APLIKASI PENGELOLAAN SURAT KELUAR MENGGUNAKAN *SEQUENTIAL SEARCH* DAN *SELECTION SORT* PADA KPU KOTA BLITAR

Nilal Haming, Sri Lestanti, Saiful Nur Budiman
Program Studi Teknik Informatika S1, Fakultas Teknologi Informasi
Universitas Islam Balitar, Jalan Majapahit Blitar, Indonesia
amingperdana29@gmail.com

ABSTRAK

Penggunaan sistem informasi bisa mencakup berbagai hal termasuk sebagai media inventaris kantor agar pekerjaan semakin mudah dan cepat. Kantor Komisi Pemilihan Umum (KPU) sangat membutuhkan sebuah sistem yang dapat membantu menyelesaikan pekerjaan menyangkut persuratan terutama surat keluar karena Bagian Umum di KPU Kota Blitar mengalami kendala dalam melakukan penomoran surat, pemilihan lokasi penyimpanan dokumen, permintaan persetujuan atasan dan penyisipan tanda tangan digital masih dilakukan secara manual. Karena selama ini Kantor KPU Kota Blitar tidak pernah dilakukan proses *sorting* dan *searching* dokumen secara otomatis maka perlu diadakan pembangunan sistem informasi pengarsipan surat keluar yang dapat melakukan pembuatan, penyuntingan, penghapusan dan penyimpanan surat secara elektonik. Aplikasi pengelola surat keluar menggunakan *sequential search* dan *selection sort* pada KPU Kota Blitar bisa dibuat dengan menerapkan alur *Software Development Life Cycle* (*SDLC*) yang diimplementasikan menggunakan bahasa pemrograman *web* pada *framework Laravel*. Konversi file dari tampilan *web* menjadi *DOCX* dapat dilakukan dengan pemformatan file menggunakan *library* yang ada dalam *Laravel*. Secara umum Aplikasi yang dibuat sudah memenuhi kebutuhan fungsional yang telah di desain meliputi kelola surat perintah, kelola surat undangan, minta persetujuan, memberi persetujuan dan kelola user. Metode *selection sort* telah diuji melalui pengoperasian langsung dari aplikasi dan bisa menghasilkan *sorting* yang baik dengan total 6 proses dan lebih mempercepat komputasi disbanding metode *bubble sort* yang menghasilkan 9 *proses*. Begitu juga dengan pengujian metode *sequential search* yang menghasilkan 4 proses komputasi dibanding binary search yang menghasilkan 11 proses komputasi dengan data yang sama. Pengujian *Black Box* juga diterapkan dengan hasil yang sangat baik yaitu mencapai 95% untuk kesesuaian dan 90% untuk tingkat kenormalan aplikasi.

Kata kunci: Surat Keluar, *Selection Sort*, *Sequential Search*, *SDLC*, *Laravel*, *Black Box Testing*

1. PENDAHULUAN

Perkembangan teknologi informasi hingga saat ini menyebabkan perubahan distribusi informasi dari konvensional menjadi berbasis elektronik. Penggunaan komputer terjadi hampir pada semua aspek kehidupan. Pada umumnya komputer digunakan untuk melakukan pengelolaan data, menyimpan berkas, menulis dokumen, melakukan perhitungan dan lain sebagainya (Lubis dkk., 2020).

Salah satu kebutuhan terbesar saat ini adalah kebutuhan akan sistem informasi. Sistem informasi yang baik akan membuat seseorang menjadi lebih produktif karena pekerjaan yang dilakukan semakin mudah dan cepat (Anggraeni, 2017). Pembuatan sistem informasi mengacu pada kebijakan masing-masing perusahaan atau instansi sehingga dibutuhkan suatu pendalaman standar operasional terlebih dahulu untuk membuatnya.

Agar pekerjaan semakin mudah dan cepat, dibutuhkan otomasi kerja dari sebuah sistem informasi. otomasi ini juga dapat mengurangi kesalahan kerja yang terjadi akibat *human error* (Cahya Pambudi, 2018). Sistem informasi yang baik memiliki fungsi-fungsi otomatis di dalamnya yang disesuaikan dengan kebutuhan masing-masing pengguna.

Tempat penelitian yang berlokasi di kantor Komisi Pemilihan Umum (KPU) sangat membutuhkan sebuah sistem yang dapat membantu menyelesaikan pekerjaan menyangkut persuratan terutama surat keluar, kerena hal tersebut yang paling sering dilakukan. Bagian Umum KPU Kota Blitar mengalami kendala dalam proses pengarsipan surat keluar yang selama ini dalam melakukan penomoran surat, pemilihan lokasi penyimpanan dokumen, permintaan persetujuan atasan dan penyisipan tanda tangan digital masih dilakukan secara manual. Proses pengarsipan surat secara manual tersebut dinilai tidak efisien dan tidak tertib dalam penyimpanan dan pengelolaan dokumennya. Tidak hanya masalah pembuatan dan penyimpanan surat saja, permasalahan juga muncul Ketika Bagian Umum harus pelakukan pengurutan (*sorting*) dan pencarian (*searching*) surat. Dokumen yang sudah disimpan selama ini tidak pernah dilakukan proses *sorting* karena selain penamaan dokumen yang tidak standar, lokasi penyimpanan juga tidak menentu. Proses *searching* dokumen selama ini juga masih dilakukan secara manual dengan melihat satu persatu dokumen dan hal ini akan memakan banyak waktu.

2. TINJAUAN PUSTAKA

2.1. Kajian Penelitian

Metode *sorting* dibahas dalam penelitian yang dilakukan oleh (Sitepu dkk., 2017) yang bertujuan untuk membandingkan antara metode *Bubble Sort* dengan *Selection Sort*. Algoritma *Bubble Sort* dan *Selection Sort* menggunakan *Array List* multidimensi mampu mengurutkan data multi kriteria yang memiliki lebih dari satu kriteria prioritas. Jumlah iterasi *Selection Sort* pada setiap pengujian data lebih sedikit dibandingkan dengan *Bubble Sort*, dan waktu eksekusi *Selection Sort* pada setiap pengujian data juga lebih cepat jika dibandingkan dengan waktu eksekusi *Bubble Sort.*

Penelitian oleh (Retnoningsih, 2018) juga melakukan perbandingan metode sorting antara *Insertion Sort* dan *Bubble Sort* untuk mengurutkan data berupa angka. Kedua metode tersebut dibandingkan dengan cara mengukur waktu yang dibutuhkan untuk melakukan pengurutan pada data acak. Hasil yang didapat adalah metode *insertion sort* lebih unggul pada jumlah data yang sedikit yaitu pada sejumlah sepuluh elemen data membutuhkan waktu 0,003 detik, lima puluh elemen data membutuhkan waktu 0,013 detik, sedangkan metode *selection sort* lebih unggul pada jumlah data yang lebih banyak yaitu pada sejumlah lima ratus elemen data membutuhkan waktu 1,083 detik.

