	}
	adapter = new ArrayAdapter<>(getActivity(),
		android.R.layout.simple_list_item_1, data);
	listNama.setAdapter(adapter);
	listNama.setOnItemClickListener(new OnItemClickListener() {
		@Override
		public void onItemClick(AdapterView<?> pa, View v, int p,
				long id) {
			cursor.moveToPosition(p);
			getOutput(p, cursor);
			}
		});
	editNama.addTextChangedListener(new TextWatcher() {
		@Override
		public void onTextChanged(CharSequence arg0, int
			arg1, int arg2, int arg3) {
			int j = 0;
			cursor.moveToFirst();
			adapter.clear();
			for (int i = 0; i < cursor.getCount(); i++) {
				String t = cursor.getString(1);
				String p = arg0.toString();
				char[] text = t.toCharArray();
				char[] pattern = p.toCharArray();
				int pos = BoyerMoore.indexOf(text, pattern);
				if (pos >= 0) {
				data.add(cursor.getString(1).toString());
				indeks.add(j, cursor.getInt(0));
				j = j + 1;
```

```
                }
                cursor.moveToNext();
            }
            if (data.size() == 0) {
            Toast.makeText(mContext, "Data tidak tersedia :(",
                Toast.LENGTH_SHORT).show();
            }
            adapter = new ArrayAdapter<>(getActivity(),
                android.R.layout.simple_list_item_1, data);
            listNama.setAdapter(PencarianActivity.adapter);
            listNama.setOnItemClickListener(new
                OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> pa,
                    View v, int p, long id) {
                    p = indeks.get(p);
                    p = p - 1;
                    cursor.moveToPosition(p);
                    getOutput(p, cursor);
                    }
                });
            }
            @Override
            public void beforeTextChanged(CharSequence arg0, int
                arg1, int arg2, int arg3) { }
            @Override
            public void afterTextChanged(Editable arg0) { }
            });
      }
}
}
```

Lampiran 9 Listing Program: PencarianNamaFragment

```
package com.yosepbahtiar.klasifikasikodesurat;

import java.util.ArrayList;
```

```
import android.app.Fragment;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

public class PencarianNamaFragment extends Fragment {
SQLHelper dbHelper;
ListView listNama;
EditText editNama;
private Context mContext;
private ArrayList<Object> data;
private static ArrayAdapter<Object> adapter;
ArrayList<Integer> indeks;

public PencarianNamaFragment() {
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup
       container, Bundle savedInstanceState) {
       View view = inflater.inflate(R.layout.fragment_
              pencarian_nama, container, false);
       mContext = getActivity();
       dbHelper = new SQLHelper(mContext);
       data = new ArrayList<>();
```

```
        indeks = new ArrayList<>();
        listNama = (ListView) view.findViewById(R.id.LVNama);
        editNama = (EditText) view.findViewById(R.id.etCariNama);
        getData();
        return view;
}

public void getOutput(int p, Cursor cursor) {
        String tingkat = TingkatKlasifikasi.cekTingkat(p);
        Intent i = new Intent(getActivity(), DetailActivity.class);
        i.putExtra("sebelumnya", false);
        i.putExtra("varTingkat", tingkat);
        i.putExtra("varKode", cursor.getString(1).toString());
        i.putExtra("varNama", cursor.getString(2).toString());
        i.putExtra("varCatatan", cursor.getString(3).toString());
        startActivity(i);
}

private void getData() {
        SQLiteDatabase db = dbHelper.getReadableDatabase();
        final Cursor cursor = db.rawQuery("SELECT *FROM
                klasifikasi", null);
        if (cursor.getCount() == 0) {
        } else {
                while (cursor.moveToNext()) {
                data.add(cursor.getString(2).toString());
        }
}

adapter = new ArrayAdapter<>(getActivity(),
android.R.layout.simple_list_item_1, data);
listNama.setAdapter(adapter);
listNama.setOnItemClickListener(new OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> pa, View v, int p,
        long id) {
        cursor.moveToPosition(p);
        getOutput(p, cursor);
```

```
        }
});
editNama.addTextChangedListener(new TextWatcher() {

@Override
public void onTextChanged(CharSequence arg0, int arg1, int
        arg2, int arg3) {
        int j = 0;
        cursor.moveToFirst();
        adapter.clear();
        for (int i = 0; i < cursor.getCount(); i++) {
        String t = cursor.getString(2).toLowerCase();
        String p = arg0.toString().toLowerCase();
        char[] text = t.toCharArray();
        char[] pattern = p.toCharArray();
        int pos = BoyerMoore.indexOf(text, pattern);
                if (pos >= 0) {
                data.add(cursor.getString(2).toString());
                indeks.add(j, cursor.getInt(0));
                j = j + 1;
        }
        cursor.moveToNext();
}
        if (data.size() == 0) {
        Toast.makeText(mContext, "Data tidak tersedia :(",
        Toast.LENGTH_SHORT).show();
        }
        adapter = new ArrayAdapter<>(getActivity(),
        android.R.layout.simple_list_item_1, data);
        listNama.setAdapter(adapter);
        listNama.setOnItemClickListener(new
                OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> pa, View
                        v, int p, long id) {
                        p = indeks.get(p);
                        p = p - 1;
                        cursor.moveToPosition(p);
```

