**IMPLEMENTASI ALGORITMA *BOYER-MOORE* DALAM SISTEM INFORMASI *LOST AND FOUND* UNIVERSITAS BAKRIE**

**TUGAS AKHIR**

****

**RISTANTI SEPTA AYU ANGGRAINI**

**1122001015**

**FAKULTAS TEKNIK DAN ILMU KOMPUTER**

**PROGRAM STUDI INFORMATIKA**

**UNIVERSITAS BAKRIE**

**JAKARTA**

**2016**

# **PERNYATAAN PEMBIMBING**

Yang bertanda tangan di bawah ini saya:

Pembimbing

Nama : Yusuf Lestanto, S.T., M.Sc.

Dengan ini menyatakan bahwa penelitian tugas akgir skripsi dengan judul:

“Implementasi Algoritma Boyer-Moore dalam Sistem Informasi *Lost and Found*  di Universitas Bakrie”

Yang disusun oleh:

Nama : Ristanti Septa Ayu Anggraini

NIM : 1122001015

Telah selesai dan siap untuk diajukan dan dipertahankan di hadapan sidang tugas akhir.

Demikian pernyataan ini dibuat untuk dapat dipergunakan sebagai syarat kelengkapan mengikuti sidang tugas akhir di Universitas Bakrie.

|  |
| --- |
| Jakarta, Agustus 2016  Yang menyatakan  Dosen Pembimbing  Yusuf Lestanto, S.T., M.Sc. |

**IMPLEMENTASI ALGORITMA *BOYER-MOORE* DALAM SISTEM INFORMASI *LOST AND FOUND* UNIVERSITAS BAKRIE**

Ristanti Septa Ayu Anggraini

# **ABSTRAK**

Proses penemuan dan laporan kehilangan di lingkungan kampus sering menjadi persoalan di bagian *security.*  Hal ini dikarenakan proses pembukuan dan rekapitulasi barang yang ditemukan dan laporan kehilangan masih dilakukan secara manual sehingga cara tersebut kurang efektif. Tidak hanya itu proses laporan pencarian barang yang sudah lama sulit ditemukan apabila masih menggunakan dokumen manual. Berdasarkan masalah tersebut diperlukan sebuah aplikasi pencarian dan sebagai penyimpanan data laporan barang kehilangan atau temuan. Aplikasi ini merupakan pengelolaan informasi tepat dan akurat yang dirancang untuk memberikan kemudahan dalam mengakses dan menyimpan data laporan temuan atau kehilangan barang. Aplikasi *Lost and Found* ini menerapkan algoritma *Boyer-Moore* pada proses *string searching* dengan fitur *auto-complete*, dimana pada proses pencocokan *string*dilakukan dari kanan ke kiri*.* Hasil dari penelitian ini berupa sistem informasi berbasis *web* dengan menggunakan fitur *auto-complete.*

Kata Kunci:

*Boyer-Moore*, *Auto-complete,* Laporan Temuan Barang, Laporan Kehilangan Barang

***BOYER-MOORE* *STRING* SEARCH ALGORITHM IMPLEMENTATION OF INFORMATION SYSTEM “*LOST AND FOUND* ” IN BAKRIE UNIVERSITY**

Ristanti Septa Ayu Anggraini

# **ABSTRACT**

*The report process of Lost and Found in university is often a problem in security. This is because documenting process and recapitulation of items Lost and Found is still done manually so the less effective way. Searching process for the Lost and Found item that have been stored too hard to find because searching data still use manual document. Based on these problems, required a search application and as a report of Lost and Found item in Universitas Bakrie. This application is precise and accurate management in*form*ation designed to provide ease of accessing and storing data of Lost and Found reports. Application “Lost and Found ” applies Boyer-Moore algorithm in string searching process with auto-complete feature, which is the string matching process do from right to left. The results of this research is a web-based in*form*ation system by using the auto-complete feature.*

*Keywords:*

*Boyer-Moore, Auto-Complete, Finding Report Item, Lost Report Item*

# **DAFTAR ISI**

[**PERNYATAAN PEMBIMBING** ii](#_Toc458166106)

[**ABSTRAK** iii](#_Toc458166107)

[**ABSTRACT** iv](#_Toc458166108)

[**DAFTAR ISI** v](#_Toc458166109)

[**DAFTAR GAMBAR** vii](#_Toc458166110)

[**DAFTAR TABEL** ix](#_Toc458166111)

[**DAFTAR RUMUS** x](#_Toc458166112)

[**DAFTAR SINGKATAN** xi](#_Toc458166113)

[**BAB I** 1](#_Toc458166114)

[1.1 Latar Belakang Masalah 1](#_Toc458166115)

[1.2 Identifikasi Masalah 2](#_Toc458166116)

[1.3 Rumusan Masalah 3](#_Toc458166117)

[1.4 Batasan Masalah 3](#_Toc458166118)

[1.5 Tujuan Penelitian 3](#_Toc458166119)

[1.6 Manfaat Penelitian 3](#_Toc458166120)

[1.7 Sistematika Penulisan 4](#_Toc458166121)

[**BAB II** 5](#_Toc458166122)

[2.1 Penelitian Terdahulu 5](#_Toc458166123)

[2.2 Konsep Dasar Sistem Informasi 11](#_Toc458166124)

[2.2.1 Definisi Sistem Informasi 11](#_Toc458166125)

[2.2.2 Komponen Sistem Informasi 11](#_Toc458166126)

[2.3 Konsep Dasar *String Searching* 12](#_Toc458166127)

[2.4 Algoritma *Boyer-Moore* 14](#_Toc458166128)

[2.4.1 *Good-suffix shift rule* 16](#_Toc458166129)

[2.4.2 *Bad-character rule* 16](#_Toc458166130)

[2.5 Model Siklus Pengembangan Perangkat Lunak 17](#_Toc458166131)

[2.6 Bahasa Pemrograman 21](#_Toc458166132)

[2.7 *Unified Modeling Language* (UML) 24](#_Toc458166133)

[2.8 Pengujian 25](#_Toc458166134)

[**BAB III** 27](#_Toc458166135)

[3.1 Metode Perancangan dan Pengembangan 27](#_Toc458166136)

[3.1.1 Pengamatan dan Perencanaan 27](#_Toc458166137)

[3.1.2 Analisa Kebutuhan Aplikasi 27](#_Toc458166138)

[3.1.3 Perancangan dan Pembangunan 27](#_Toc458166139)

[3.1.4 *Testing* 44](#_Toc458166140)

[3.1.5 Implementasi 46](#_Toc458166141)

[3.2 Kerangka Penelitian 46](#_Toc458166142)

[3.3 Jenis Penelitian 48](#_Toc458166143)

[3.4 Objek Penelitian 48](#_Toc458166144)

[3.5 Metode Pengumpulan Data 48](#_Toc458166145)

[3.6 Implementasi Algoritma *Boyer-Moore* 49](#_Toc458166146)

[3.7 Rencana Kegiatan Penelitian 52](#_Toc458166147)

[**BAB IV** 53](#_Toc458166148)

[4.1 *Design* *User Interface* 53](#_Toc458166149)

[4.2 Implementasi Sistem 57](#_Toc458166150)

[4.3 Implementasi Perancangan Antarmuka 57](#_Toc458166151)

[4.4 Implementasi Data 64](#_Toc458166152)

[4.5 Implementasi Algoritma *Boyer-Moore* pada Fitur *Auto-Complete* 64](#_Toc458166153)

[4.6 Hasil Pencarin Berdasarkan Kata Kunci 67](#_Toc458166154)

[4.7 Pengujian Algoritma 69](#_Toc458166155)

[4.7.1 Menentukan *Pattern* pada Text 69](#_Toc458166156)

[4.7.2 Proses Pencarian Algoritma 70](#_Toc458166157)

[4.7.3 Menentukan Bobot Kriteria 74](#_Toc458166158)

[4.7.4 Pemberian Nilai Pada Setiap Kriteria 75](#_Toc458166159)

[4.7.5 Menghitung Skor 75](#_Toc458166160)

[4.7.6 Menentukan Prioritas Keputusan 77](#_Toc458166161)

[**BAB V** 78](#_Toc458166162)

[5.1 Simpulan 78](#_Toc458166163)

[5.2 Saran 78](#_Toc458166164)

[**DAFTAR PUSTAKA** 80](#_Toc458166165)

[**LAMPIRAN** 83](#_Toc458166166)

# **DAFTAR GAMBAR**

[Gambar 2. 1 Komponen sebuah Sistem Informasi [7] 12](#_Toc458082234)

[Gambar 2. 2 Good-suffix shift, u terjadi lagi didahului karakter c berbeda dari a [12] 16](#_Toc458082235)

[Gambar 2. 3 *Good-suffix shift*, hanya *suffix* dari u yang terjadi lagi di *pattern* x [12] 16](#_Toc458082236)

[Gambar 2. 4 *Bad-character shift*, b terdapat di *pattern* x [12] 16](#_Toc458082237)

[Gambar 2. 5 *Bad-character shift*, b tidak ada di *pattern* x [12] 17](#_Toc458082238)

[Gambar 2. 6 *Software Engineering Layers* [13] 17](#_Toc458082239)

[Gambar 2. 7 *Web Development Life Cycle Model* (WDLC) 19](#_Toc458082240)

[Gambar 2. 8 *Usage statistics of Web technologies* [19] 22](#_Toc458082241)

[Gambar 3. 1 *Use case diagram* aplikasi *Lost and Found* 28](#_Toc458082245)

[Gambar 3. 2 *Sequence diagram* halaman *login* 35](#_Toc458082246)

[Gambar 3. 3 *Sequence diagram* lihat barang temuan 36](#_Toc458082247)

[Gambar 3. 4 *Sequence diagram* lihat dan *edit* barang hilang 37](#_Toc458082248)

[Gambar 3. 5 *Sequence diagram* tambah barang temuan 38](#_Toc458082249)

[Gambar 3. 6 *Sequence diagram* Tambah Barang Hilang 39](#_Toc458082250)

[Gambar 3. 7 *Sequence diagram search* barang 40](#_Toc458082251)

[Gambar 3. 8 *Sequence diagram* tambah konfirmasi barang 40](#_Toc458082252)

[Gambar 3. 9 *Sequence diagram* lihat konfirmasi barang 41](#_Toc458082253)

[Gambar 3. 10 *Sequence diagram Logout* 42](#_Toc458082254)

[Gambar 3. 11 *Class Diagram* Aplikasi *Lost and Found* 43](#_Toc458082255)

[Gambar 3. 12 Data Model *Lost and Found* 44](#_Toc458082256)

[Gambar 3. 13 Kerangka Penelitian 47](#_Toc458082257)

[Gambar 3. 14 *Flowchart* Algoritma *Boyer-Moore* [24] 50](#_Toc458082258)

[Gambar 4. 1 GUI Halaman Awal *User* 53](#_Toc458082259)

[Gambar 4. 2 Halaman *Home* 54](#_Toc458082260)

[Gambar 4. 3 Halaman Barang Temuan 55](#_Toc458082261)

[Gambar 4. 4 Tampilan Barang Hilang 55](#_Toc458082262)