Penelitian ketiga tentang *sorting* dilakukan oleh (Saputro & Khasanah, 2018) yang membandingkan antara metode *Selection Sort* dan *Bubble Sort* untuk pengurutan data angka. Penelitian ini berhasil melakukan proses *sorting* terhadap tiga kelompok bilangan, kelompok pertama rentang 1 s/d 100, kelompok kedua rentang 100 s/d 1000, dan kelompok ketiga rentang 1000 s/d 1000000. Dari hasil analisa waktu yang telah dilakukan menunjukkan bahwa metode *Selection Sort* membutuhkan waktu yang lebih singkat dibandingkan dengan metode *Bubble Sort* dimana *Selection Sort* membutuhkan rata-rata waktu sekitar 0,031 detik, sedangkan metode *Bubble Sort* membutuhkan rata-rata waktu sekitar 0,078 detik.

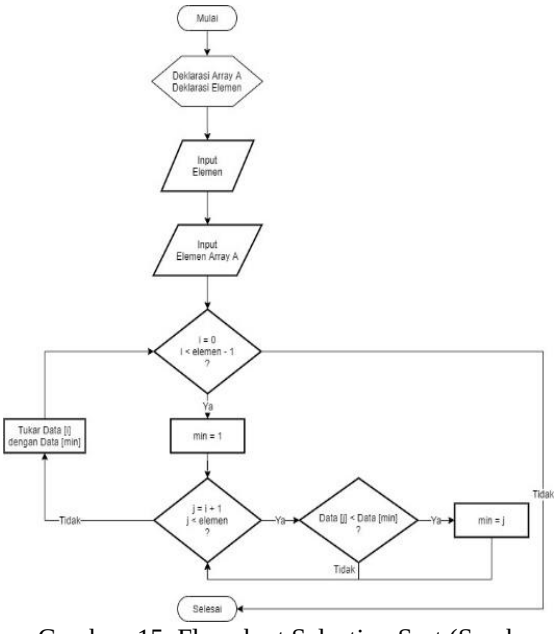
Penelitian yang dilakukan oleh (Sihotang, 2018), bertujuan untuk mengatasi masalah Pengadilan Tinggi Medan yang masih menggunakan sistem yang konvesional dalam surat menyurat sehingga sangat diharapkan agar sistem ini dapat berjalan dengan baik. Pengagendaan surat berdasarkan perancangan sistem yang telah disusun berdasarkan *use case diagram*, *activity* dan *class diagram* serta bahasa pemrograman *PHP*, *database MySQL*. Dalam tahap penulisan coding, digunakan *tools Dreamweaver* dan menjalankan program menggunakan *Xampp Control.*

Metode *searching* dalam penelitian berjudul "Implementasi Algoritma *Sequential Searching* untuk Pencarian Nomor Surat pada Sistem Arsip Elektronik" oleh (Sonita & Sari, 2018) menerapkan metode *Sequential Searching* untuk melakukan pencarian nomor surat pada aplikasi persuratan. Metode *Sequential Search* terbukti dapat di terapkan pada pencarian arsip berdasarkan nomor arsip dan berjalan dengan baik sesuai perencanaan. Permasalahan pada sistem arsip yang masih manual dapat teratasi dengan adanya sistem arsip elektronik ini sehingga menjadi lebih efisien dari segi penyimpanan, pengolahan dan proses.

Penelitian sistem administrasi persuratan Biro Organisasi Sekretariat Daerah Provinsi Jawa Jateng oleh (Wahyuningsih & Nada, 2019) bertujuan untuk mengatasi permasalahan arsip yang tidak terkelola dengan baik, surat terlambat hingga surat tidak diterima oleh penerima yang seharusnya dan surat yang rusak saat distribusi. Aplikasi administrasi persuratan dibuat pada basis *web* menggunakan metode *SDLC Waterfall* dengan *framework Codeigniter*. Perancangan sistem dengan menggunakan pemodelan *UML* lebih mudah karena alurnya jelas sehingga mempermudah ketika implementasi ke dalam sistem dan telah berhasil membantu staff dan karyawan di bidang persuratan dengan lebih efisien dan cepat sehingga manajemen surat lebih teratur dan dapat diakses kapanpun.

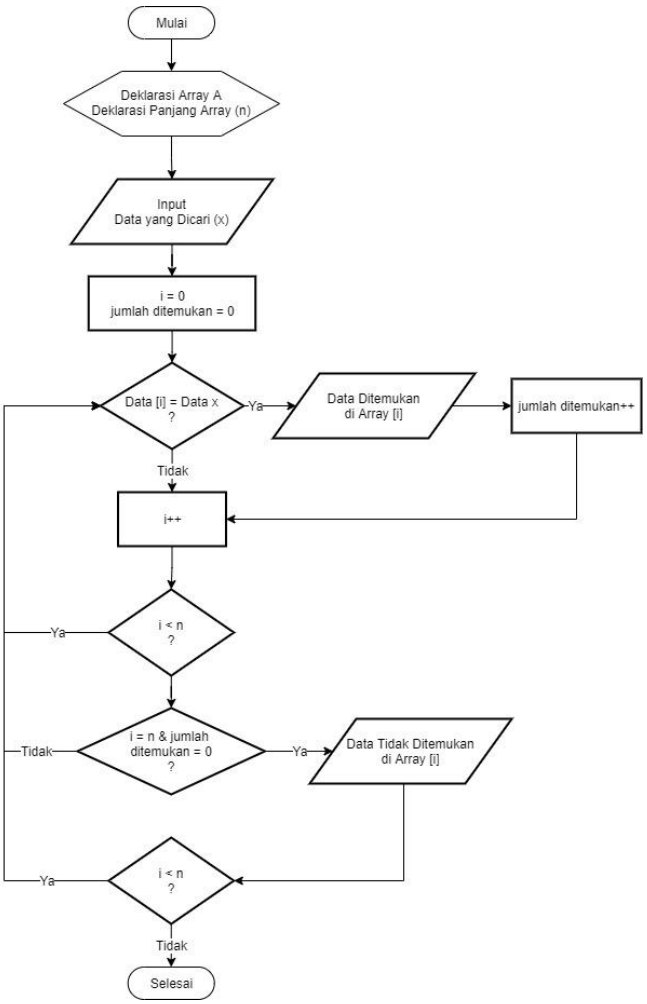
2.2. *Selection Sort*

Sorting (pengurutan) adalah proses mengurutkan data dengan aturan tertentu, sehingga tersusun secara teratur sesuai dengan aturan tersebut. Pada dasarnya ada dua aturan *sorting* yang biasa digunakan yaitu *ascending* (mengurutkan data dari data terkecil sampai data terbesar) dan *descending* (mengurutkan data dari yang terbesar sampai data yang terkecil). Proses yang terjadi dalam pengurutan data adalah proses perbandingan data dan pertukaran data.