```
                    getOutput(p, cursor);
                    }
              });
       }
              @Override
              public void beforeTextChanged(CharSequence arg0,
                    int arg1, int arg2, int arg3) { }
              @Override
              public void afterTextChanged(Editable arg0) { }
       });
}
}
```

Lampiran 10 Listing Program: SQLHelper.java

```
package com.yosepbahtiar.klasifikasikodesurat;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;
import android.widget.Toast;

public class SQLHelper extends SQLiteOpenHelper {
private static final String DATABASE_NAME =
       "db_kode_klasifikasi.sqlite";
private static final int DATABASE_VERSION = 1;

private static String DB_PATH =
"/data/data/com.yosepbahtiar.klasifikasikodesurat/databases/";
public static final String TABLE = "list_kode_utama";
private Context myContext;
public SQLHelper(Context context) {
```

```
	super(context, DATABASE_NAME, null, DATABASE_VERSION);
	myContext = context;
}

public void createDataBase() throws IOException {
	if (DataBaseisExist()) {
	} else {
	this.getReadableDatabase();
	try {
	copyDataBase();
		Toast.makeText(myContext,
			"Basis Data berhasil di-Import :)",
		Toast.LENGTH_LONG).show();
	} catch (IOException e) {
		throw new Error("Error saat menyalin Basis Data :(");
		}
	}
}

private boolean DataBaseisExist() {
	SQLiteDatabase checkDB = null;
	try {
	String myPath = DB_PATH + DATABASE_NAME;
	checkDB = SQLiteDatabase.openDatabase(myPath, null,
		SQLiteDatabase.OPEN_READONLY);
	} catch (SQLiteException e) {
	}
	if (checkDB != null) {
		checkDB.close();
	}
	if (checkDB != null)
		return true;
	else
		return false;
}

private void copyDataBase() throws IOException {
	InputStream myInput = myContext.getAssets()
```

```
              .open(DATABASE_NAME);
       String outFileName = DB_PATH + DATABASE_NAME;
       OutputStream myOutput = new FileOutputStream(outFileName);
       byte[] buffer = new byte[1024];
       int length;
       while ((length = myInput.read(buffer)) > 0) {
              myOutput.write(buffer, 0, length);
       }
       myOutput.flush();
       myOutput.close();
       myInput.close();
}

@Override
public void onCreate(SQLiteDatabase db) {
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion,
       int newVersion) {
}
}
```

Lampiran 11 Listing Program: StrukturNomorFragment.java

```
package com.yosepbahtiar.klasifikasikodesurat;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class StrukturNomorFragment extends Fragment {
public StrukturNomorFragment() { }

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
```

```
        Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_struktur_nomor,
                container, false);
        return view;
}
}
```

Lampiran 12 Listing Program: TentangFragment.java

```
package com.yosepbahtiar.klasifikasikodesurat;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class TentangFragment extends Fragment {
public TentangFragment() { }

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_tentang,
                container, false);
        return view;
}
}
```

Lampiran 13 Listing Program: TingkatKlasifikasi.java

```
package com.yosepbahtiar.klasifikasikodesurat;
public class TingkatKlasifikasi {
public static String cekTingkat(int p) {
String tingkat = "";
        // keuangan
        if (p >= 2327) {
```