[Gambar 4. 5 *Insert* Data Barang 56](#_Toc458082263)

[Gambar 4. 6 Konfirmasi Barang 56](#_Toc458082264)

[Gambar 4. 7 Halaman *Login* 58](#_Toc458082265)

[Gambar 4. 8 Halaman Awal Barang Temuan 58](#_Toc458082266)

[Gambar 4. 9 Tampilan Menu Barang Hilang 59](#_Toc458082267)

[Gambar 4. 10 Tambahkan data barang temuan atau hilang 60](#_Toc458082268)

[Gambar 4. 11 Formulir konfirmasi barang 60](#_Toc458082269)

[Gambar 4. 12 Tampilan details barang temuan 61](#_Toc458082270)

[Gambar 4. 13 Tampilan data pemilik 62](#_Toc458082271)

[Gambar 4. 14 Notifikasi barang telah diambil 62](#_Toc458082272)

[Gambar 4. 15 Pesan Konfirmasi barang hilang via email 63](#_Toc458082273)

[Gambar 4. 16 Konfirmasi pengambilan barang via email 63](#_Toc458082274)

[Gambar 4. 17 Program JavaScript untuk Fitur Auto-Complete 65](#_Toc458082275)

[Gambar 4. 18 Kode autocomplete.php 65](#_Toc458082276)

[Gambar 4. 19 Fungsi cari() dalam boyermoore.php 66](#_Toc458082277)

[Gambar 4. 20 Bentuk String yang akan diolah 66](#_Toc458082278)

[Gambar 4. 21 Fungsi makechartable() 66](#_Toc458082279)

[Gambar 4. 22 Search dengan fitur auto-complete kondisi pertama 67](#_Toc458082280)

[Gambar 4. 23 Hasil search pada kondisi pertama 68](#_Toc458082281)

[Gambar 4. 24 Search dengan fitur autocomple kondisi kedua 68](#_Toc458082282)

[Gambar 4. 25 Hasil dari pencarian string kondisi kedua 69](#_Toc458082283)

[Gambar 4. 26 Grafik Perhitungan Skor 77](#_Toc458082284)

# **DAFTAR TABEL**

[Tabel 2. 1 Algoritma *Boyer-Moore* 7](#_Toc450636192)

[Tabel 2. 2 Rangkuman Penelitian Terdahulu 8](#_Toc450636193)

[Tabel 2. 3 Perbandingan dari kasus terbaik dan kasus terburuk [9] 13](#_Toc450636194)

[Tabel 2. 4 Contoh Algoritma *Boyer-Moore* 14](#_Toc450636195)

[Tabel 2. 5 Contoh Algoritma *Boyer-Moore* 14](#_Toc450636196)

[Tabel 2. 6 Perbandingan model pengembangan aplikasi [15] 17](#_Toc450636197)

[Tabel 2. 7 Tabel Perbandingan antara ASP.NET dan PHP [20] 21](#_Toc450636198)

[Tabel 3. 1 *Use Case Scenario Login* 28](#_Toc458082334)

[Tabel 3. 2 *Use Case Scenario* Mencari Data Barang Hilang 29](#_Toc458082335)

[Tabel 3. 3 *Use Case Scenario* Mencari Data Barang Temuan 30](#_Toc458082336)

[Tabel 3. 4 *Use Case Scenario* Melihat Data Barang Hilang 30](#_Toc458082337)

[Tabel 3. 5 *Use Case Scenario* Melihat Data Barang Temuan 31](#_Toc458082338)

[Tabel 3. 6 *Use Case Scenario* Mengedit Barang Hilang 32](#_Toc458082339)

[Tabel 3. 7 *Use Case Scenario* Membuat Laporan Kehilangan 32](#_Toc458082340)

[Tabel 3. 8 *Use Case Scenario* Membuat Konfirmasi Barang 33](#_Toc458082341)

[Tabel 3. 9 *Use Case Scenario* Membuat Laporan Penemuan 34](#_Toc458082342)

[Tabel 3. 10 Penentuan Kriteria [2] 45](#_Toc458082343)

[Tabel 4. 1 Penentuan *Pattern* dan Teks setelah jumlah hurufnya disamakan 69](#_Toc458082349)

[Tabel 4. 2 Simulasi Cara Kerja Algoritma *Brute Force* 71](#_Toc458082350)

[Tabel 4. 3 Simulasi Cara Kerja Algoritma *Boyer-Moore* 73](#_Toc458082351)

[Tabel 4. 4 Pembobotan Kriteria 75](#_Toc458082352)

[Tabel 4. 5 Pemberian Nilai Pada Setiap Kriteria 75](#_Toc458082353)

[Tabel 4. 6 Simulasi Perhitungan Analisa Menggunakan Perhitungan Perbandingan Eksponensial 76](#_Toc458082354)

[Tabel 4. 7 Prioritas Keputusan 77](#_Toc458082355)

# **DAFTAR RUMUS**

[Rumus 2. 1 Rumus Metode Perbandingan Eksponensial [2] 25](#_Toc458082469)

# **DAFTAR SINGKATAN**

|  |  |
| --- | --- |
| ASCII | *American Standard Code for Information Interchange* |
| CSS | *Cascading Style Sheet* |
| HTML | *Hyper Text Markup Language* |
| J2EE | *Java 2, Enterprise Edition* |
| JSON | *JavaScript Object Nation* |
| KMP | Knuth Morris Pratt |
| MPE | Metode Perbandingan Eksponensial |
| MVC | *Model View Controller* |
| PHP | *Hypertext Preprocessor* |
| RAD | *Rapid Application Development* |
| SDLC | *Software Development Life Cycle* |
| SI | Sistem Informasi |
| UML | *Unified Modeling Language* |
| WDLC | *Web Development Life Cycle* |

# **BAB I**

**PENDAHULUAN**

1. **Latar Belakang Masalah**

Berdasarkan kemajuan teknologi di setiap instansi di Inodonesia, penerapan sistem berbasis teknologi berfungsi untuk mempermudah setiap kegiatan yang dilakukan secara manual. Penerapan teknologi ini untuk mendukung penyampaian informasi yang dapat diakses dengan mudah, cepat, dan sesuai dengan kebutuhan. Selain itu, untuk membuat implementasi yang sesuai dengan kegiatan operasional, dan penyediaan informasi untuk mendukung ketersediaan pelayanan publik [1].

Kehilangan barang-barang pribadi di Universitas Bakrie merupakan hal yang sudah sering terjadi. Sebagian besar mahasiswa yang merasa kehilangan atau menemukan barang di lingkungan Universitas Bakrie akan segera melapor ke petugas *security* terdekat. Hal ini yang membuat *staff* *security* merasa kesulitan dalam melakukan penyimpanan dan pencarian barang yang telah ditemukan atau barang yang dicari karena jangka waktu penemuan terlalu lama ataupun catatan yang sudah menumpuk. Dalam satu minggu, pihak *security* bisa menerima laporan kehilangan atau penemuan sebanyak 10 kali laporan, jumlah ini termasuk cukup banyak mengingat jumlah mahasiswa di Universitas Bakrie yang sangat banyak (Lampiran II.).

Dari permasalahan tersebut, perlu adanya sebuah sistem atau sarana khusus yang dapat menampung data temuan barang dan lokasi penyimpanan barang tersebut. Salah satunya dengan membangun sistem informasi *Lost and Found* berbasis *web* yang dapat diakses oleh *staff* *security* di berbagai pos keamanan di Universitas Bakrie. Dengan adanya sistem informasi temuan barang ini akan memudahkan *staff* *security* untuk mengelola temuan barang dan laporan kehilangan secara efektif.

Dalam pembangunan sistem informasi *Lost and Found* dibutuhkan suatu metode pencarian yang dapat memudahkan sistem untuk melakukan pencarian. Hadirnya mesin pencarian *(Search Engine)* di dalam sistem informasi memudahkan pengguna komputer dalam mencari berbagai informasi. Untuk memudahkan penggunanya, *Search Engine* menambahkan fitur pencari sugesti hasil terdekat pencarian yaitu menggunakan fitur *Word Suggestion* [2]*.* Dalam pencarian fitur *Word Suggestion,*  diperlukan sebuah algoritma dalam pencarian *string*. Algoritma yang digunakan untuk pencarian *string* semakin berkembang dari hari ke hari. Tujuan utamanya tentu saja untuk mencari *string* seakurat dan secepat mungkin. Sampai saat ini algoritma pencarian *string* dibagi menjadi 3 kategori berdasarkan arah pencocokan *string* yaitu dari kiri ke kanan, kanan ke kiri, dan dari arah yang ditentukan secara spesifik. Metode pencocokan dari kiri ke kanan merupakan metode yang paling natural karena sesuai dengan arah membaca, pencocokan *string* dari kanan ke kiri merupakan metode yang dianggap paling efisien dalam praktiknya, dan pencocokan *string* dari arah yang ditentukan secara spesifik merupakan algoritma yang memiliki hasil yang paling baik secara teoritis [3].

Algoritma yang dianggap memiliki hasil yang paling baik dalam praktiknya merupakan algoritma yang bergerak mencocokkan *string* dari arah kanan ke kiri. Algoritma *Boyer-Moore* adalah salah satu contoh algoritma yang menggunakan arah dari kanan ke kiri. Algoritma ini telah banyak dikenal oleh masyarakat dan dianggap paling efisien untuk pencarian *string.* Ide dibalik algoritma ini adalah bahwa dengan memulai pencocokan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat.

1. **Identifikasi Masalah**

Ditinjau dari latar belakang dan masalah-masalah di atas dapat diidentifikasikan bahwa:

1. Sistem penemuan barang di Universitas Bakrie yang dikelola oleh bagian keamanan masih dilakukan secara manual dengan menggunakan *document based*  sehingga masih mengalami kesulitan dalam proses pencarian data barang yang ditemukan
2. Adanya kecenderungan hilangnya buku daftar temuan barang sebagai sumber data barang pencarian dan barang temuan
3. Pencatatan manual yang dilakukan oleh bagian keamanan menyebabkan kelambatan pencarian data barang hilang atau barang yang ditemukan.
4. **Rumusan Masalah**

Berdasarkan latar belakang di atas, dirumuskan masalah dalam penelitian ini adalah bagaimana penerapan algoritma *Boyer-Moore* dalam sistem informasi *Lost and Found* berbasis *web*?

1. **Batasan Masalah**

Untuk membatasi pembahasan agar tidak keluar dari konteks topik penelitian, maka batasan dalam pembahasan masalah sebagai berikut:

1. Sistem *Lost and Found* dibangun sebagai sarana mempermudah bagian kemanan Universitas Bakrie
2. Sistem *Lost and Found* hanya mengelola pencatatan barang yang telah ditemukan dan mendata laporan kehilangan dari civitas akademika Universitas Bakrie.
3. **Tujuan Penelitian**

Tujuan dilakukannya penelitian ini adalah dapat menerapkan dan membuktikan algoritma *Boyer-Moore* sehingga mempercepat proses pencarian data.