Gambar 15. Flowchart Selection Sort (Sumber: Sitepu dkk., 2017)

Selection Sort merupakan metode pengurutan yang membandingkan elemen yang sekarang dengan elemen berikutnya sampai ke elemen yang terakhir. Pada pengurutan *ascending*, jika ditemukan elemen lain yang lebih kecil dari elemen sekarang maka dicatat posisisnya dan langsung ditukar. Konsep proses *Selection Sort* adalah mencari (memilih) nilai terkecil dan menukarnya dengan elemen paling awal (paling kiri) pada setiap tahap untuk kasus pengurutan *ascending* dan sebaliknya jika *descending*. (Sitepu dkk., 2017).



Gambar 2. Flowchart Sequential Search (Sumber: Sonita & Sari, 2018)

Sequential Search merupakan proses membandingkan setiap elemen *array* secara beruntun dimulai dari elemen pertama hingga elemen yang dicari ditemukan atau hingga elemen terakhir dari *array*. Metode *Sequential Search* atau biasa disebut pencarian beruntun dapat dipakai untuk melakukan pencarian data baik pada *array* yang sudah urut maupun yang belum urut. Langka-langkah *Sequential Search* adalah sebagai berikut:

1. membaca *array* data
2. menentukan data yang dicari
3. Mulai dari data pertama sampai dengan data terakhir, data yang dicari dibandingkan dengan masing-masing data di dalam *array*. Jika data yang dicari tidak ditemukan maka semua data atau elemen *array* dibandingkan sampai selesai. Jika data yang dicari ditemukan maka perbandingan akan dihentikan.

Proses pencarian data dengan metode ini cukup sederhana dan mudah. Pencarian data dilakukan dengan mencocokkan data yang dilakukan secara berurut satu demi satu dimulai dari data ke-1 hingga data pada urutan terakhir. Jika data yang dicari mempunyai nilai yang sama dengan data yang ada dalam kelompok data, berarti data telah ditemukan. Jika data yang dicari tidak ada yang cocok dengan data dalam sekelompok data, data tersebut tidak ada dalam sekelompok data (Sonita & Sari, 2018).

Setiap aplikasi yang digunakan pasti hanya bisa membaca atau menjalankan *file* dengan ekstensi tertentu yang mereka dukung saja. Sebagai contoh, dalam aplikasi *MS Word* bisa mengelola *file* dengan ekstensi "*.docx*", dalam aplikasi *adobe reader* bisa mengelola *file* "*.pdf*" dan lain sebagainya. *User* sering kali memerlukan perubahan ekstensi *file* untuk menyelesaikan suatu tugas tertentu.

Ketika sedang berurusan dengan web dan ingin membuat file dokumen berdasarkan tampilan *HTML*, maka dibutuhkan fungsi konversi file. *Phpdocx* menawarkan cara yang cukup canggih untuk memasukkan konten berformat *HTML* ke dalam dokumen *Word* (Wu *dkk*., 2018). Terdapat dua metode untuk memasukkan *HTML* ke dalam dokumen *Word* yaitu metode *add external file* dan *embed HTML*.

Metode *add external file* memungkinkan penyisipan dokumen *docx*, *HTML*, *RTF*, atau *MHT* eksternal ke dalam dokumen *word* yang akan ditunjuk dengan memperhatikan Teknik-teknik pemformatannya. Metode *embed HTML* memungkinkan penyisipan *HTML* ke dalam dokumen *Word* yang dituju dan mendukung hampir semua tag *HTML* dan *CSS*. Metode ini mengubah *HTML* menjadi *WordML* dan kompatibel dengan *docx reader* apapun seperti *LibreOffice*, *OpenOffice*, *Google Docs*, dan lainnya

3. METODE PENELITIAN

3.1. Jenis Penelitian

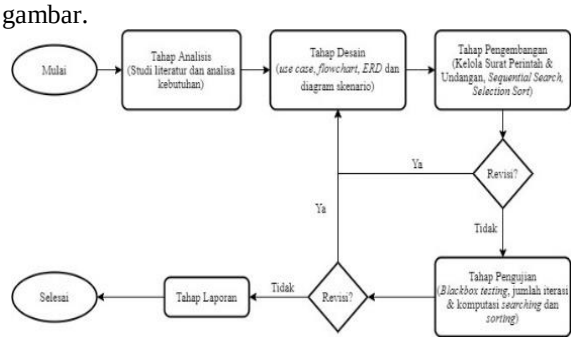
Jenis penelitian ini adalah *Research and Development* (*R&D*) yang dipakai untuk pengembangan lebih lanjut dari sebuah hasil penelitian atau produk penelitian. Penelitian ini bertujuan untuk mengubah metode pengarsipan lama di KPU Kota Blitar yang masih melakukan segala kegiatan secara manual menjadi otomatis. Setiap penelitian yang dihasilkan oleh tiap generasi pasti masih punya kekurangan yang dapat terus dierbaiki dan ditingkatkan kualitasnya agar lebih tepat guna dan berdaya guna, maka dari itu penelitian *R&D*

merupakan penelitian dengan jangka waktu yang panjang (Samsu, 2017).

Metode *Research and Development* digunakan untuk membuat suatu produk tertentu dan menguji efektif tidaknya produk tersebut (Arifin, 2020). Produk yang dimaksud tidak selalu berupa benda fisik seperti buku, perangkat keras atau alat peraga tapi juga bisa berupa perangkat lunak. *Research and Development* juga bisa berarti menyempurnakan produk yang sudah ada dengan penuh tanggung jawab sesuai aturan yang berlaku.

3.2. Prosedur Pengembangan

Perlu dilakukan penyesuaian pada jenis penelitian *Research and Development* (*R&D*) karena dalam penelitian ini yang dikembangkan adalah perangkat lunak berbasis web. Prosedur pengembangan yang dipilih untuk penelitian ini adalah pengembangan perangkat lunak *berdasarkan Software Development Life Cycle (SDLC)* merupakan siklus hidup pengembangan perangkat lunak yang digunakan untuk membangun suatu sistem yang terdiri dari tahap analisis, desain, pengembangan, pengujian dan tahap pembuatan laporan (Pressman & Maxim, 2019). Berdasarkan pertimbangan tersebut, prosedur pengembangan yang dipakai dalam penelitian ini dibuat dalam 5 tahap seperti pada gambar.