```
            tingkat = "Keuangan\\Bendaharawan";
      } else if (p >= 2314) {
            tingkat = "Keuangan\\Pendapatan";
      } else if (p >= 2307) {
            tingkat = "Keuangan\\Pembinaan Kebendaharaan";
      } else if (p >= 2297) {
            tingkat = "Keuangan\\Perbendaharaan";
      } else if (p >= 2291) {
            tingkat = "Keuangan\\Pembukuan";
      } else if (p >= 2281) {
            tingkat = "Keuangan\\Verifikasi";
      } else if (p >= 2273) {
            tingkat = "Keuangan\\Otorisasi/SKO";
      } else if (p >= 2261) {
            tingkat = "Keuangan\\Anggaran";
      } else if (p >= 2251) {
            tingkat = "Keuangan";
      }
      // kepegawaian
      else if (p >= 2205) {
            tingkat = "Kepegawaian\\Pendidikan Pegawai";
      } else if (p >= 2187) {
            tingkat = "Kepegawaian\\Pemberhentian Pegawai";
      } else if (p >= 2167) {
            tingkat = "Kepegawaian\\Tata Usaha Kepegawaian";
      } else if (p >= 2145) {
            tingkat = "Kepegawaian\\Penilaian";
      } else if (p >= 2135) {
            tingkat = "Kepegawaian\\Cuti";
      } else if (p >= 2103) {
            tingkat = "Kepegawaian\\Kesejahteraan Pegawai";
      } else if (p >= 2091) {
            tingkat = "Kepegawaian\\Kedudukan";
      } else if (p >= 2046) {
            tingkat = "Kepegawaian\\Mutasi";
      } else if (p >= 2027) {
            tingkat = "Kepegawaian\\Pengadaan";
      } else if (p >= 2020) {
```

```
		tingkat = "Kepegawaian";
	}
	// pengawasan
	else if (p >= 2010) {
		tingkat = "Pengawasan\\Bidang Keuangan";
	} else if (p >= 2000) {
		tingkat = "Pengawasan\\Bidang Kepegawaian";
	} else if (p >= 1987) {
		tingkat = "Pengawasan\\Bidang Pekerjaan Umum";
	} else if (p >= 1977) {
		tingkat = "Pengawasan\\Bidang Perekonomian";
	} else if (p >= 1967) {
		tingkat = "Pengawasan\\Bidang Kesejahteraan Rakyat";
	} else if (p >= 1957) {
		tingkat = "Pengawasan\\Bidang Keamanan/Ketertiban";
	} else if (p >= 1949) {
		tingkat = "Pengawasan\\Bidang Politik";
	} else if (p >= 1939) {
		tingkat = "Pengawasan\\Bidang Pemerintahan";
	} else if (p >= 1929) {
		tingkat = "Pengawasan";
	}

	// pekerjaan umum dan ketenagaan
	else if (p >= 1883) {
		tingkat = "Pekerjaan Umum dan Ketenagaan\\Air Minum";
	} else if (p >= 1879) {
		tingkat = "Pekerjaan Umum dan Ketenagaan\\Peralatan";
	} else if (p >= 1846) {
		tingkat = "Pekerjaan Umum dan Ketenagaan\\Ketenagaan";
	} else if (p >= 1829) {
		tingkat = "Pekerjaan Umum dan Ketenagaan\\Tata
			Lingkungan";
	} else if (p >= 1788) {
		tingkat = "Pekerjaan Umum dan Ketenagaan\\Tata Kota";
	} else if (p >= 1728) {
		tingkat = "Pekerjaan Umum dan Ketenagaan\\Bangunan";
	} else if (p >= 1620) {
```

```
            tingkat = "Pekerjaan Umum dan Ketenagaan\\Jembatan";
      } else if (p >= 1526) {
            tingkat = "Pekerjaan Umum dan Ketenagaan\\Jalan";
      } else if (p >= 1375) {
            tingkat = "Pekerjaan Umum dan Ketenagaan\\Pengairan";
      } else if (p >= 1334) {
            tingkat = "Pekerjaan Umum dan Ketenagaan";
      }
      // perekonomian
      else if (p >= 1252) {
            tingkat = "Perekonomian\\Agraria";
      } else if (p >= 1242) {
            tingkat = "Perekonomian\\Perbankan/Moneter";
      } else if (p >= 1234) {
            tingkat = "Perekonomian\\Permodalan";
      } else if (p >= 1221) {
            tingkat = "Perekonomian\\Tenaga Kerja";
      } else if (p >= 1169) {
            tingkat = "Perekonomian\\Perhubungan";
      } else if (p >= 1147) {
            tingkat = "Perekonomian\\Pertambangan/Kesamudraan";
      } else if (p >= 1133) {
            tingkat = "Perekonomian\\Perindustrian";
      } else if (p >= 1023) {
            tingkat = "Perekonomian\\Pertanian";
      } else if (p >= 1004) {
            tingkat = "Perekonomian\\Perdagangan";
      } else if (p >= 996) {
            tingkat = "Perekonomian";
      }
      // kesejahteraan
      else if (p >= 993) {
            tingkat = "Kesejahteraan\\Pengaduan Masyarakat";
      } else if (p >= 972) {
            tingkat = "Kesejahteraan\\Media Massa";
      } else if (p >= 853) {
            tingkat = "Kesejahteraan\\Kependudukan";
      } else if (p >= 817) {
```