1. **Manfaat Penelitian**

Manfaat yang akan diperoleh dengan adanya penelitian ini adalah

1. Memudahkan bagian *security* dalam melakukan pencatatan dan pencarian data barang yang telah ditemukan serta mengurangi kehilangan data inventaris
2. Menambah referensi bagi peneliti lain dalam penerapan algoritma *Boyer-Moore* dalam proses pencarian.
3. **Sistematika Penulisan**

Sistematika penulisan menguraikan secara singkat mengenai penelitian yang dilakukan. Adapun sistematika penulisan adalah sebagai berikut:

**BAB I PENDAHULUAN**

Bab ini menjelaskan mengenai latar belakang penelitian, identifikasi masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

**BAB II TINJAUAN PUSTAKA**

Bab ini menjelaskan *review* dari penelitian sebelumnya. Teori tersebut antara lain *Web Development Life Cycle,* Algoritma *Boyer-Moore,* dan *Unified Modeling Language.*

**BAB III METODE PENELITIAN**

Bab ini menjelaskan metode penelitian dan alokasi waktu yang dibutuhkan dalam menyelesaikan penelitian ini.

**BAB IV IMPLEMENTASI DAN PENGUJIAN**

Bab ini menjelaskan mengenai hasil implementasi algoritma *Boyer-Moore* dan pengujian meggunakan Metode Perbandingan Eksponensial.

**BAB V SIMPULAN DAN SARAN**

Bab ini menjelaskan simpulan dan saran yang mencakup seluruh hasil dari penelitian yang telah dilakukan

# **BAB II**

**TINJAUAN PUSTAKA**

1. **Penelitian Terdahulu**

Dasar atau acuan yang berupa teori-teori atau temuan-temuan melalui hasil berbagai penelitian sebelumnya merupakan hal yang perlu dan dapat dijadikan data pendukung. Salah satu data pendukung yang menurut peneliti perlu dijadikan bagian tersendiri adalah penelitian terdahulu yang relevan dengan permasalahan yang sedang dibahas dalam penelitian ini. Dalam hal ini, fokus penelitian terdahulu yang dijadikan acuan adalah terkait dengan penerapan algoritma *Boyer-Moore* dan sistem *Lost and Found.*  Oleh karena itu, peneliti melakukan langkah kajian terhadap beberapa hasil penelitian berupa tesis dan jurnal melalui internet.

Berikut ini adalah beberapa pemaparan singkat tentang perbandingan dari analisis yang akan dilakukan dengan menganalisis jurnal

1. Analisa Perbandingan Algoritma *Brute Force* dan *Boyer-Moore* dalam pencarian *Word Suggestion* Menggunakan Metode Perbandingan Eksponensial (2013).

Dalam penelitian ini dirancang sebuah aplikasi *Word Suggestion,* yang merupakan aplikasi pencarian sugesti hasil terdekat dengan menggunakan *string matching* dalam sebuah pencarian, dalam perancangannya aplikasi tersebut membandingkan dua metode algoritma pencarian. Membandingkan metode algoritma *Boyer-Moore* dan *Brute Force* dengan beberapa percobaan dan menghitung keefektifan kedua algoritma menggunakan metode Perbandingan Eksponensial. Hasil dari penelitian tersebut adalah dari beberapa pencarian kata yang diperoleh berdasarkan jumlah iterasi yang ada pada kedua algoritma menunjukkan bahwa Algoritma *Boyer-Moore* memiliki jumlah iterasi paling sedikit sehingga menunjukkan Algoritma *Boyer-Moore* merupakan algoritma tercepat dibandingkan dengan Algoritma *Brute Force* [2].

1. Rancang Bangun Sistem Informasi Pencarian Benda Hilang ‘*Lost and Found’* Berbasis *Website* di Universitas Brawijaya (2012).

Penelitian tersebut dilakukan dikarenakan tingkat pencarian dan penemuan barang hilang yang cukup banyak dan belum adanya sarana yang menampung barang yang ditemukan. Pencarian barang dan penemuan barang sebelumnya dilakukan dengan menempelkan brosur di tembok yang merupakan cara tradisional dan merusak keindahan tembok UB. Sehingga, peneliti membuat sarana informasi pencarian benda hilang berbasis *web* yang terintegrasi dengan layanan BAIS (*Brawijaya Authentication and Identification System)* yang memungkinkan pengguna *single sign on* untuk aplikasi dalam domain UB. Hanya pengguna otentik saja yang dapat menggunakan layanan aplokasi *Lost and Found.* Peneliti tersebut juga menerapkan rancang bangun menggunakan *Framework Codeigniter* dengan penerapan pola desain *Model View Controller* (MVC). Dalam perancangan aplikasi tersebut peneliti menggunakan 4 diagram yaitu *use case diagram, activity diagram, sequence diagram, dan class diagram.* Hasil dari penelitian tersebut adalah aplikasi *lost and found* berhasil dibangun dan ditujukan untuk mahasiswa Universitas Brawijaya setelah dilakukan dua tahap pengujian yaitu pengujian validasi dan pengujian *feedback* dari *user* [4]*.*

1. Studi Perbandingan Implementasi Algoritma *Boyer-Moore, Turbo Boyer-Moore,* dan *Tuned Boyer-Moore* dalam Pencarian *String* (2013).

Dalam penelitian tersebut, peneliti menganalis dan membuat pencarian *string* dengan menggunakan tiga jenis Algoritma *Boyer-Moore* yaitu Algotima *Boyer-Moore, Turbo Boyer-Moore,* dan *Tuned Boyer-Moore* dengan tujuan adalah untuk mengetahui bagaimana performa algoritma-algoritma tersebut, terutama di bidang waktu yang diperlukan untuk mencari suatu *pattern* dalam *text.* Peneliti membangun sebuah aplikasi menggunakan metode *prototyping* dan menggunakan Microsoft Visual Studio dengan bahasa C#. Aplikasi tersebut mendukung pencarian dengan menggunakan tiga algoritma, pengubah kata *(replace), highlight* kata yang dicari, dan pemberian informasi waktu yang dibutuhkan masing-masing algoritma untuk pencarian serta algoritma mana yang membutuhkan waktu paling sedikit untuk pencarian. Dari penelitian yang dilakukan, dapat disumpulkan bahwa algoritma yang tercepat dalam pencarian *string* adalah algoritma *Boyer-Moore*. Algoritma Turbo *Boyer-Moore* merupakan algoritma tercepat kedua dan yang paling lambat adalah *Tuned Boyer-Moore* [5]*.*

**Tabel 2. 1 Algoritma Boyer-Moore**

|  |  |
| --- | --- |
| **Algoritma** | **Karakteristik** |
| *Boyer-Moore* | Pencocokan karakter dari kanan ke kiri dan bukan dari kiri ke kanan sehingga akan lebih banyak informasi yang didapat |
| *Turbo Boyer-Moore* | Membutuhkan ruang lebih tetapi membutuhkan pemrosesan ekstra. Ruang ekstra yang diperlukan berguna untuk mengingat faktor dari teks yang cocok dengan akhiran dari *string* yang dicari selama *attempt* terakhir dan hanya jika *good-suffix* dilakukan |
| *Tuned Boyer-Moore* | Fitur utama dari algoritma ini adalah simplifikasi dari algoritma *Boyer-Moore,* mudah untuk diimplementasikan, hanya menggunakan *bad-character shift,*  dan sangat cepat dalam praktiknya |

1. Perancangan dan Pembuatan Sistem Informasi Kehilangan Berbasis *Web* (2014).

Penelitian tersebut bertujuan untuk membuat aplikasi berbasis *web* untuk sistem informasi kehilangan di Universitas Muhammadiyah Surakarta. Penelitian ini dilakukan mengingat mahasiswa merasa kesulitan dalam menemukan barang yang telah hilang dan mahasiswa tidak dapat mengandalkan pihak satpam saja, sehingga dibutuhkan suatu sarana yang dapat diakses oleh semua pihak untuk menemukan barang dan dapat dijadikan arsip oleh pihak satpam setiap bulannya. Perancangan dilakukan menggunakan *tool* *software ApacheFriends XAMPP* (Barispaket) *version 1.6.7 (MySQL 5.0.51 (Community Server), PHP 5.2.6* dan PHP 5.2.6 dan *phpMyAdmin 2.11.7),* dengan *web* desainer *Macromedia Dreamweaver 8.* Sistem kehilangan berbasis *web* tersebut sudah dibenahi dan dibuat dengan menggunakan bahasa pemrograman PHP dan *database* *MySQL*. Dari hasil pengujian dengan berbagai macam internet *browser* secara *localhost* maupun *online*  dapat dilihat bahwa sistem dapat berjalan lancar [6].

**Tabel 2. 2 Rangkuman Penelitian Terdahulu**

| **No** | **Judul** | **Pengarang** | **Tahun** | **Permasalahan** | **Hasil** |
| --- | --- | --- | --- | --- | --- |
| 1 | Analisa Perbandingan Algoritma *Brute Force* dan *Boyer-Moore* dalam pencarian *Word Suggestion* Menggunakan Metode Perbandingan Eksponensial | Andri Januardi | 2013 | Pada proses perbandingan kecepatan dua algoritma pencarian dalam *word suggestion* yang dilakukan. | Dibuat sebuah aplikasi *word suggestion* pada mesin pencarian dengan menggunakan dua perbandingan algoritma yaitu *Boyer-Moore* dan *Brute Force,* dan hasil dari cara kerja kedua algoritma menunjukkan bahwa Algoritma *Boyer-Moore* merupakan algoritma paling efektif daripada *Brute Force.* |
| 2 | Rancang Bangun Sistem Inforamsi Pencarian Benda Hilang ‘*Lost and Found ’* Berbasis *Website* di Universitas Brawijaya. | Dedi Arief Wibisono, Diah Priharsari, ST., MT, Adharul Muttaqin, ST., MT | 2012 | Pencarian barang hilang dengan menempelkan info kehilangan di tembok-tembok, belum tentu orang mempedulikan info kehilangan tersebut. Terlebih, pengumuman kehilangan tersebut hanya akan mengotori tembok dan mengganggu pemandangan. | Dibuat sebuah aplikasi berbasis *web* yang diintergrasikan dengan menggunakan NIM & *password* yang dimiliki mahasiswa, maka mahasiswa dapat login di aplikasi dan dapat memberi informasi mengenai kehilangan/penemuan benda. |
| 3 | Studi Perbandingan Implementasi Algoritma *Boyer-Moore, Turbo Boyer-Moore,* dan *Tuned Boyer-Moore* dalam Pencarian *String*. | Vina Sagita, Maria Irmina Prasetiyowati | 2013 | Seiring dengan banyaknya pencarian *string* dalam jumlah besar, maka diperlukan pencocokan karakter menggunakan algoritma yang paling efektif dalam *string searching.* | Peneliti membandingkan dan membangun sebuah aplikasi yang dapat menghitung kecepatan sebuah pencarian data berdasarkan algoritma yang dipakai, hasilnya adalah algoritma *Boyer-Moore* merupakan algoritma tercepat jika dibandingkan dengan algoritma *Turbo Boyer-Moore* dan *Tuned Boyer-Moore.* |
| 4 | Perancangan dan Pembuatan Sistem Informasi Kehilangan Berbasis *Web* | Supriyanto | 2014 | Kasus hilangnya KTM atau barang lain seperti STNK sering terjadi di lingkungan UMS, dan tidak ada tindak lanjut yang pasti dari pihak satpam, pihak satpam hanya memberikan laporan dan tidak memberikan kepastian apakah barang ditemukan atau tidak. Sehingga tidak ada suatu wadah yang memungkinkan untuk informasi mengenai barang hilang | Perancangan *website* kehilangan dengan menggunakan *software ApacheFriends* XAMPP (Barispaket) *version* 1.6.7 (*MySQL* 5.0.51 *(Community Server),* PHP 5.2.6 dan *phpMyAdmin 2.11.7),* dengan *web* desainer *Macromedia Dreamweaver 8.* |