Gambar 3. Prosedur Pengembangan (Sumber : Pressman & Maxim, 2019)

3.3. Tahap Analisis

Pada tahap ini dilakukan beberapa kegiatan yaitu studi literatur dan analisa kebutuhan. Studi literatur berguna untuk dapat menemukan kasus yang dapat ditangani oleh sistem dan juga mengartikan sebuah sistem. Analisa kebutuhan sistem dan membuat batasan sistem menggunakan observasi juga harus dilakukan. Dengan begitu, pengembang mengetahui denan tepat kondisi di lapangan yang akan dijadikan sistem.

Tahap analisis kemudian dilanjutkan dengan kegiatan pengumpulan data dimana data yang digunakan dalam penelitian ini meliputi data pengguna aplikasi, data kebutuhan perangkat lunak persuratan KPU Kota Blitar, data surat perintah dan data surat undangan. Data pengguna aplikasi diperoleh melalui observasi dengan tujuan mengetahui siapa saja yang berhak menggunakan aplikasi surat keluar.

Data kebutuhan perangkat lunak diperoleh dengan melakukan wawancara tentang bagaimana alur persuratan selama ini lalu menterjemahkan hasilnya menjadi *use case diagram* yang berisi kebutuhan-kebutuhan fungsional perangkat lunak yang dikembangkan. Hasil lembar observasi ini akan dipakai sebagai dasar pada tahap desain menggunakan *UML*. Wawancara yang tepat akan menghasilkan aplikasi yang tepat pula

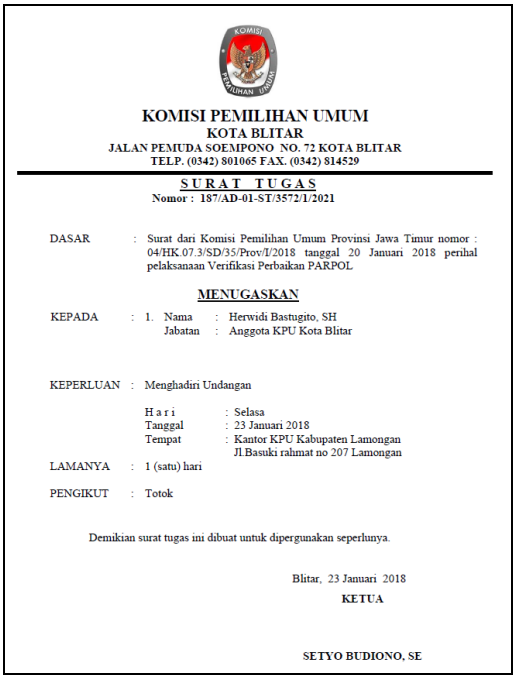
3.4. Tahap Desain

Tahap desain dikerjakan dengan menggunakan bahasa pemodelan perangkat lunak yang disebut *UML* (*Unified Modeling Language*). *UML* merupakan pengganti dari metode analisis berorientasi object dan design berorientasi object yang dibuat sekitar awal tahun 90-an dan telah melalui proses standarisasi dengan *OMG* (*Object Management Group*) sehingga *UML* menjadi bahasa standar pemodelan pengembangan perangkat lunak (Putra & Andriani, 2019). Terdapat 4 desain UML yang dibuat dalam penelitian ini, yaitu *use case*, *flowchart*, *ERD* dan diagram skenario.

Use Case Diagram adalah sebuah model diagram *UML* yang digunakan untuk pengembangan suatu sistem yang sesuai dengan kebutuhan demi menjelaskan berbagai proses yang berlangsung dalam suatu sistem serta mendokumentasikannya. Diagram *use case* menggambarkan secara singkat siapa (aktor) yang menggunakan sistem dan apa (*case*) yang bisa dilakukan. Use case penelitian ini telah disesuaikan dengan kondisi pelaksanaan kegiatan persuratan yang ada di KPU Kota Blitar.

4. HASIL DAN PEMBAHASAN

4.1. Format Surat Perintah



KOMISI PEMILIHAN UMUM
KOTA BLITAR
JALAN PEMUDA SOEMPONO NO. 72 KOTA BLITAR
TELP. (0342) 801065 FAX. (0342) 814529

SURAT TUGAS
Nomor : 187/AD-01-ST/3572/1/2021

DASAR : Surat dari Komisi Pemilihan Umum Provinsi Jawa Timur nomor : 04/HK.07.3/SD/35/Prov/I/2018 tanggal 20 Januari 2018 perihal pelaksanaan Verifikasi Perbaikan PARPOL

MENUGASKAN

KEPADA : 1. Nama : Herwidi Bastugito, SH
Jabatan : Anggota KPU Kota Blitar

KEPERLUAN : Menghadiri Undangan

Hari : Selasa
Tanggal : 23 Januari 2018
Tempat : Kantor KPU Kabupaten Lamongan
Jl.Basuki rahmat no 207 Lamongan

LAMANYA : 1 (satu) hari

PENGIKUT : Totok

Demikian surat tugas ini dibuat untuk dipergunakan seperlunya.

Blitar, 23 Januari 2018
KETUA

SETYO BUDIONO, SE

Gambar 4. Format Surat Tugas KPU

Isi dan format penulisan surat perintah dalam aplikasi mengacu pada format surat asli yang digunakan di KPU Kota Blitar seperti pada gambar. Komponen yang harus ada di dalam surat perintah seperti nomor surat, dasar, nama pegawai yang diberi tugas, jabatan, keperluan, hari, tanggal, tempat, lama tugas dan nama pengikut sudah diterapkan ke dalam aplikasi. Surat perintah ini dibuat oleh Bagian Umum dan disetujui oleh Ketua KPU, namun jika Ketua KPU berhalangan, maka bisa diwakilkan oleh Sekretaris KPU.

Penomoran surat yang digunakan dalam surat perintah harus mengikuti format yang ada di KPU Kota Blitar. Terdapat 5 komponen penyusun nomor surat dimulai dari sebelah kiri yaitu nomor surat, kode bagian diikuti dengan "ST", kode lokasi, bulan dan terakhir adalah tahun. Nomor surat akan dibuat secara otomatis oleh aplikasi.