```
            tingkat = "Kesejahteraan\\Sosial";
      } else if (p >= 764) {
            tingkat = "Kesejahteraan\\Agama";
      } else if (p >= 717) {
            tingkat = "Kesejahteraan\\Kesehatan";
      } else if (p >= 700) {
            tingkat = "Kesejahteraan\\Kebudayaan";
      } else if (p >= 651) {
            tingkat = "Kesejahteraan\\Pendidikan";
      } else if (p >= 538) {
            tingkat = "Kesejahteraan\\Pembangunan Desa";
      } else if (p >= 533) {
            tingkat = "Kesejahteraan";
      }
      // keamanan dan ketertiban
      else if (p >= 517) {
            tingkat = "Keamanan dan Ketertiban\\Kecelakaan/Sar";
      } else if (p >= 507) {
            tingkat = "Keamanan dan Ketertiban\\Bencana";
      } else if (p >= 497) {
            tingkat = "Keamanan dan Ketertiban\\Kejahatan";
      } else if (p >= 492) {
            tingkat = "Keamanan dan Ketertiban\\Pertahanan Sipil";
      } else if (p >= 478) {
            tingkat = "Keamanan dan Ketertiban\\Keamanan";
      } else if (p >= 468) {
            tingkat = "Keamanan dan Ketertiban\\Kemiliteran";
      } else if (p >= 460) {
            tingkat = "Keamanan dan Ketertiban\\Pertahanan";
      } else if (p >= 456) {
            tingkat = "Keamanan dan Ketertiban";
      }
      // politik
      else if (p >= 451) {
            tingkat = "Politik\\Pengucapan Sumpah Janji MPR, DPR,
                  DPD";
      } else if (p >= 436) {
            tingkat = "Politik\\Pemilihan Umum";
```

```
	} else if (p >= 426) {
		tingkat = "Politik\\Organisasi Wanita";
	} else if (p >= 417) {
		tingkat = "Politik\\Organisasi Buruh, Tani, Nelayan, dan
			Angkutan";
	} else if (p >= 408) {
		tingkat = "Politik\\Organisasi Pemuda";
	} else if (p >= 398) {
		tingkat = "Politik\\Organisasi Profesi dan Fungsional";
	} else if (p >= 386) {
		tingkat = "Politik\\Organisasi Kemasyarakatan";
	} else if (p >= 379) {
		tingkat = "Politik\\Kepartaian";
	} else if (p >= 372) {
		tingkat = "Politik";
	}
	// pemerintahan
	else if (p >= 360) {
		tingkat = "Pemerintahan\\Hubungan Luar Negeri";
	} else if (p >= 310) {
		tingkat = "Pemerintahan\\Hukum";
	} else if (p >= 294) {
		tingkat = "Pemerintahan\\DPRD Kabupaten";
	} else if (p >= 276) {
		tingkat = "Pemerintahan\\DPRD Provinsi";
	} else if (p >= 258) {
		tingkat = "Pemerintahan\\Legislatif";
	} else if (p >= 237) {
		tingkat = "Pemerintahan\\Pemerintahan Desa/Kelurahan";
	} else if (p >= 208) {
		tingkat = "Pemerintahan\\Pemerintah Kabupaten/Kota";
	} else if (p >= 184) {
		tingkat = "Pemerintahan\\Pemerintah Provinsi";
	} else if (p >= 166) {
		tingkat = "Pemerintahan\\Pemerintahan Pusat";
	} else if (p >= 160) {
		tingkat = "Pemerintahan";
	}
```

```
        // umum
        else if (p >= 150) {
                tingkat = "Umum\\Perjalanan Dinas";
        } else if (p >= 140) {
                tingkat = "Umum\\Konferensi/Rapat/Seminar";
        } else if (p >= 130) {
                tingkat = "Umum\\Penelitian";
        } else if (p >= 117) {
                tingkat = "Umum\\Organisasi Ketatalaksanaan";
        } else if (p >= 103) {
                tingkat = "Umum\\Perencanaan";
        } else if (p >= 74) {
                tingkat = "Umum\\Perpustakaan
                        Dokumentasi/Kearsipan/Sandi";
        } else if (p >= 67) {
                tingkat = "Umum\\Kekayaan Daerah";
        } else if (p >= 56) {
                tingkat = "Umum\\Peralatan";
        } else if (p >= 33) {
                tingkat = "Umum\\Urusan Dalam";
        } else if (p >= 0) {
                tingkat = "Umum";
        }
        return tingkat;
}
}
```

Lampiran 14 Listing Program: activity_detail.xml

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.yosepbahtiar.klasifikasikodesurat
        .DetailActivity"
```

```
    tools:ignore="MergeRootFrame" />
```