1. **Konsep Dasar Sistem Informasi**
2. Definisi Sistem Informasi

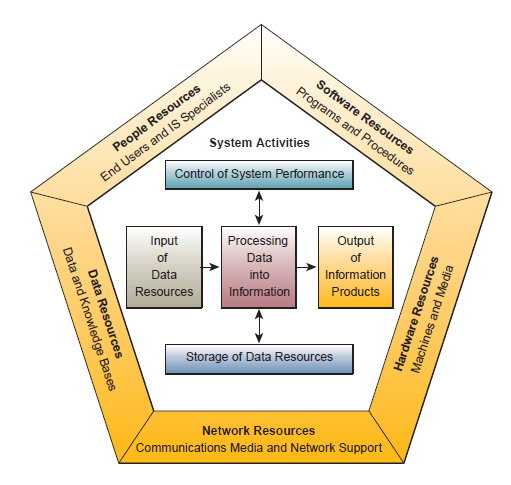
Suatu sistem informasi (SI) dapat berupa kombinasi yang terorganisir pada manusia, *hardware, software,* jaringan komunikasi, *data resources,* kebijakan dan prosedur yang menyimpan, mengambil, mengubah dan menyebarkan informasi dalam sebuah organisasi. Manusia mengandalkan sistem informasi modern untuk berkomunikasi dengan satu sama lain menggunakan berbagai perangkat fisik *(hardware),* instruksi pemrosesan informasi dan prosedur *(software),* saluran komunikasi *(networks),* dan data yang tersimpan *(data resources)* [7].

Menurut Al-Bahra, sistem informasi dapat didefinisikan sebagai berikut :

1. Suatu sistem yang dibuat oleh manusia yang terdiri dari komponen-komponen dalam organisasi untuk mencapai suatu tujuan yaitu menyajikan informasi.
2. Sekumpulan prosedur organisasi yang pada saat dilaksanakan akan memberikan informasi bagi pengambil keputusan dan atau untuk mengendalikan organisasi.
3. Suatu sistem dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi, mendukung operasi, bersifat manajerial, dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan [8].
4. Komponen Sistem Informasi

Model sistem informasi ini menyoroti hubungan antara komponen-komponen dan kegiatan sistem informasi. Hal ini juga menyediakan kerangka kerja yang menekankan pada empat konsep utama yang dapat diterapkan untuk semua jenis sistem informasi:

1. Manusia, *hardware, software,* data, dan jaringan adalah lima sumber dasar sistem informasi
2. Sumber daya manusia termasuk pengguna akhir dan IS *Specialist,* sumber daya perangkat keras terdiri dari mesin dan media, sumber daya perangkat lunak mencakup program dan prosedur, sumber data meliputi data dan basis pengetahuan, dan sumber daya jaringan meliputi media komunikasi dan jaringan
3. Sumber data yang diubah oleh kegiatan pengolahan informasi menjadi berbagai produk informasi bagi pengguna akhir *(end user)*
4. Pengolahan informasi terdiri dari kegiatan sistem *input*, pengolahan, *output, storage,* dan kontrol [7].



**Gambar 2. 1 Komponen sebuah Sistem Informasi** [7]

1. **Konsep Dasar *String Searching***

*String* adalah urutan dari karakter, yang mana karakter ini dapat terdiri dari beberapa alfabet. Misalnya adalah *string* biner yang terdiri dari dua alfabet, yaitu 0 dan 1, jadi *string* biner merupakan suatu urutan karakter 0 maupun 1. Contoh lain adalah *string American Standard Code for Information Interchange* (ASCII) yang terdiri dari 256 alfabet.

*String searching* pada dasarnya adalah mencari pola P yang memiliki panjang m dalam suatu teks T yang memiliki panjang n. Baik pola maupun teks dinyatakan dalam *array,* pola dinyatakan dengan P[0 ... m-1] dan teks dinyatakan dengan T[0 ... n-1].

Kecocokan adalah apabila karakter pada teks dan karakter pada pola dibandingan adalah sama sedangkan ketidakcocokan adalah sebaliknya.

*window*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | A | A | B | B | A | A | B | A | C |
| A | A | B | A | C |

* Abu-abu muda menunjukkan kecocokan
* Abu Abu tua menunjukkan ketidakcocokan

*Window* adalah suatu kotak pada teks yang mempunyai ukuran sama dengan panjang pola, fungsinya untuk membantu dalam pencarian pola, pertama kali *window* ini diletakkan di posisi paling kiri dari teks. Kemudian dilakukan percobaan, yaitu perbandingan karakter-karakter di *window* dengan karakter-karakter di pola. Pergeseran *window* ke kanan akan dilakukan jika terjadi dua sebab, yang pertama adalah jika terjadi kecocokan dari semua karakter di pola atau berarti pola ditemukan di teks. Pergeseran karena sebab yang pertama ini dilakukan untuk mencari pola yang berikutnya. Sebab yang kedua adalah jika terjadi ketidakcocokan. Mekanisme ini diulang terus-menerus sampai batas kanan dari *window* melebihi batas kanan dari teks.

Misalnya S adalah sebuah *string* dengan panjang m maka ada beberapa bagian dari *string:*

* *Substring* S[i ... j] adalah bagian dari *string* antara i dan j

|  |  |  |
| --- | --- | --- |
|  |  |  |

0 i j m-1

* *Prefix* dari S adalah sebuah *substring* yaitu S[0 ... i]

|  |  |
| --- | --- |
|  |  |

0 i

* *Suffix* dari S adalah sebuah *substring* yaitu S[i ... m-1]

|  |  |
| --- | --- |
|  |  |

i m-1

Dimana i dan j adalah suatu indeks *array* antara 0 dan m-1

Contoh:

Sebuah *string S*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | n | D | r | E | W |

0 5

Panjang *string =* 6

*Substring* S[1 ... 3] = “ndr”

Semua kemungkinan *prefix* dari S:

“andrew”, “andre”, “andr”, “and”, “an”, “a”

Semua kemungkinan *suffix* dari S:

“andrew”, “ndrew”, “drew”, “rew”, “ew”, “w”

Dalam penerapan *string searching.* terdapat beberapa algoritma dengan proses pencarian yang berbeda-beda. Adapun algoritma-algoritma yang telah berkembang adalah algoritma *Brute Force*, *Knuth Morris Pratt* (KMP)*, Boyer-Moore, Karp Rabin,* dll.

**Tabel 2. 3 Perbandingan dari kasus terbaik dan kasus terburuk** [9]

|  |  |  |
| --- | --- | --- |
| **Algoritma** | ***Best Case*** | ***Worse Case*** |
| Brute Force | m+n | m.n |
| KMP | m+n | m+n |
| *Boyer-Moore* | n/m | m+n |
| Karp Rabin | m+n | m.n |

1. **Algoritma *Boyer-Moore***

Menurut Wibisono [3], algoritma *Boyer-Moore* adalah salah satu algoritma untuk mencari *string* di dalam teks, 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 kecocokan maka dilakukan perbandingan karakter teks dan karakter pola yang sebelumnya, yaitu dengan sama-sama mengurangi indeks teks dan pola masing-masing sebanyak satu [10].

Menurut Chiquita [11], dengan menggunakan algoritma *Boyer-Moore* ini, secara rata-rata proses pencarian akan menjadi lebih cepat jika dibandingkan dengan algoritma lainnya. Alasan melakukan pencocokan dari kanan (posisi terakhir *string* yang dicari) ditunjukkan dalam contoh berikut:

**Tabel 2. 4 Contoh Algoritma Boyer-Moore**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M | A | K | A | N |  | T | O | M | A | T |
| T | O | M | A | T |  |  |  |  |  |  |

Pada contoh di atas, dengan melakukan perbandingan dari posisi paling akhir *string* dapat dilihat bahwa karakter “n” pada *string* “makan” tidak cocok dengan karakter “t” pada *string* “tomat” yang dicari, dan karakter “n” tidak pernah ada dalam *string* “tomat” yang dicari sehingga *string* “tomat” dapat digeser melewati *string* “makan”, sehingga posisinya seperti berikut.

**Tabel 2. 5 Contoh Algoritma Boyer-Moore**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M | A | K | A | N |  | T | O | M | A | T |
|  |  |  |  |  | T | O | M | A | T |  |

Dalam contoh terlihat bahwa algoritma *Boyer-Moore* 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 [10].

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

1. Algoritma *Boyer-Moore* mulai mencocokkan *pattern* pada awal teks
2. Dari kanan ke kiri, algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter di teks yang bersesuaian sampai salah satu kondisi berikut dipenuhi:
   1. Karakter di *pattern* dan di teks yang dibandingkan tidak cocok *(mismatch)*
   2. Semua karakter di *pattern* cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
3. Algoritma kemudian menggeser *pattern* dengan memaksimalkan nilai pergeseran *good-suffix* dan pergeseran *bad-character,* lalu mengulangi langkah 2 sampai *pattern* berada di ujung teks

Algoritma *Boyer-Moore* ini juga memiliki beberapa aturan untuk pergeseran *pattern* yaitu *good-suffix rule* dan *bad character rule.*

1. *Good-suffix shift rule*

*Good-suffix rule* hanya membandingkan karakter yang sudah cocok ke karakter *pattern,* aturan dari *good-suffix rule* adalah sebagai berikut:

1. Pergeseran dari x[i]=a ke karakter lain yang letaknya lebih kiri dari x[i] dan terletak di sebelah kiri segmen u.



**Gambar 2. 2 Good-suffix shift, u terjadi lagi didahului karakter c berbeda dari a** [12]

1. Jika tidak ada segmen yang sama dengan u, maka dicari u yang merupakan *suffix* terpanjang u.



**Gambar 2. 3 Good-suffix shift, hanya suffix dari u yang terjadi lagi di pattern x** [12]

1. *Bad-character rule*

*Bad-character rule* hanya membandingkan karakter yang tidak cocok ke karakter *pattern*, aturan dari *bad-character rule* adalah sebagai berikut:

1. Jika *bad-character* y[i+j] terdapat pada *pattern* di posisi tekanan k yang lebih kiri dari x[i] maka *pattern* digeser ke kanan sejauh i-k



**Gambar 2. 4 Bad-character shift, b terdapat di pattern x** [12]

1. Jika *bad-character* y[i+j] tidak ada *pattern* sama sekali, maka *pattern* digeser ke kanan sejauh i



**Gambar 2. 5 Bad-character shift, b tidak ada di pattern x** [12]

1. Jika *bad-character* y[i+j] terdapat pada *pattern* di posisi tekanan k yang lebih kanan dari x[i], maka *pattern* seharusnya digeser sejauh i-k yang hasilnya negatif (*pattern* digeser kembali ke kiri). Maka bila kasus ini terjadi akan diabaikan.