4.2. Format Surat Undangan

Isi dan format penulisan surat undangan dalam aplikasi mengacu pada format surat asli yang digunakan di KPU Kota Blitar seperti pada gambar. Komponen yang harus ada di dalam surat undangan seperti nomor surat, sifat, jumlah lampiran, perihal, tujuan, kalimat pembuka, hari, tanggal, tempat, waktu dan catatan surat sudah diterapkan ke dalam aplikasi. Surat undangan ini dibuat oleh Bagian Umum dan disetujui oleh Ketua KPU, namun jika Ketua KPU berhalangan, maka bisa diwakilkan oleh Sekretaris KPU.

Penomoran surat yang digunakan dalam surat undangan harus mengikuti format yang ada di KPU Kota Blitar. Terdapat 5 komponen penyusun nomor surat dimulai dari sebelah kiri yaitu nomor surat, kode bagian diikuti dengan "SU", kode lokasi, bulan dan terakhir adalah tahun. Nomor surat akan dibuat secara otomatis oleh aplikasi.

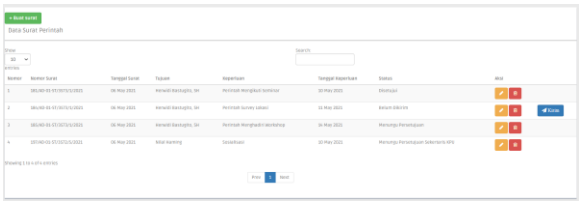
4.3. Penyimpanan Surat

Aplikasi surat keluar ini mendukung penyimpanan otomatis untuk memudahkan pengguna dan standarisasi administrasi. Sebelum disetujui oleh Ketua atau Sekretaris KPU, surat yang sudah dibuat melalui aplikasi akan disimpan secara otomatis dengan format file "*.doc*". Setelah disetujui oleh Ketua atau Sekretaris KPU, surat yang telah dibuat akan disimpan secara otomatis dengan format "*.docx*".

Dokumen surat perintah akan disimpan pada direktori "C:/surat/surat_perintah/nomor surat", nomor surat akan menyesuaikan dengan masing-masing nomor surat perintah. Dokumen surat undangan akan disimpan pada direktori "C:/surat/surat_undangan/nomor surat", nomor surat akan menyesuaikan dengan masing-masing nomor surat undangan. Pengguna aplikasi tidak perlu membuat direktori-direktori tersebut saat memasang aplikasi, secara otomatis aplikasi akan membuat direktori tersebut di *drive* "C".

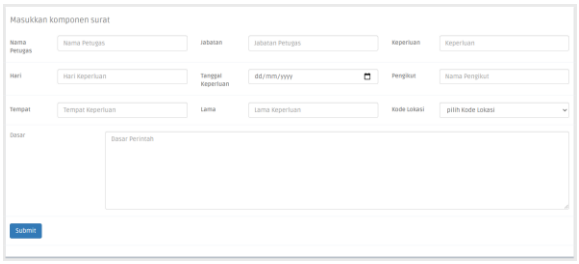
4.4. Kelola Surat Perintah

Menu kelola surat perintah memiliki 4 fungsi yaitu lihat, buat, ubah dan hapus surat perintah. Fungsi lihat surat dapat diakses oleh Bagian Umum, Sekretaris dan Ketua KPU. Surat perintah yang muncul pada Bagian Umum adalah semua surat perintah, surat yang muncul pada akun Sekretaris KPU hanya surat yang harus ditandatangani oleh Sekretaris, begitu juga yang muncul pada akun Ketua KPU adalah surat yang harus ditandatangani oleh Ketua saja.



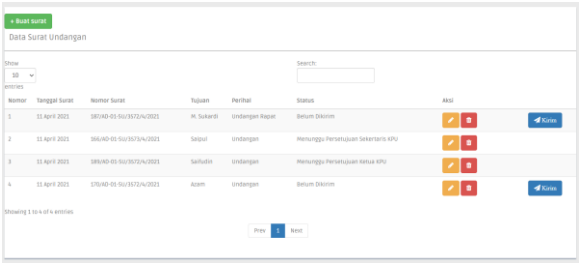
Gambar 5. Kelola Surat Perintah

Fungsi "buat surat perintah" bisa dipakai oleh Bagian Umum dengan menekan tombol "Buat Surat" pada menu "surat perintah". Komponen yang harus dimasukkan dalam membuat surat perintah adalah, nama petugas, jabatan, keperluan, hari, tanggal, pengikut, tempat, lama kegiatan, kode lokasi dan dasar penulisan surat. Setelah selesai dibuat, Bagian Umum bisa memilih siapa yang akan memberi persetujuan surat tersebut.



Gambar 6. Buat Surat Perintah

4.5. Kelola Surat Undangan

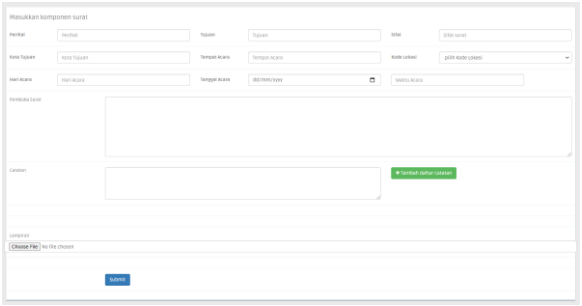


Gambar 7. Kelola Surat Undangan

Menu kelola surat undangan memiliki 4 fungsi yaitu lihat, buat, ubah dan hapus surat undangan. Fungsi lihat surat dapat diakses oleh Bagian Umum, Sekretaris dan Ketua KPU. Surat undangan yang muncul pada Bagian Umum adalah semua surat undangan yang sudah dibuat, surat yang muncul pada

akun Sekretaris KPU hanya surat yang harus ditandatangani oleh Sekretaris, begitu juga yang muncul pada akun Ketua KPU adalah surat yang harus ditandatangani oleh Ketua saja.

Fungsi "buat surat undangan" bisa dipakai oleh Bagian Umum dengan menekan tombol "Buat Surat" pada menu "surat undangan". Komponen yang harus dimasukkan dalam membuat surat undangan adalah, perihal, tujuan, sifat surat, kota tujuan, tempat acara, kode lokasi, hari, tanggal, waktu, pembuka surat, catatan surat, dan lampiran surat jika ada. Setelah selesai dibuat, Bagian Umum bisa memilih siapa yang akan memberi persetujuan surat tersebut.