Lampiran 15 Listing Program: activity_main.xml

```
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#f6f6f6"
    tools:context="com.yosepbahtiar.klasifikasikodesurat
        .MainActivity" >

    <FrameLayout
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <fragment
        android:id="@+id/navigation_drawer"
        android:name="com.yosepbahtiar.klasifikasikodesurat
        .NavigationDrawerFragment"
        android:layout_width="@dimen/navigation_drawer_width"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        tools:layout="@layout/fragment_navigation_drawer" />
</android.support.v4.widget.DrawerLayout>
```

Lampiran 16 Listing Program: activity_pencarian.xml

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#f6f6f6"
```

```
    tools:context="com.yosepbahtiar.klasifikasikodesurat.
        PencarianActivity"
    tools:ignore="MergeRootFrame" />
```

Lampiran 17 Listing Program: fragment_detail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scrollView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.yosepbahtiar.klasifikasikodesurat
    .DetailActivity$PlaceholderFragment" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin" >

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:gravity="center_horizontal"
            android:text="@string/title_activity_detail"
            android:textAppearance="
            ?android:attr/textAppearanceLarge" />

        <TextView
            android:id="@+id/tvTingkat"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```
            android:layout_marginTop="15dp"
            android:gravity="start"
            android:textAppearance="
            ?android:attr/textAppearanceMedium"
            android:textColor="#FF0F0F"
            android:textSize="15sp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:gravity="start"
            android:text="@string/detail_kode"
            android:textAppearance="
            ?android:attr/textAppearanceMedium"
            android:textColor="#0000FF"
            android:textSize="14sp" />

        <TextView
            android:id="@+id/tvKode"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="10dp"
            android:layout_marginTop="7dp"
            android:gravity="start"
            android:textAppearance="
            ?android:attr/textAppearanceMedium"
            android:textSize="19sp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:gravity="start"
            android:text="@string/detail_nama"
            android:textAppearance="
            ?android:attr/textAppearanceMedium"
            android:textColor="#0000FF"
```

```
            android:textSize="14sp" />

        <TextView
            android:id="@+id/tvNama"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="10dp"
            android:layout_marginTop="7dp"
            android:gravity="start"
            android:textAppearance="
            ?android:attr/textAppearanceMedium"
            android:textSize="18sp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:gravity="start"
            android:text="@string/detail_catatan"
            android:textAppearance="
            ?android:attr/textAppearanceMedium"
            android:textColor="#0000FF"
            android:textSize="14sp" />

        <TextView
            android:id="@+id/tvCatatan"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="10dp"
            android:layout_marginTop="7dp"
            android:gravity="start"
            android:textAppearance="
            ?android:attr/textAppearanceMedium"
            android:textSize="18sp" />
    </LinearLayout>
</ScrollView>
```

Lampiran 18 Listing Program: fragment_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scrollView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.yosepbahtiar.klasifikasikodesurat
    .MainActivity$PlaceholderFragment" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin" >

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal" >

            <ImageView
                android:id="@+id/imageView1"
                android:layout_width="80dp"
                android:layout_height="80dp"
                android:layout_centerHorizontal="true"
                android:contentDescription="@string/logo"
                android:gravity="center_horizontal|center_vertical"
                android:src="@drawable/logo" />

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
```

```
            android:layout_below="@id/imageView1"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="15dp"
            android:gravity="center_horizontal|center_vertical"
            android:text="@string/app_name"
            android:textAppearance="?android:
            attr/textAppearanceMedium" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/textView1"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="15dp"
            android:gravity="center_horizontal|center_vertical"
            android:text="@string/info_tentang_aplikasi"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <TextView
            android:id="@+id/textView4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/textView2"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="15dp"
            android:gravity="center_horizontal|center_vertical"
            android:text="@string/sumber_data"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="12sp" />
    </RelativeLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="24dp"
            android:background="@drawable/border"
            android:gravity="center_horizontal"
            android:orientation="vertical"
            android:paddingBottom="20dp"
            android:paddingLeft="5dp"
            android:paddingRight="5dp"
            android:paddingTop="20dp" >

            <TextView
                android:id="@+id/textView3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="center_horizontal|center_vertical"
                android:text="@string/cara_pakai_aplikasi"
                android:textAppearance="?android:
                attr/textAppearanceSmall" />

            <Button
                android:id="@+id/btn_cari_kode"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:background="@drawable/btn_rounded"
                android:clickable="true"
                android:gravity="center_horizontal|center_vertical"
                android:onClick="methodTombol"
                android:paddingLeft="13dp"
                android:paddingRight="13dp"
                android:text="@string/tombol_cari_kode"
                android:textSize="15sp" />