Pada kasus ketidakcocokan di atas, algoritma akan membandingkan langkah yang diambil oleh fungsi *good-suffix shift* dan *bad-charater shift* dimana langkah yang paling besar yang akan digunakan [12].

1. **Model Siklus Pengembangan Perangkat Lunak**

Pengembangan perangkat lunak dapat diartikan sebagai proses membuat suatu perangkat lunak baru untuk menggantikan perangkat lunak lama secara keseluruhan atau memperbaiki perangkat lunak yang telah ada. Agar lebih cepat dan tepat dalam mendeskripsikan solusi dan mengembangkan perangkat lunak, juga hasilnya lebih mudah untuk dikembangkan dan dipelihara, maka pengembangan perangkat lunak memerlukan suatu metodologi khusus. Metodologi pengembangan perangkat lunak adalah suatu pengorganisasian kumpulan metode dan konvensi notasi yang telah didefinisikan untuk mengembangkan perangkat lunak. Berikut batu landasan yang menopang rekayasa perangkat lunak [13, p. 14].

**Gambar 2. 6 Software Engineering Layers** [13]

Setiap pengembangan perangkat lunak tidak akan terlepas dari sebuah SDLC *(Software Development Life Cycle).* Dalam SDLC terdapat banyak model pengembangan yang dapat dipakai untuk mengembangkan sebuah perangkat lunak aplikasi yaitu: *Waterfall, Spiral, RAD (Rapid, Application Development),* dan sebagainya [14]. Penggunaan model yang sesuai dengan kebutuhan pada projek pengembangan aplikasi yang dilakukan akan berdampak pada kualitas aplikasi. Maka dari itu, pengembang aplikasi harus dapat mengukut aplikasi yang akan dibuat sebelum memilih SDLC yang sesuai dalam praktik pengembangan aplikasi. Untuk meningkatkan kualitas pengembangan aplikasi, maka perlu untuk mengetahui karakteristik dari masing-masing model pengembangan dan menyesuaikan kebutuhan dari beberapa aspek dalam projek yang dilakukan.

**Tabel 2. 6 Perbandingan model pengembangan aplikasi** [15]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Model/Features*** | ***Waterfall*** | ***Incremental*** | ***Spiral*** | ***Agile*** | ***RUP*** |
| ***Requirement Specifications*** | *Beginning* | *Beginning* | *Beginning* | *Frequently changed* | *Beginning* |
| ***Cost*** | *Low* | *Low* | *Expensive* | *Very High* | *Expensive* |
| ***Resource Control*** | *Yes* | *Yes* | *Yes* | *No* | *Yes* |
| ***Simplicity*** | *Simple* | *Intermediate* | *Intermediate* | *Complex* | *Simple and clear* |
| ***Risk Analysis*** | *Only at beginning* | *Intermediate* | *High* | *High* | *Only at beginning of last phase* |
| ***Flexibility*** | *Rigid* | *Less Flexible* | *Flexible* | *Highly Flexible* | *Considerable* |
| ***Reusability*** | *Limited* | *Yes* | *Yes* | *Use Case reuse* | *Support reusability of existing classes* |

*Web Development Life Cycle* (WDLC) adalah suatu metodologi baru yang diusulkan khusus untuk pengembangan aplikasi *web*. Metodologi ini didasarkan pada metodologi sebelumnya yang ditemukan dalam literatur untuk menciptakan suatu proses terstruktur untuk masalah yang sangat terstruktur dari pengembangan aplikasi *web* itu sendiri. WDLC adalah hibrida dari dua metodologi sebelumnya yang dikenal sebagai *Systems Development Life Cycle and Prototyping.* WDLC menggunakan komponen dari masing-masing metodologi, menggabungkan ke dalam sebuah pendekatan baru yang akan mengurangi waktu pengembangan, menambahkan struktur untuk masalah yang tidak terstruktur dan menjaga pengguna yang terlibat dalam seluruh siklus hidup pengembangan [16].

Ada lima tahapan WDLC yang memungkinkan proses perancangan selesai. Masing-masing tahapan mencakup seperangkat tugas, yang mengandalkan teknik yang menghasilkan *file* dokumen tertentu untuk memahami proyek [17].



**Gambar 2. 7 Web Development Life Cycle Model (WDLC)**

1. *Website Planning*

Fase pertama dalam WDLC adalah *planning.* Beberapa tahapan yang harus dilakukan antara lain:

1. Mengidentifikasi tujuan dari *website* yang akan dibangun, sehingga dapat menentukan perencanaan secara tepat
2. Memahami siapakah yang akan menggunakan *website,* hal ini merupakan tahapan untuk mengidentifikasi target pengguna, menentukan halaman yang akan diakses oleh pengguna, dan mengidentifikasi teknologi yang dibutuhkan dalam mengakses *website* tersebut
3. Memahami teknologi *website* apa yang akan digunakan, seperti *web browser,* akses internet, dan resolusi layar monitor
4. Mengidentifikasi isi konten dari *website* yang dibedakan berdasarkan penggunanya
5. Menentukan informasi apa saja yang perlu diletakkan di dalam *website* tersebut.
6. *Website Analysis*

Tahapan ini merupakan rangkaian aktivitas dimana seorang analis menggabungkan informasi yang dibutuhkan oleh pengguna, menganalisis kebutuhan fungsional dari sistem, kebutuhan masukan data dan sumber daya, serta kebutuhan presentasi dan keluaran data. Berikut ini adalah tahapan-tahapan yang perlu dilakukan.

1. Mengidentifikasi tugas atau pekerjaan yang harus diselesaikan oleh masing-masing pengguna sistem
2. Mengidentifikasi *site map,* menentukan struktur dari *website,* dan finalisasi konten yang akan diletakkan pada halaman *web*
3. Menganalisa kualitas kebutuhan data yang memang benar-benar dibutuhkan oleh pengguna agar dapat dihasilkan keluaran yang benar dan tepat.
4. *Web Page Design and Development*

Tahapan ini meliputi *blue print* dari *website* dengan mempresentasikannya ke dalam desain *logical* dan *physical* yang akan dibangun pada tahapan *development.* Desain yang dibuat meliputi *data models, process models,* dan *presentation models.* Desain tersebut dibuat dalam bentuk dokumen sebagai panduan dalam pengembangan dan pengujian sistem.

Seorang *developer* memiliki tanggung jawab dalam pembangunan kode program, dan membuat *data sets* untuk masukan serta memverifikasi bahwa program dapat menghasilkan keluaran sesuai yang diharapkan oleh pengguna. Hanya pada tahapan ini, konseptual *website* diterjemahkan ke dalam sebuah *website* yang bermanfaat dan atraktif.

1. *Website Testing*

Tim pengembangan mendemonstrasikan *website* kepada pengguna. Mereka memastikan bahwa *website* berjalan sesuai dengan yang diharapkan oleh pengguna. Hal ini meliputi perencanaan pengujian, membuat *text data,* mengeksekusi *text runs,* mencocokkan hasil teks sesuai dengan yang diharapkan, menganalisa dan memperbaiki *bugs* yang terjadi hingga tidak terjadi kesalahan. *Website* harus diuji pada tahapan yang berbeda, yang meliputi *content, functionality, usability,* dan *correctnest.*

1. *Website Implementation and Maintenance*

Tahapan ini meliputi instalasi *website* pada sistem komputer yang dipakai oleh pengguna. Selanjutnya adalah tahapan pemeliharaan *website* yang ditujukan untuk memastikan bahwa kebutuhan informasi masih sesuai dengan yang diharapkan. Hal ini dapat menjaga agar *website*  tetap *up to date.*

1. **Bahasa Pemrograman**

Teknologi layanan *web* didasarkan pada konsep komputasi berorientasi layanan-layanan *web* standar yang mengintegrasikan aplikasi berbasis *web* melalui menghubungkan dan berbagi proses bisnis di seluruh jaringan di mana aplikasi dari vendor yang berbeda, bahasa dan platform komunikasi satu sama lain dengan klien [18]. Beberapa teknologi terkemuka yang sebagian besar digunakan dalam mengembangkan dan menerapkan aplikasi berbasis *web* adalah ASP.NET, *Hypertext Preprocessor* (PHP) , *ColdFusion*, *Perl*, *Phyton*, *Java 2 Enterprise Edition* (J2EE), *Ruby on Rails* dll. Dari statistik di bawah ini menunjukkan bahwa PHP berada pada posisi teratas yang paling sering digunakan sekitar 24692359 *websites,* dan posisi kedua adalah ASP.NET yang digunakan lebih dari 19365837 *websites* [19]*.*



**Gambar 2. 8 Usage statistics of Web technologies** [19]

Salah satu aspek penting yang dicatat selama pengembangan adalah penggunaan memori. ASP.NET terlihat cukup mahal dengan penggunaan memori yang dapat menjadi masalah serius ketika mengembangkan aplikasi *web* yang lebih besar. Sedangkan penggunaan memori pada PHP lebih efisien daripada ASP.NET. Alasannya adalah PHP memiliki *code path* kecil yang berarti kode sisi server lebih sedikit jika dibandingkan dengan ASP.NET [20].

**Tabel 2. 7 Tabel Perbandingan antara ASP.NET dan PHP** [20]

| **Pengukuran** | **ASP.NET** | **PHP** |
| --- | --- | --- |
| Biaya | Program ASP perlu IIS untuk dipasang di *plat*form *Windows server*.  Konektivitas *database* mahal; MS-SQL adalah produk Microsoft | Program PHP berjalan di *Apache* pada server *Linux* dan *Unix* secara gratis.  Menggunakan MySQL sebagai *database* (gratis) |
| Kecepatan | ASP.NET adalah bahasa yang dioptimalkan, dikompilasi dan lebih cepat dalam eksekusi | PHP adalah bahasa yang ditafsirkan dan kurang cepat dalam eksekusi |
| Penggunaan Memori | Panjangnya *code path* membuat penggunaan memori lebih mahal | Kecilnya *code path* membuat penggunaan memori lebih efisien |
| *Support and Resources* | Perbaikan dan pembaruan dibuat oleh jumlah yang tersedia dari *developer* Microsoft sendiri.  Kurang dukungan yang tersedia untuk memecahkan tantangan baru. | Lebih banyak pengembang *open source* dan sumber daya yang tersedia dari forum PHP.  Dukungan lebih tersedia dari forum PHP |
| Editor dan perangkat | Paling banyak *Microsoft Visual Studio* yang digunakan untuk membangun aplikasi .NET | Editor independen. Memiliki akses ke editor dalam jumlah yang luas. |
| Pengembangan dan *coding* | Rata-rata waktu pengembangan lebih lama untuk situs yang lebih kecil. | Rata-rata waktu pengembangan lebih singkat untuk situs yang lebih kecil. |

1. ***Unified Modeling Language* (UML)**

UML adalah suatu kumpulan pemodelan yang digunakan untuk menggambarkan sebuah sistem *software* yang terkait dengan objek, yang dijelaskan sebagai berikut [21].