Gambar 8. Buat Surat Undangan

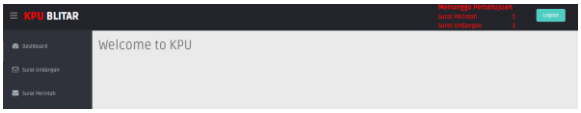
4.6. Implementasi Pengarsipan *DOCX* Otomatis

Surat yang selesai dibuat oleh Bagian Umum akan dimintakan persetujuan melalui aplikasi ke Ketua KPU atau Sekretaris KPU (Jika Ketua berhalangan). Surat yang belum disetujui tidak bisa diubah formatnya ke *DOCX* oleh bagian umum, namun Ketua atau Sekretaris bisa mengunduhnya dalam bentuk *DOCX* untuk dibaca lalu diputuskan untuk ditolak atau disetujui. Setelah disetujui, *icon* status surat tersebut pada akun Bagian Umum akan berubah sehingga bisa diunduh dalam bentuk *DOCX*.

Gambar 9. Perubahan Status Surat Pada Bagain Umum

4.7. Pemberitahuan Otomatis

Surat yang sudah di kirimkan kepada Ketua atau Sekretaris KPU akan muncul pemberitahuan otomatis pada aplikasi Ketua atau Sekretaris KPU. Ketika surat yang dikirimkan sudah di tindak lanjuti, maka pemberitahuan akan berkurang otomatis sesuai dengan berapa surat yang sudah di tindak lanjuti.



Gambar 10. Contoh Pemberitahuan Otomatis

4.8. Pengujan *Black Box*

Berikut ini adalah hasil pengujian *Black Box Testing* pada aplikasi dengan melibatkan 10 orang penguji yang terdiri dari pegawai KPU Kota Blitar sebanyak 5 orang dan masyarakat di luar KPU sebanyak 5 orang:

Tabel 1. Rekap Kesesuaian dan Tingkat Kenormalan

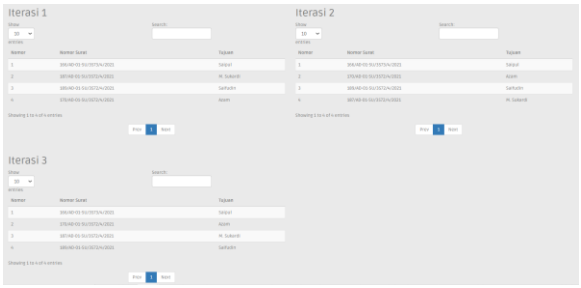
No.	Menu	Kesesuaian	Tingkat Kenormalan
1	Kelola surat perintah	95%	95%
2	Kelola surat undangan	95%	95%
3	Minta persetujuan	100%	100%
4	Memberi persetujuan	100%	100%
5	Kelola *user*	95%	95%
Rata-Rata		**95%**	**90%**

4.9. Implementasi dan Pengujian *Selection Sort*

Surat yang telah dibuat bisa diurutkan menggunakan algoritma selection *sort* pada menu "*Selection Sort*". Pada awal membuka menu, pengguna akan ditampilkan seluruh surat yang ada dalam posisi acak.

Gambar 11. Halaman Selection Sort

Pengguna hanya perlu menekan tombol "*sort*" lalu dokumen surat akan diurutkan dan ditampilkan iterasi yang dihasilkan oleh algoritma *selection sort* tersebut.



Gambar 12. Hasil pengujian iterasi selection sort dalam aplikasi

Pada menu ini diimplementasikan algoritma *selection sort* dengan perubahan iterasi yang juga diperlihatkan untuk pengguna. Posisi awal urutan surat undangan adalah surat dengan nomor 187; 166; 189; 170 berdasarkan algoritma *selection sort* maka dilakukan pengecekan seperti tabel berikut.

Tabel 2. Tabel Kebenaran Pengujian Selection Sort

Perulangan / Iterasi	Nomor Surat	Pengecekan	Jawaban
1	187 ; 166 ; 189 ; 170	187 < 166 ?	Tidak, tukar isi index 1 dengan 2
	166 ; 187 ; 189 ; 170	166 < 189 ?	Ya
	166 ; 187 ; 189 ; 170	166 < 170 ?	Ya
Iterasi 1 selesai karena sudah melakukan pengecekan index ke-1 sampai ke-4. Urutan terbaru setelah iterasi 1 adalah **166 ; 187 ; 189 ; 170**			
2	166 ; 187 ; 189 ; 170	187 < 189	Ya
	166 ; 187 ; 189 ; 170	187 < 170 ?	Tidak, Tukar isi index 2 dengan 4
Iterasi 2 selesai karena sudah melakukan pengecekan index ke-2 sampai ke-4. Urutan terbaru setelah iterasi 2 adalah **166 ; 170 ; 189 ; 187**			
3	166 ; 170 ; 189 ; 187	189 < 187	Tidak, tukar isi index 3 dengan 4
Iterasi 3 selesai karena sudah melakukan pengecekan index ke-3 sampai ke-4. Urutan terbaru setelah iterasi 3 adalah **166 ; 170 ; 187 ; 189**			

Maka dari algoritma *selection sort* diperoleh hasil akhir iterasi menjadi 166; 170; 187; 189 dengan jumlah iterasi sebanyak 3.

Kode sumber untuk tampilan halaman *selection sort* bisa dilihat di direktori "*C:\xampp\htdocs\kpu\resources\views\sort*". Logika yang digunakan untuk halaman *sorting* bisa dilihat pada *controller* "*sort_Controller.php*" yang ada pada direktori "*C:\xampp\htdocs\kpu\app\Http\Controllers*".

```
function selection_sort($data){
    $dataq = $data;
    for($i=0; $i<count($data)-1; $i++) {
        $min = $i;
        for($j=$i+1; $j<count($data); $j++) {
            if ($data[$j]<$data[$min]) {
                $min = $j;
            }
        }
        $sort1 = DB::select("select max(iterasi) as it from sort");
        foreach($sort1 as $ss){
            $iteras = $ss->it + 1;
        }
        $data = swap_positions($data, $i, $min);
            if($data[$i]<$data[$min]){
                $kecil = min($data);
                $sortt = new sort();
                $sortt->nomor = $kecil;
                $sortt->iterasi = $iteras;
                $sortt->save();
                for($k=1;$k<count($data);$k++){
                    $sortq = new sort();
                    $sortq->nomor = $data[$k];
                    $sortq->iterasi = $iteras;
                    $sortq->save();
                }
            }
    }
    return $data;
    echo implode(', ',$dataq)."<br>".implode(', ',$data);die();
}
```

Gambar 13. Kode Sumber selection sort

Dari *controller* tersebut terlihat bahwa kode sumber yang ditulis sudah menerapkan desain algoritma *selection sort* pada method "*selection_sort*" dengan menerapkan pengulangan dan pengecekan dimana jika data di depan lebih kecil dari data yang ada di belakang, maka data tersebut akan ditukar posisinya.