            <Button
                android:id="@+id/btn_cari_nama"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
```

```
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:background="@drawable/btn_rounded"
                android:clickable="true"
                android:gravity="center_horizontal|center_vertical"
                android:onClick="methodTombol"
                android:paddingLeft="13dp"
                android:paddingRight="13dp"
                android:text="@string/tombol_cari_nama"
                android:textSize="15sp" />
        </LinearLayout>
    </LinearLayout>
</ScrollView>
```

Lampiran 19 Listing Program: fragment_navigation_drawer.xml

```
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFFFFF"
    android:choiceMode="singleChoice"
    android:divider="#888888"
    android:dividerHeight="2dp"
    tools:context="com.yosepbahtiar.klasifikasikodesurat
    .NavigationDrawerFragment" />
```

Lampiran 20 Listing Program: fragment_pencarian_kode.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
```

```
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.yosepbahtiar.klasifikasikodesurat
    .PencarianActivity$PlaceholderFragment" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/berdasarkan_kode"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/etCariKode"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignEnd="@+id/textView1"
        android:layout_alignStart="@+id/textView1"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="18dp"
        android:ems="10"
        android:hint="@string/hint_masukan_kode"
        android:inputType="numberDecimal"
        android:textColor="#000000" >
        <requestFocus />
    </EditText>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/etCariKode"
        android:orientation="vertical" >

        <ListView
            android:id="@+id/LVKode"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:scrollbars="vertical" >
        </ListView>
```

```
    </LinearLayout>
</RelativeLayout>
```

Lampiran 21 Listing Program: fragment_pencarian_nama.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.yosepbahtiar.klasifikasikodesurat
        .PencarianNamaFragment" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/berdasarkan_nama"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/etCariNama"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignEnd="@+id/textView1"
        android:layout_alignStart="@+id/textView1"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="18dp"
        android:ems="10"
        android:hint="@string/hint_masukan_nama"
        android:inputType="textPersonName"
        android:textColor="#000000" >
```

```
        <requestFocus />
    </EditText>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/etCariNama"
        android:orientation="vertical" >

        <ListView
            android:id="@+id/LVNama"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:scrollbars="vertical" >
        </ListView>
    </LinearLayout>
</RelativeLayout>
```

Lampiran 22 Listing Program: fragment_struktur_nomor.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scrollView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.yosepbahtiar.klasifikasikodesurat
        .StrukturNomorFragment" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin" >
```

```
        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:gravity="center_horizontal"
            android:text="@string/title_section2"
            android:textAppearance="?android:
            attr/textAppearanceLarge" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="18dp"
            android:gravity="start"
            android:text="@string/surat_umumnya"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="2dp"
            android:gravity="start"
            android:text="@string/surat_a"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="2dp"
            android:gravity="start"
            android:text="@string/surat_b"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
```

```
            android:textSize="14sp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="2dp"
            android:gravity="start"
            android:text="@string/surat_c"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="2dp"
            android:gravity="start"
            android:text="@string/surat_d"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="2dp"
            android:gravity="start"
            android:text="@string/surat_e"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:gravity="start"
            android:text="@string/surat_biasanya"
```

```
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="2dp"
            android:gravity="start"
            android:text="@string/surat_contoh"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:background="@drawable/border_2"
            android:gravity="start"
            android:paddingTop="8dp"
            android:text="@string/surat_untuk"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <ImageView
            android:layout_width="290dp"
            android:layout_height="160dp"
            android:layout_gravity="center"
            android:layout_marginTop="15dp"
            android:contentDescription="@string/title_section2"
            android:src="@drawable/contoh_1" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="4dp"
```

```
            android:layout_marginBottom="10dp"
            android:gravity="center_horizontal|center"
            android:text="@string/surat_cth_1"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <ImageView
            android:layout_width="290dp"
            android:layout_height="160dp"
            android:layout_gravity="center"
            android:layout_marginTop="15dp"
            android:contentDescription="@string/title_section2"
            android:src="@drawable/contoh_2" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="4dp"
            android:layout_marginBottom="10dp"
            android:gravity="center_horizontal|center"
            android:text="@string/surat_cth_2"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <ImageView
            android:layout_width="290dp"
            android:layout_height="160dp"
            android:layout_gravity="center"
            android:layout_marginTop="15dp"
            android:contentDescription="@string/title_section2"
            android:src="@drawable/contoh_3" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="4dp"
```

```
            android:gravity="center_horizontal|center"
            android:text="@string/surat_cth_3"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />
    </LinearLayout>
</ScrollView>
```

Lampiran 23 Listing Program: fragment_tentang.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scrollView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.yosepbahtiar.klasifikasikodesurat
    .TentangFragment" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin" >