* 1. Objek, merupakan sesuatu yang dapat dilihat, disentuh, atau dirasakan. Sehingga *user* dapatmenyimpan serta melakukan pencatatan perilaku mengenai objek tersebut. Setiap objek memiliki dua karakteristik yaitu:

1. Atribut, merupakan data yang mewakili karakteristik *interest* mengenai sebuah objek.
2. *Behavior,* merupakan kumpulan dari sesuatu yang dapat dilakukan oleh objek dan terkait dengan fungsi-fungsi yang bertindak pada data objek (atribut). Pada siklus berorientasi objek, perilaku objek merujuk kepada metode, operasi, atau fungsi.
   1. Kelas, merupakan suatu set objek yang memiliki atribut dan *behavior* yang sama, biasanya disebut dengan *object class.*
   2. Generalisasi/Spesialisasi, merupakan sebuah teknik dimana atribut dan *behavior* yang umum pada beberapa tipe kelas objek, dikelompokkan (atau diabstraksi) ke dalam kelasnya sendiri, disebut sebagai *supertype.* Atribut dan metode kelas objek *supertype* kemudian diwariskan oleh kelas objek tersebut *(subtype).*
   3. *Inheritance,* merupakan konsep dimana metode dan atau atribut yang ditentukan di dalam sebuah *object class* lainnya.

Menurut penelitian yang dilakukan oleh Nabil Mohammed Ali Munasar yang berjudul *Comparison Between Traditional Approach and Object Oriented Approach in Software Engineering Development* (2011), di dalam UML terdapat beberapa diagram yang digunakan untuk menjelaskan sistem berorientasi objek [22].

1. *Use Case*

*Use case* merupakan deskripsi secara *static* yang menggambarkan bagaimana sistem itu digunakan oleh konsumen atau *user* dan sistem lainnya. Selain itu juga, *use case* diagrammenjelaskan hubungan satu sama lainnya di dalam sistem. Lingkaran pada *use case* mempresentasikan aktivitas sedangkan *person* menggambarkan *user.*

1. *Class* Diagram

*Class* diagrammenggambarkan kelas-kelas yang terdapat pada sistem. Di dalam *class* diagramterdapat kotak yang menggambarkan kelas itu sendiri serta hubungan antar kelas. Di dalam kotak tersebut terdapat *function* yang bisa digunakan dari kelas tersebut.

1. *Sequence* Diagram

*Sequence* diagrammenggambarkan tentang interaksi antara objek pada sistem. *Sequence* digunakan selama desain *subsystem* dan merupakan pemodelan dinamis selama analisa, desain sistem bahkan penangkapan kebutuhan pada sistem.

1. **Pengujian**

Metode Perbandingan Eksponensial (MPE) merupakan salah satu metode untuk menentukan urutan prioritas alternatif keputusan dengan kriteria jamak. Dalam pengerjaannya, metode perbandingan eksponensial memiliki beberapa prosedur antara lain:

1. Menyusun alternatif-alternatif
2. Menentukan kriteria atau perbandingan
3. Menentukan tingkat kepentingan dari setiap kriteria keputusan
4. Melakukan penilaian terhadap semua alternatif pada setiap kriteria
5. Menghitung skor atau nilai total setiap alternatif
6. Menentukan urutan prioritas keputusan.

Adapun rumus matematika yang dipakai dalam Metode Perbandingan Eksponensial adalah:

............(Rumus 2.1)

**Rumus 2. 1 Rumus Metode Perbandingan Eksponensial** [2]

Keterangan:

: Total nilai alternatif ke-i

: Derajat kepentingan relatif kriteria ke-j pada pilihan keputusan i

: Derajat kepentingan kriteria keputusan ke-j; TKKj > 0

: Jumlah kriteria keputusan

: Jumlah pilihan keputusan

: 1,2,3,...m; m : Jumlah kriteria

: 1,2,3,...n; n : Jumlah pilihan kriteria [23]

# **BAB III**

**METODOLOGI PENELITIAN**

1. **Metode Perancangan dan Pengembangan**

Penelitian ini menggunakan metode WDLC [17] yang merupakan gabungan dari metode SDLC dengan metode *prototype.* Metode ini akan diterapkan pada pembangunan sistem informasi *“Lost and Found ”* Universitas Bakrie. Dalam penelitian ini, penggunaan metode pengembangan diterapkan dalam fase-fase sebagai berikut.

* + 1. Pengamatan dan Perencanaan

Pada fase ini dilakukan pengamatan selama dua minggu mengenai masalah yang ada, melakukan wawancara dengan pihak *security* dan beberapa mahasiswa terkait dengan masalah yang ada untuk menganalisa dan melakukan perumusan masalah. Selain melakukan wawancara, pengumpulan indormasi juga dilakukan dengan studi kepustakaan, yaitu dengan *review* buku dan jurnal.

* + 1. Analisa Kebutuhan Aplikasi

Pada tahapan ini akan menggabungkan seluruh informasi yang diperoleh pada tahap sebelumnya, kemudian menganalisis kebutuhan fungsional dan non-fungsional dari sistem tersebut, menganalisis kebutuhan data masukan dan data keluaran. Hasil dari analisis tersebut berupa elisitasi.

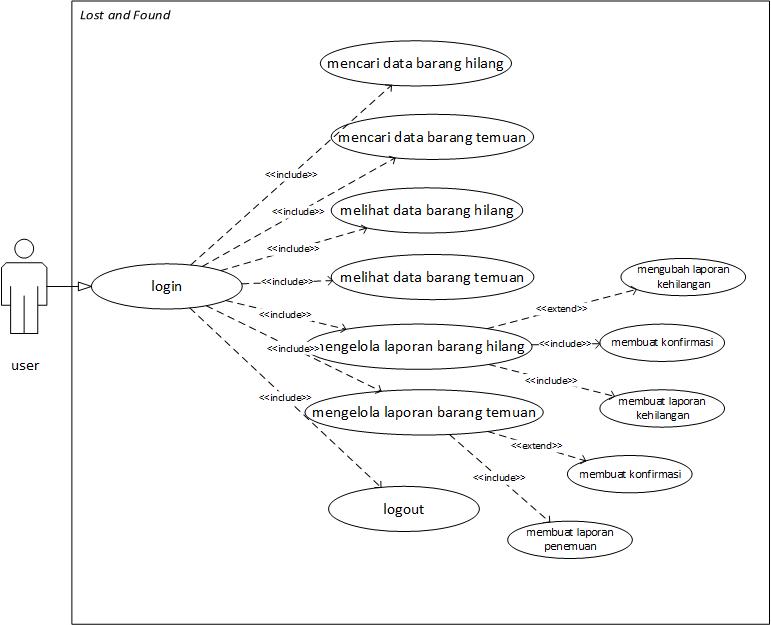
Selain itu, pada tahap ini dilakukan analisis *user requirement.* Terdapat dua *user requirement* yaitu *functional requirement* dan *non-functional requirement.* Keduanya akan menampilkan detail kebutuhan yang diinginkan oleh *user*. Pada tahap ini juga akan menjelaskan secara detail sistem yang akan dibuat dan menentukan informasi yang akan ditampilkan dalam sistem.

* + 1. Perancangan dan Pembangunan

Pada tahap ini akan dilakukan proses perancangan sistem berdasarkan hasil dari tahapan sebelumnya. Hasil dari perancangan sistem ini akan digunakan pada tahap pembangunan.

1. *Use Case Diagram*

*Use case* diagram menggambarkan apa saja fitur yang ada dalam perancangan aplikasi ini. Berikut *use case* diagram dari perancangan aplikasi *Lost and Found .*

**

**Gambar 3. 1 Use case diagram aplikasi Lost and Found**

Deskripsi Gambar 3.1 dari *use case diagram* di atas akan dijelaskan secara lebih detail dalam tabel *use case scenario* berikut:

**Tabel 3. 1Use Case Scenario Login**

|  |  |  |
| --- | --- | --- |
| ***Use case name*** | Login | |
| ***Use case ID*** | 1 | |
| ***Actor*** | *User* | |
| ***Description*** | *Use case* ini menggambarkan kegiatan pada saat *user* melakukan *login* | |
| ***Pre-Condition*** | *User* membuka aplikasi pada *browser* | |
| ***Trigger*** | *Use case* ini dilakukan agar *actor* dapat masuk ke dalam aplikasi. | |
| ***Typical of Event*** | ***Actor Action*** | ***System Response*** |
|  | 1. Membuka aplikasi | 1. Menampilkan halaman untuk *login* ke dalam sistem |
|  | 1. Memasukkan *username* dan *password* | 1. Cek validasi *username* dan *password* |
|  |  | 1. Menampilkan halaman sesuai hak akses yang dimiliki *actor* |
| ***Alternate Course*** | Jika *username* dan *password* salah, maka akan muncul notifikasi lalu Aktor harus melakukan *login* kembali. | |
| ***Post-Condition*** | Aplikasi menampilkan halaman home. | |

**Tabel 3. 2 Use Case Scenario Mencari Data Barang Hilang**

|  |  |  |
| --- | --- | --- |
| ***Use case name*** | Mencari data barang hilang | |
| ***Use case ID*** | 2 | |
| ***Actor*** | *User* | |
| ***Description*** | *Use case* ini menggambarkan kegiatan pada saat *user* melakukan pencarian data barang hilang pada *database* dengan fitur *auto-complete* | |
| ***Pre-Condition*** | *User* membuka aplikasi pada *browser* | |
| ***Trigger*** | *Use case* ini dilakukan agar *actor* dapat mencari barang hilang yang terdapat pada sistem | |
| ***Typical of Event*** | ***Actor Action*** | ***System Response*** |
|  | 1. Membuka aplikasi | 1. Menampilkan halaman untuk *login* ke dalam sistem |
|  | 1. Memasukkan *username* dan *password* | 1. Cek validasi *username* dan *password* |
|  |  | 1. Menampilkan halaman sesuai hak akses yang dimiliki *actor* |
|  | 1. Mengisi *field search* barang hilang | 1. Menampilkan *auto-complete* sesuai dengan *string* yang dicari |
|  | 1. Klik tombol *search* | 1. Menampilkan data sesuai dengan yang dicari |
| ***Alternate Course*** | Jika barang yang dicari tidak ada di *database* maka akan muncul *alert* dan *actor* diharapkan untuk menambah data barang hilang. | |
| ***Post-Condition*** | Aplikasi menampilkan halaman data barang hilang yang dicari | |