Jika dibandingkan dengan metode sorting lain seperti *bubble sort*, *selection sort* memiliki kelebihan yaitu jumlah proses yang lebih sedikit sehingga memperingan komputasi.

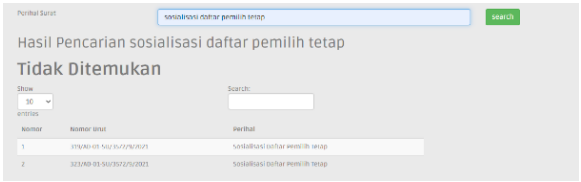
Tabel 3. Tabel Kebenaran Pengujian Buble Sort

Perulangan / Iterasi	Nomor Surat	Pengecekan	Jawaban
1	187 ; 166 ; 189 ; 170	187 < 166 ?	Tidak, tukar isi index 1 dengan 2
	166 ; 187 ; 189 ; 170	187 < 189 ?	Ya
	166 ; 187 ; 189 ; 170	166 < 170 ?	Ya
Iterasi 1 selesai karena sudah melakukan pengecekan index ke-1 sampai ke-4. Urutan terbaru setelah iterasi 1 adalah **166 ; 187 ; 189 ; 170**			
2	166 ; 187 ; 189 ; 170	166 < 187 ?	Ya
	166 ; 187 ; 189 ; 170	187 < 189 ?	Ya
	166 ; 187 ; 189 ; 170	189 < 170 ?	Tidak, tukar index 3 dengan 4
Iterasi 2 selesai karena sudah melakukan pengecekan index ke-1 sampai ke-4. Urutan terbaru setelah iterasi 2 adalah **166 ; 187 ; 170 ; 189**			
3	166 ; 187 ; 170 ; 189	189 < 187 ?	Ya
	166 ; 187 ; 170 ; 189	187 < 170 ?	Tidak, tukar index 2 dengan 3
	166 ; 170 ; 187 ; 189	187 < 189 ?	Ya
Iterasi 3 selesai karena sudah melakukan pengecekan index ke-3 sampai ke-4. Urutan terbaru setelah iterasi 3 adalah **166 ; 170 ; 187 ; 189**			

Dari tabel diatas dapat kita lihat bahwa *selection sort* memiliki jumlah proses yang menurun pada tiap iterasinya, sehingga meskipun keduanya punya jumlah iterasi yang sama namun jumlah prosesnya berbeda. *Selection sort* menghasilkan 6 proses sedangkan *bubble sort* menghasilkan 9 proses dengan data yang sama, ini berarti *selection sort* dapat mempercepat komputasi.

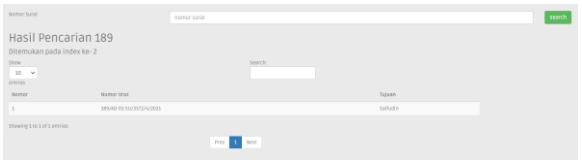
4.10. Implementasi dan Pengujian *Sequential Search*

Sequential search yang telah didesain, diterapkan pada aplikasi pada menu "*Sequential Search*". Pada menu ini pengguna bisa memasukkan indek pencarian (nomor surat, perihal surat, tujuan surat) dan menekan tombol "*Search*".



Gambar 14. Hasil pencarian Sequential Search menggunakan perihal surat

Gambar 15. Hasil pencarian Sequential Search menggunakan tujuan surat



Gambar 16. Hasil pencarian Sequential Search menggunakan nomor surat

Pada gambar 13 adalah contoh pencarian surat dimana yang di masukan adalah *index* perihal surat, pada gambar 14 adalah contoh pencarian menggunakan *index* tujuan surat, sedangkan pada gambar 16 adalah contoh pencarian surat menggunakan *index* nomor surat, setelah itu aplikasi akan mencari surat tersebut menggunakan algoritma *sequential search* yang telah ditanamkan. Jika surat yang dicari ditemukan, aplikasi akan menampilkan pada *index* ke berapa surat tersebut ditemukan karena algoritma *sequential search* akan menampung semua nomor surat pada *array* dan melakukan pencarian secara *sequential.*

Berdasarkan gambar 20 bisa kita lihat bahwa data awal nomor surat (masih sama seperti saat *sorting*) adalah 187; 166; 189; 170. Aplikasi berhasil menemukan nomor surat yang dicari yaitu 189 menggunakan algoritma *sequential search* dengan perulangan yang terjadi bisa dilihat pada tabel berikut.

Tabel 4. Tabel Kebenaran Pengujian Sequential Search

Perulangan ke-	**Aray index ke-**	**Data**	**Pengecekan**	**Jawaban**
1	1	187	Data = 189 ?	Tidak
2	2	166	Data = 189 ?	Tidak
3	3	189	Data = 189 ?	Ya, simpan index ke-3
4	4	170	Data = 189 ?	Tidak

Berdasarkan perulangan yang dilakukan aplikasi menggunakan algoritma *sequential search* maka diperoleh hasil pencarian nomor surat 189 sebanyak 1x yaitu pada index ke-3 dan perulangan ke-3.

Dibandingkan dengan *binary search*, *sequential search* memiliki jumlah iterasi dan proses yang lebih sedikit sehingga dapat mempercepat komputasi, seperti pada tabel dibawah ini.

Tabel 5. Tabel Kebenaran Pengujian Binary Search

Iterasi 1 = Melakukan sorting data. Jika menggunakan selction sort maka total proses = 6	6 proses
Mencari titik tengah array dengan rumus: = round (Jumlah data / 2) = 4 / 2 → Titik tengah = 2	2 proses

Iterasi 3 = Melakukan searching pada data yang sudah diurutkan.
Data yang dicari = 189 ; titik tengah = index ke-2 = 170
Karena data yang dicari > titik tengah, maka arah pencarian ke kanan

Perulangan ke-	Data	Pengecekan	Jawaban
1	166 ; 170 ; 187 ; 189	Data = 189 ?	Tidak
2	166 ; 170 ; 187 ; 189	Data = 189 ?	Tidak
3	166 ; 170 ; 187 ; 189	Data = 189 ?	Ya, simpan index ke-3

Pencarian sudah selesai, data ketemu di perulangan ke-3 iterasi ke-3

Total proses yang dipakai dalam binary search adalah 11 proses sedangkan *sequential search* hanya 4. Hal ini menunjukkan bahwa *sequential search* dapat mempercepat komputasi dibandingkan *binary search* untuk pencarian surat di KPU.