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="@string/title_section3"
            android:textAppearance="?android:
            attr/textAppearanceLarge" />
```

```
        <ImageView
            android:id="@+id/imageView1"
            android:layout_width="160dp"
            android:layout_height="160dp"
            android:layout_gravity="center"
            android:layout_marginTop="18dp"
            android:contentDescription="@string/title_section3"
            android:src="@drawable/saya" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:background="@drawable/border_2"
            android:gravity="start"
            android:paddingTop="8dp"
            android:text="@string/info_tentang_saya"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <TextView
            android:id="@+id/textView4"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:gravity="start"
            android:text="@string/info_tentang_aplikasi_2"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:gravity="center_horizontal|center_vertical"
            android:text="@string/dospem"
```

```
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal|center_vertical"
            android:text="@string/dospem_1"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal|center_vertical"
            android:text="@string/dospem_2"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal|center_vertical"
            android:text="@string/narasumber"
            android:textAppearance="?android:
            attr/textAppearanceMedium"
            android:textSize="14sp" />
    </LinearLayout>
</ScrollView>
```

Lampiran 24 Listing Program: strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
```

```
    <!-- string activity/fragment halaman utama -->
    <string name="app_name">Klasifikasi Kode Surat</string>
    <string name="title_section1">Halaman Utama</string>
    <string name="info_tentang_aplikasi">Aplikasi untuk mencari kode
klasifikasi pada arsip/surat di lingkungan pemerintahan</string>
    <string name="cara_pakai_aplikasi">Untuk mulai mencari kode
arsip/surat, silakan tekan salah satu pilihan yang tersedia dibawah
ini</string>
    <string name="sumber_data">Basis data klasifikasi kode arsip/surat
dalam aplikasi ini mengacu dan bersumber dari Lampiran Keputusan
Menteri Dalam Negeri Republik Indonesia no. 78 tahun 2012 tentang Tata
Kearsipan di lingkungan Kementerian Dalam Negeri dan Pemerintah
Daerah, serta Peraturan Menteri Dalam Negeri Republik Indonesia no.
135 tahun 2017 tentang Perubahan pada peraturan sebelumnya.</string>
    <string name="navigation_drawer_open">Open navigation
drawer</string>
    <string name="navigation_drawer_close">Close navigation
drawer</string>
    <string name="logo">Logo Aplikasi Klasifikasi Surat</string>
    <string name="tombol_cari_kode">Berdasarkan Kode Surat</string>
    <string name="tombol_cari_nama">Berdasarkan Nama Surat</string>

    <!-- string activity/fragment pencarian nama & kode -->
    <string name="berdasarkan_kode">Berdasarkan Kode:</string>
    <string name="hint_masukan_kode">Masukan Klasifikasi Kode
Surat</string>
    <string name="berdasarkan_nama">Berdasarkan Nama:</string>
    <string name="hint_masukan_nama">Masukan Klasifikasi Nama
Surat</string>
    <string name="title_activity_pencarian">Pencarian</string>

    <!-- string fragment bagian surat (struktur nomor surat) -->
    <string name="title_section2">Struktur Nomor Surat</string>
    <string name="surat_umumnya">Secara umum, struktur nomor surat
dinas resmi biasanya terdiri dari:</string>
    <string name="surat_a">(a) Kode surat, kode yang memiliki pola
klasifikasi berdasarkan subjek/pokok masalah, berupa angka
desimal.</string>
```

```
    <string name="surat_b">(b) Nomor urutan surat yang
dikeluarkan.</string>
    <string name="surat_c">(c) Nama lembaga/unit kerja (disebut juga
komponen) yang mengeluarkan surat, unit kerja ini berupa singkatan
atau akronim.</string>
    <string name="surat_d">(d) Bulan saat surat dikeluarkan
(dituliskan dengan angka romawi).</string>
    <string name="surat_e">(e) Tahun saat surat dikeluarkan.</string>
    <string name="surat_biasanya">Biasanya ke 5 komponen tersebut
ditulis berurutan dengan menggunakan tanda (‘/’) sebagai pemisah.
Berikut adalah struktur penulisannya (ditiap instansi pemerintahan
mungkin dapat berbeda):</string>
    <string name="surat_contoh">Nomor: (a) / (b) / (c) / (d) /
(e)</string>
    <string name="surat_untuk">Untuk lebih memahami tentang struktur
penulisan nomor surat dinas, dibawah telah disediakan contoh surat
dari beberapa instansi pemerintah:</string>
    <string name="surat_cth_1">MENDAGRI (Setjen) perihal Penerapan
Protokol Kesehatan</string>
    <string name="surat_cth_2">PemKab Bandung (Disnaker) perihal
Pelatihan Bahasa Asing</string>
    <string name="surat_cth_3">MENDAGRI (Setjen) perihal Penghentian
Pemotongan Tabungan Perumahan</string>

    <!-- string fragment tentang -->
    <string name="title_section3">Tentang Pengembang</string>
    <string name="info_tentang_saya">Yosep Bahtiar, seorang mahasiswa
Universitas Bale Bandung jurusan Teknik Informatika yang akhir-akhir
ini gemar menghabiskan waktunya mempelajari bahasa pemrograman Java
dan XML di Android.</string>
    <string name="dospem_1">Pembimbing [1]: Rustiyana, S.T, M.T,
M.Pd</string>
    <string name="dospem_2">Pembimbing [2]: Yusuf Muharam, S.Kom,
M.Kom</string>
    <string name="info_tentang_aplikasi_2">Aplikasi ini dibangun
terkait dengan penelitian yang dilakukan olehnya sebagai pengembang
dan peneliti, yang tujuan penelitiannya untuk mengimplementasikan
```

```
algoritma Boyer-Moore sebagai algoritma pencocokan string (kalimat)
dalam aplikasi pencarian Klasifikasi Surat ini.</string>
    <string name="dospem">Dalam penelitian ini, ia dibimbing
oleh:</string>
    <string name="narasumber">Narasumber: N. Maesaroh</string>

    <!-- string fragment tentang -->
    <string name="title_activity_detail">Detail Surat</string>
    <string name="detail_kode">Kode Surat: </string>
    <string name="detail_nama">Nama Surat: </string>
    <string name="detail_catatan">Catatan: </string>
</resources>
```