**Tabel 3. 3 Use Case Scenario Mencari Data Barang Temuan**

|  |  |  |
| --- | --- | --- |
| ***Use case name*** | Mencari data barang temuan | |
| ***Use case ID*** | 3 | |
| ***Actor*** | *User* | |
| ***Description*** | *Use case* ini menggambarkan kegiatan pada saat *user* melakukan pencarian data barang temuan pada *database* dengan fitur *auto-complete* | |
| ***Pre-Condition*** | *User* membuka aplikasi pada *browser* | |
| ***Trigger*** | *Use case* ini dilakukan agar *actor* dapat mencari barang temuan yang terdapat pada sistem | |
| ***Typical of Event*** | ***Actor Action*** | ***System Response*** |
|  | 1. Membuka aplikasi | 1. Menampilkan halaman untuk *login* ke dalam sistem |
|  | 1. Memasukkan *username* dan *password* | 1. Cek validasi *username* dan *password* |
|  |  | 1. Menampilkan halaman sesuai hak akses yang dimiliki *actor* |
|  | 1. Mengisi *field search* barang hilang | 1. Menampilkan *auto-complete* sesuai dengan *string* yang dicari |
|  | 1. Klik tombol *search* | 1. Menampilkan data sesuai dengan yang dicari |
| ***Alternate Course*** | Jika barang yang dicari tidak ada di *database* maka akan muncul *alert* dan *actor* diharapkan untuk menambah data barang hilang. | |
| ***Post-Condition*** | Aplikasi menampilkan halaman data barang temuan yang dicari. | |

**Tabel 3. 4 Use Case Scenario Melihat Data Barang Hilang**

|  |  |  |
| --- | --- | --- |
| ***Use case name*** | Melihat data barang hilang | |
| ***Use case ID*** | 4 | |
| ***Actor*** | *User* | |
| ***Description*** | *Use case* ini menggambarkan kegiatan untuk melihat data Barang Hilang yang ada dalam bentuk tabel | |
| ***Pre-Condition*** | *User* membuka aplikasi pada *browser* | |
| ***Trigger*** | *Use case* ini dilakukan agar *actor* dapat mencari barang hilang yang terdapat pada sistem | |
| ***Typical of Event*** | ***Actor Action*** | ***System Response*** |
|  | 1. Membuka aplikasi | 1. Menampilkan halaman untuk *login* ke dalam sistem |
|  | 1. Memasukkan *username* dan *password* | 1. Cek validasi *username* dan *password* |
|  |  | 1. Menampilkan halaman sesuai hak akses yang dimiliki *actor* |
|  | 1. Klik menu barang hilang | 1. Menampilkan data barang hilang |
| ***Alternate Course*** | Tidak terdapat *scenario error input* pada tahap ini. | |
| ***Post-Condition*** | Aplikasi menampilkan halaman data barang hilang | |

**Tabel 3. 5 Use Case Scenario Melihat Data Barang Temuan**

|  |  |  |
| --- | --- | --- |
| ***Use case name*** | Melihat data barang temuan | |
| ***Use case ID*** | 5 | |
| ***Actor*** | *User* | |
| ***Description*** | *Use case* ini menggambarkan kegiatan untuk melihat data Barang Temuan yang ada dalam bentuk tabel | |
| ***Pre-Condition*** | *User* membuka aplikasi pada *browser* | |
| ***Trigger*** | *Use case* ini dilakukan agar *actor* dapat mencari barang temuan yang terdapat pada sistem | |
| ***Typical of Event*** | ***Actor Action*** | ***System Response*** |
|  | 1. Membuka aplikasi | 1. Menampilkan halaman untuk *login* ke dalam sistem |
|  | 1. Memasukkan *username* dan *password* | 1. Cek validasi *username* dan *password* |
|  |  | 1. Menampilkan halaman sesuai hak akses yang dimiliki *actor* |
|  | 1. Klik menu barang hilang | 1. Menampilkan data barang hilang |
| ***Alternate Course*** | Tidak terdapat *scenario error input* pada tahap ini. | |
| ***Post-Condition*** | Aplikasi menampilkan halaman data barang temuan. | |

**Tabel 3. 6 Use Case Scenario Mengedit Barang Hilang**

|  |  |  |
| --- | --- | --- |
| ***Use case name*** | Mengelola Barang Hilang | |
| ***Use case ID*** | 6 | |
| ***Actor*** | *User* | |
| ***Description*** | *Use case* ini menggambarkan kegiatan untuk mengelola barang hilang yang telah disimpan dalam *database* | |
| ***Pre-Condition*** | *User* membuka aplikasi pada *browser* | |
| ***Trigger*** | *Use case* ini dilakukan agar *actor* dapat mengelola barang hilang yang terdapat pada sistem | |
| ***Typical of Event*** | ***Actor Action*** | ***System Response*** |
|  | 1. Membuka aplikasi | 1. Menampilkan halaman untuk *login* ke dalam sistem |
|  | 1. Memasukkan *username* dan *password* | 1. Cek validasi *username* dan *password* |
|  |  | 1. Menampilkan halaman sesuai hak akses yang dimiliki *actor* |
|  | 1. Klik menu barang hilang | 1. Menampilkan data barang hilang |
|  | 1. Klik ikon edit | 1. Menampilkan data barang yang akan diedit |
|  | 1. Insert data yang akan di edit kemudian tekan ikon update | 1. Meng-*update* data barang hilang dan menampilkan notifikasi bahwa data telah di *update,* kemudian sistem akan langsung menuju data barang hilang |
| ***Alternate Course*** | Tidak terdapat *scenario error input* pada tahap ini. | |
| ***Post-Condition*** | Aplikasi menampilkan halaman data barang hilang. | |

**Tabel 3. 7 Use Case Scenario Membuat Laporan Kehilangan**

|  |  |  |
| --- | --- | --- |
| ***Use case name*** | Membuat Laporan Kehilangan | |
| ***Use case ID*** | 7 | |
| ***Actor*** | *User* | |
| ***Description*** | *Use case* ini menggambarkan kegiatan untuk membuat laporan kehilangan dan akan disimpan dalam *database* | |
| ***Pre-Condition*** | *User* membuka aplikasi pada *browser* | |
| ***Trigger*** | *Use case* ini dilakukan agar *actor* dapat menambahkan data barang hilang | |
| ***Typical of Event*** | ***Actor Action*** | ***System Response*** |
|  | 1. Membuka aplikasi | 1. Menampilkan halaman untuk *login* ke dalam sistem |
|  | 1. Memasukkan *username* dan *password* | 1. Cek validasi *username* dan *password* |
|  |  | 1. Menampilkan halaman sesuai hak akses yang dimiliki *actor* |
|  | 1. Klik menu barang hilang | 1. Menampilkan data barang hilang |
|  | 1. Klik *button* tambahkan data | 1. Menampilkan formulir data barang dan data pelapor |
|  | 1. Input data barang hilang dan data pelapor dan klik *button* tambah | 1. System akan menginputkan data ke dalam *database* dan akan menampilkan notifikasi bahwa data telah di input |
| ***Alternate Course*** | Apabila ada data barang atau data pelapor yang tidak diisi maka akan muncul notifikasi, dan *user* harus melengkapi data tersebut | |
| ***Post-Condition*** | Aplikasi menampilkan halaman data barang hilang. | |

**Tabel 3. 8 Use Case Scenario Membuat Konfirmasi Barang**

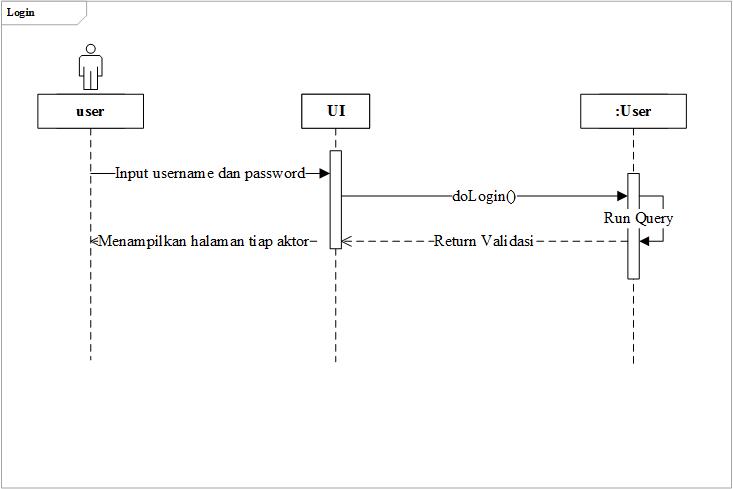
|  |  |  |
| --- | --- | --- |
| ***Use case name*** | Membuat Konfirmasi Barang | |
| ***Use case ID*** | 8 | |
| ***Actor*** | *User* | |
| ***Description*** | *Use case* ini menggambarkan kegiatan untuk membuat laporan kehilangan dan akan disimpan dalam *database* | |
| ***Pre-Condition*** | *User* membuka aplikasi pada *browser* | |
| ***Trigger*** | *Use case* ini dilakukan agar *actor* dapat membuat konfirmasi barang apabila barang telah ditemukan | |
| ***Typical of Event*** | ***Actor Action*** | ***System Response*** |
|  | 1. Membuka aplikasi | 1. Menampilkan halaman untuk *login* ke dalam sistem |
|  | 1. Memasukkan *username* dan *password* | 1. Cek validasi *username* dan *password* |
|  |  | 1. Menampilkan halaman sesuai hak akses yang dimiliki *actor* |
|  | 1. Klik menu barang hilang | 1. Menampilkan data barang hilang |
|  | 1. Klik details barang | 1. Menampilkan details barang dan pelapor |
|  | 1. Klik ikon konfirmasi | 1. Menampilkan formulir konfirnasi barang |
| ***Alternate Course*** | Apabila ada data pelapor yang tidak diisi maka akan muncul notifikasi, dan *user* harus melengkapi data tersebut | |
| ***Post-Condition*** | Aplikasi menampilkan notifikasi barang dan mengirim konfirmasi via email. | |

**Tabel 3. 9 Use Case Scenario Membuat Laporan Penemuan**

|  |  |  |
| --- | --- | --- |
| ***Use case name*** | Membuat Laporan Penemuan | |
| ***Use case ID*** | 9 | |
| ***Actor*** | *User* | |
| ***Description*** | *Use case* ini menggambarkan kegiatan untuk membuat laporan penemuan dan akan disimpan dalam *database* | |
| ***Pre-Condition*** | *User* membuka aplikasi pada *browser* | |
| ***Trigger*** | *Use case* ini dilakukan agar *actor* dapat menambahkan data barang temuan | |
| ***Typical of Event*** | ***Actor Action*** | ***System Response*** |
|  | 1. Membuka aplikasi | 1. Menampilkan halaman untuk *login* ke dalam sistem |
|  | 1. Memasukkan *username* dan *password* | 1. Cek validasi *username* dan *password* |
|  |  | 1. Menampilkan halaman sesuai hak akses yang dimiliki *actor* |
|  | 1. Klik menu barang temuan | 1. Menampilkan data barang temuan |
|  | 1. Klik *button* tambahkan data | 1. Menampilkan formulir data barang dan data pelapor |
|  | 1. Input data barang temuan dan data pelapor dan klik *button* tambah | 1. System akan menginputkan data ke dalam *database* dan akan menampilkan notifikasi bahwa data telah di input |
| ***Alternate Course*** | Apabila ada data barang atau data pelapor yang tidak diisi maka akan muncul notifikasi, dan *user* harus melengkapi data tersebut | |
| ***Post-Condition*** | Aplikasi menampilkan halaman data barang temuan. | |