Kode sumber untuk tampilan halaman *sequential search* bisa dilihat di direktori "*C:\xampp\htdocs\kpu\resources\views\search*". Logika yang digunakan untuk halaman *searching* bisa dilihat pada *controller* "*search_Controller.php*" yang ada pada direktori "*C:\xampp\htdocs\kpu\app\Http\Controllers*".

```
public function index(Request $request){
    if($request->cari !=""){
        $cari = $request->cari;
        $undangan = DB::select("select * from surat_undangan");
        $m=0;
        foreach($undangan as $u){
            $nomor[$m++] = $u->nomor_urut;
        }
        $jml = count($nomor);
        for($i=0;$i<$jml;$i++){
            if($cari == $nomor[$i]){
                $surats = DB::select("select * from surat_undangan where nomor_surat like '%$cari%'");
                $index = $i;
                $ketemu = 99;
                break;
            }else{
                $index =0;
                $surats = DB::select("select * from surat_undangan where nomor_surat = '$cari' ");
                $ketemu =0;
            }
        }
    }else{
        $surats = undangan::all();
        $cari = $request->cari;
        $ketemu = 2;
        $index = 0;
    }
    return view('search/index',compact('surats','cari','index','ketemu'));
}
```

Gambar 17. Kode sumber Sequential Search

Dari *controller* tersebut terlihat bahwa kode sumber yang ditulis sudah menerapkan desain algoritma *sequential search* pada method "*index*" dengan melakukan pengecekan pada seluruh surat yang ada dan menyimpan *index* dimana surat tersebut ditemukan, jika tidak ada *index* yang ditemukan berarti surat yang dicari tidak terdapat pada basis data.

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berikut ini adalah beberapa kesimpulan yang dapat diambil dari pengerjaan penelitian mulai awal hingga selesai: Aplikasi pengelolaan surat keluar menggunakan *sequential search* dan *selection sort* pada KPU Kota Blitar bisa dibuat dengan menerapkan alur *Software Development Life Cycle (SDLC)* mulai tahap analisis, desain, pengembangan, pengujian dan tahap pembuatan laporan. Kode sumber aplikasi dengan memadukan bahasa pemrograman *web* dan *Framework Laravel*. Secara umum Aplikasi yang dibuat sudah memenuhi kebutuhan fungsional yang telah di desain meliputi kelola surat perintah, kelola surat undangan, minta persetujuan, memberi persetujuan dan kelola *user*, Konversi file dari tampilan web menjadi *DOCX* dapat dilakukan dengan pemformatan *file* menggunakan *library* yang ada dalam *Laravel*. *Layout* isi *file* mulai dari kop surat hingga tanda tangan disusun sedemikian rupa sehingga mirip dengan format aslinya, Metode *selection sort* dan *sequential search* diaplikasikan ke dalam kode program dengan menerapkan cara kerjanya masing-masing yang sudah di desain sebelumnya menggunakan *flowchart* dimana kode program ditulis menggunakan bahasa pemrograman *php* dengan perulangan dan percabangan di dalamnya. Metode *selection sort* telah diuji melalui pengoperasian langsung dari aplikasi dan bisa menghasilkan *sorting* yang baik. *Selection sort* menghasilkan total 6 proses dan ini lebih menghemat komputasi dari metode *bubble sort* yang menghasilkan 9 proses. Begitu juga dengan pengujian metode *sequential search* yang menghasilkan hasil *searching* yang sesuai dengan desain. *Sequential search* menghasilkan 4 proses sedangkan metode lain seperti binary search menghasilkan 11 proses yang menunjukkan bahwa *sequential search* dapat mempercepat komputasi. Hasil pengujian melalui pengoperasian aplikasi juga dibuktikan dengan tabel kebenaran yang ditulis secara manual berdasarkan algoritma *selection sort* dan *sequential search* dan mendapat hasil yang sesuai antara desain, algoritma dan *output* pada aplikasi

5.2. Saran

Berikut ini adalah saran-saran yang dapat digunakan untuk memperbaiki penelitian ini di masa depan: Aplikasi surat keluar bisa dikembangkan lagi untuk jenis surat lain, Backup *database* secara berkala bisa dibuat menjadi otomatis untuk mencegah kehilangan data, Surat yang selesai dibuat bisa mengirimkan notifikasi pada *smartphone* Ketua atau Sekretaris KPU, Server yang digunakan bisa diubah menjadi *cloud server* sehingga dapat diakses dari mana saja melalui internet.

DAFTAR PUSTAKA

[1] Lubis, M. R., Susanti, E., Wirapraja, A., Siregar, M. N. H., Simarmata, J., Fadhillah, Y., Giap, Y. C., Abdillah, L. A., Purba, R. A., & Muttaqin, M. (2020). *Pengenalan Teknologi Informasi*. Yayasan Kita Menulis.

[2] Anggraeni, E. Y. (2017). *Pengantar Sistem Informasi*. Penerbit Andi.

[3] Cahya Pambudi, C. (2018). *Rancang Bangun Sistem Informasi Pengelolaan Data Transaksi Hotel (Studi Kasus Hotel Perdana Yogyakarta)*. *Universitas Teknologi Yogyakarta.*

[4] Sitepu, R. R., Yusman, M., & Febriansyah, F. E. (2017). Implementasi Algoritma Bubble Sort Dan Selection Sort Menggunakan Arraylist Multidimensi Pada Pengurutan Data Multi Prioritas. *Jurnal Komputasi*, *5*(1).

[5] Sihotang, H. T. (2018). Sistem Informasi Pengagendaan Surat Berbasis Web Pada Pengadilan Tinggi Medan. *Journal Of Informatic Pelita Nusantara*, *3*(1).

[6] Wahyuningsih, Y., & Nada, N. Q. (2019). Perancangan Sistem Administrasi Persuratan Biro Organisasi Sekretariat Daerah Provinsi Jawa Jateng. *Seminar Nasional Science and Engineering National Seminar*, *1*(1).

[7] Sonita, A., & Sari, M. (2018). Implementasi algoritma sequential searching untuk pencarian nomor surat pada sistem arsip elektronik. *Pseudocode*, *5*(1), 1–9.

[8] Retnoningsih, E. (2018). Algoritma Pengurutan Data (Sorting) Dengan Metode Insertion Sort dan Selection Sort. *Information Management For Educators And Professionals: Journal of Information Management*, *3*(1), 95–106.

[9] Saputro, F. E., & Khasanah, F. N. (2018). Teknik Selection Sort dan Bubble Sort Menggunakan Borland C++. *Jurnal Mahasiswa Bina Insani*, *2*(2), 136–145.

[10] Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., & Weston, J. (2018). Starspace: Embed all the things! *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1).

[11] Pressman, R., & Maxim, B. (2019). *Software Engineering: A Practitioner's Approach 9th Edition.*