Lampiran 25 Listing Program: styles.xml

```
<resources xmlns:android="http://schemas.android.com/apk/res/android">
    <style name="AppBaseTheme" parent="android:Theme.Light">
    </style>
    <style name="AppTheme" parent="AppBaseTheme">
    </style>
</resources>
```

Lampiran 26 Listing Program: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.yosepbahtiar.klasifikasikodesurat"
    android:versionCode="2"
    android:versionName="2.3.7" >
    <uses-sdk
        android:minSdkVersion="19"
        android:targetSdkVersion="29" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
```

```
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.
                category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".PencarianActivity"
            android:label="@string/title_activity_pencarian"
            android:parentActivityName=".MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.yosepbahtiar.klasifikasikodesurat.
                MainActivity.PlaceholderFragment" />
        </activity>
        <activity
            android:name=".DetailActivity"
            android:label="@string/title_activity_detail" >
        </activity>
    </application>
</manifest>
```

Lampiran 27 Listing Program: border_2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android
.com/apk/res/android" >
    <item>
        <shape android:shape="rectangle" >
            <stroke
                android:width="1dp"
                android:color="#000000" />
            <solid android:color="#f6f6f6" />
        </shape>
```

```
    </item>
    <item
        android:top="1dp">
        <shape android:shape="rectangle" >
            <solid android:color="#f6f6f6" />
        </shape>
    </item>
</layer-list>
```

Lampiran 28 Listing Program: border.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:padding="10dp"
    android:shape="rectangle">
   <solid android:color="#ffffff" />
   <corners
        android:bottomLeftRadius="25dp"
        android:bottomRightRadius="25dp"
        android:topLeftRadius="25dp"
        android:topRightRadius="25dp" />
   <stroke
        android:color="#23B14D"
        android:width="2dp" />
</shape>
```

Lampiran 29 Listing Program: btn_rounded.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:padding="10dp"
    android:shape="rectangle">
    <solid android:color="#f6f6f6" />
    <corners
        android:bottomLeftRadius="25dp"
        android:bottomRightRadius="25dp"
        android:topLeftRadius="25dp"
```

```
        android:topRightRadius="25dp" />
    <stroke
        android:color="#777777"
        android:width="1dp" />
</shape>
```

RIWAYAT HIDUP

Yosep Bahtiar, lahir di Bandung pada tanggal 19 Juni 1999. Anak pertama dari tiga bersaudara, yang lahir dari pasangan Ahmad Fauzi dan Iin Parlina. Mulai mengenyam pendidikan tingkat dasar di SDN Bugel 1 (2005-2009) dan pindah ke SDN Gunungleutik III (2009-2011), kemudian melanjutkan pendidikan di SMP Karya Pembangunan Ciparay (2011-2014), serta melanjutkan masa SMK di SMK Wirakarya 1 Ciparay (2014-2017). Untuk mendapatkan gelar sarjana, Yosep melanjutkan ke jenjang S1 di Universitas Bale Bandung Fakultas Teknologi Informasi Jurusan Teknik Informatika.