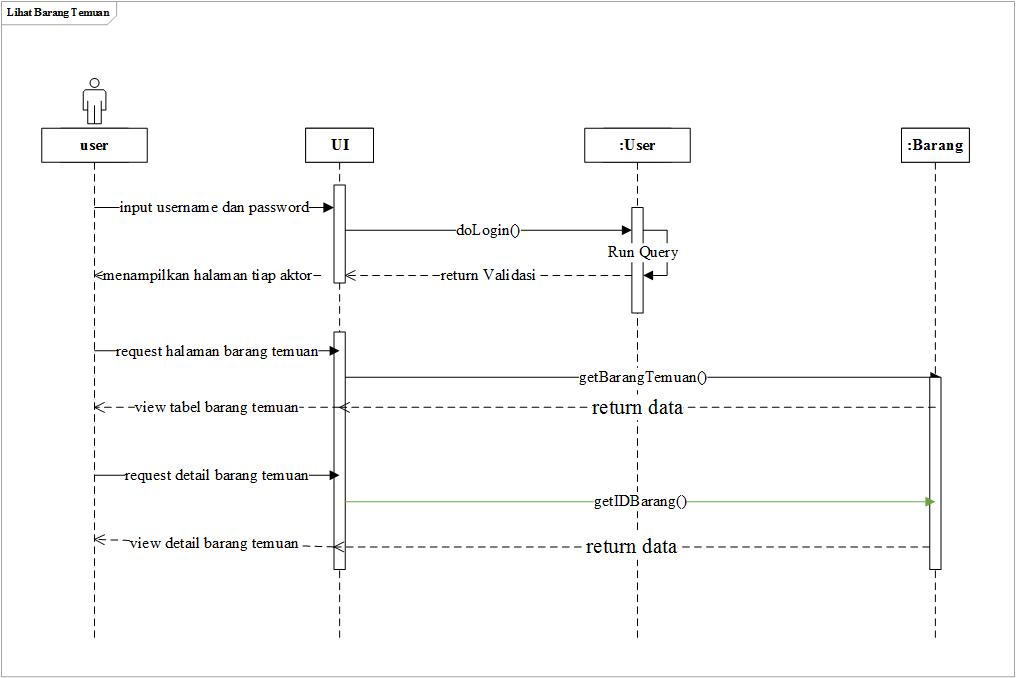
1. *Sequence Diagram*

*Sequence diagram* menggambarkan interaksi antar objek serta mengilustrasikan urutan pesan yang terjadi selama aplikasi *Lost and Found* Universitas Bakrie dijalankan yang terdapat pada *use case scenario.*



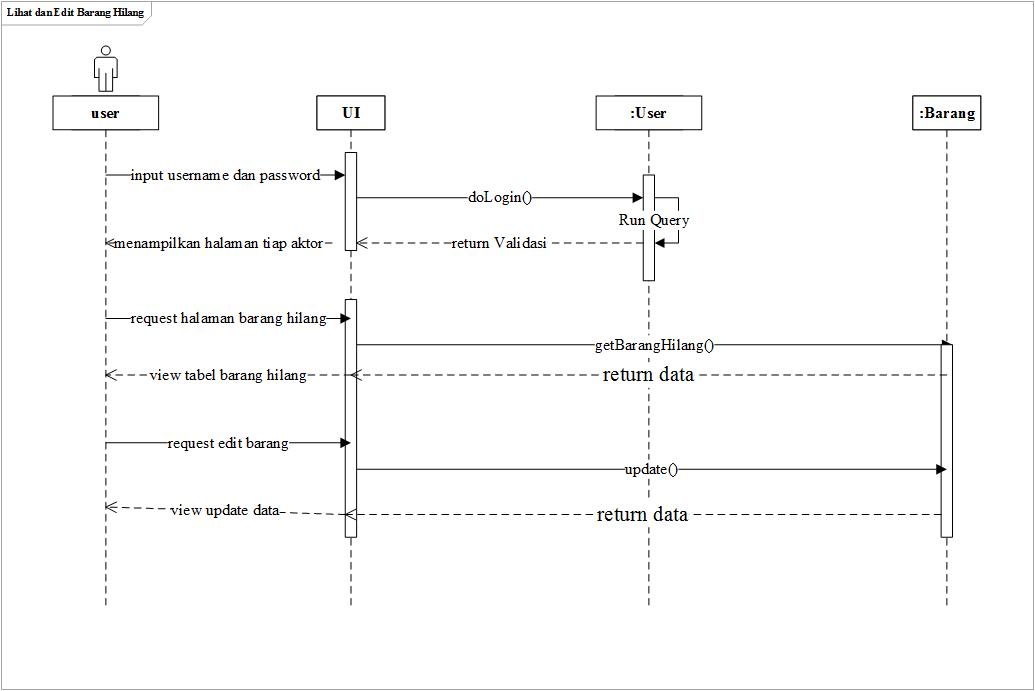
**Gambar 3. 2 Sequence diagram halaman login**

Gambar 3.2 menjelaskan apa saja yang terlibat serta apa saja yang terjadi ketika *user* melihat halaman awal aplikasi. Sistem terlebih dahulu menjalankan UI untuk menampilkan peritah pada *user* dimana *user* dapat memasukkan nomor induk dan *password.* Kemudian sistem akan mengecek ke dalam *database* dengan memanggil fungsi doLogin pada kelas *User* dan melakukan pengecekan validasi is\_loggedin, kemudian mengirimkan hasil validasi ke tampilan untuk diberitahukan kepada pengguna aplikasi.



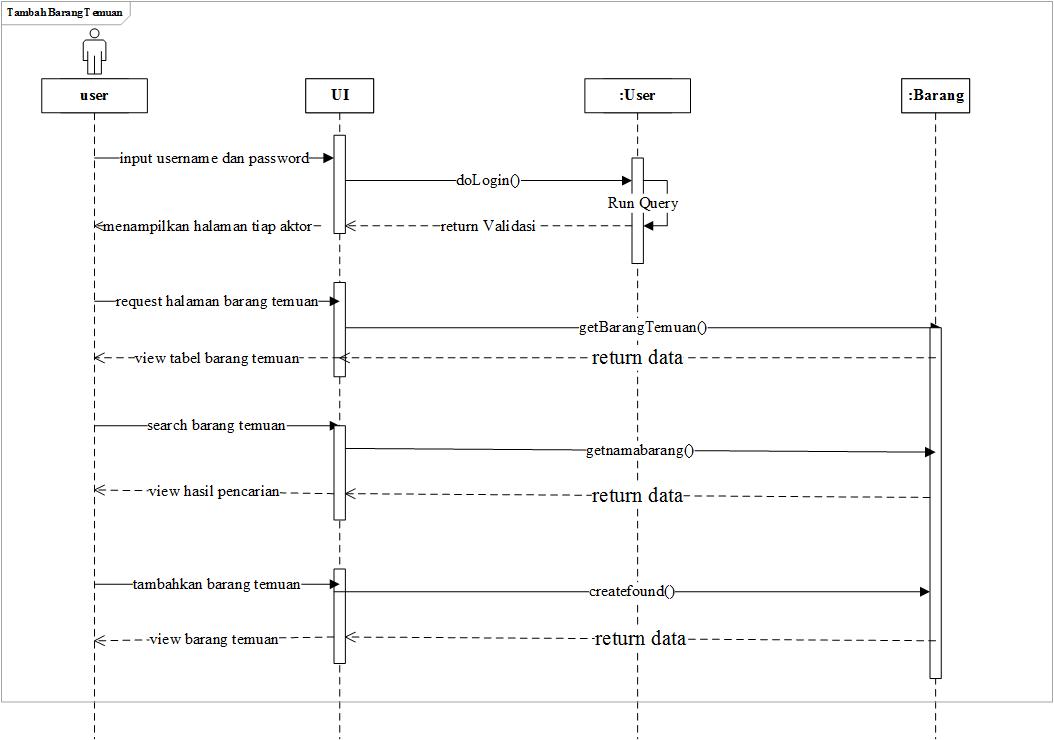
**Gambar 3. 3 Sequence diagram lihat barang temuan**

Gambar 3.3 menjelaskan kelas apa saja yang terlibat serta pesan apa saja yang terjadi ketikas *user* ingin melihat data barang temuan yang disediakan. Ketika *user* menekan menu yang diinginkan maka UI aplikasi akan memberikan menu yang tersedia. Seperti Gambar 3.3 menerangkan proses *request* halaman barang temuan untuk melihat data barang temuan yang ada, setelah itu *user* menekan menu detail barang untuk melihat detail barang yang temuan yang diinginkan.



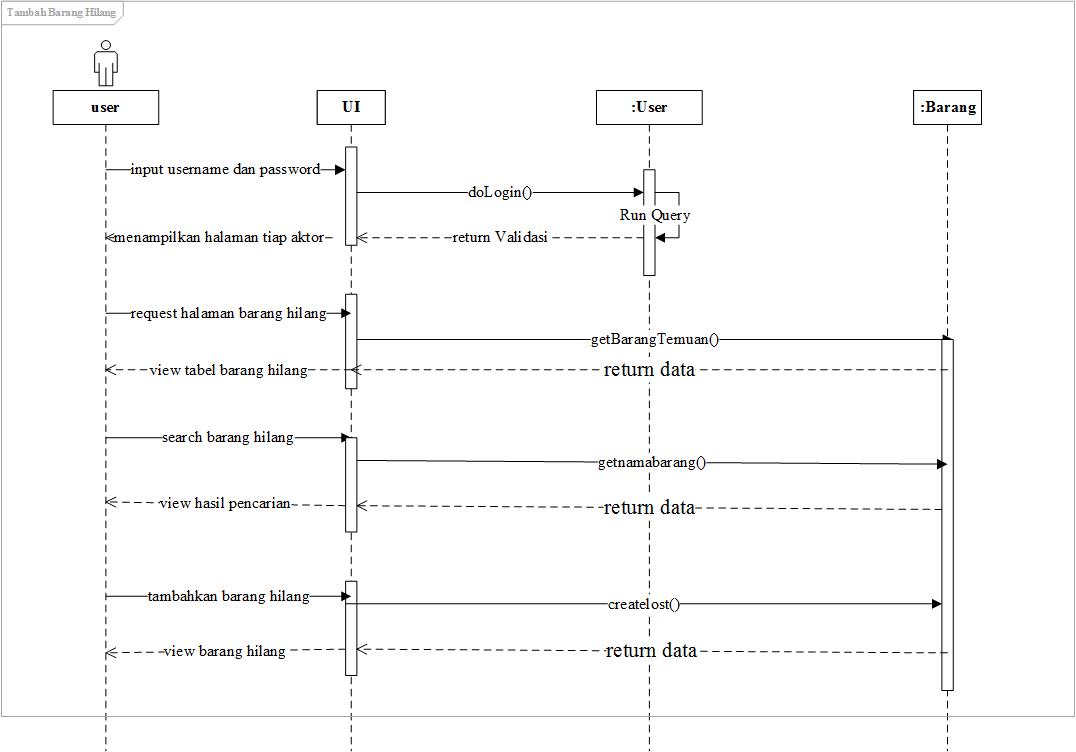
**Gambar 3. 4 Sequence diagram lihat dan edit barang hilang**

Gambar 3.4 menjelaskan proses lihat dan edit data barang hilang yang telah diinputkan oleh *user.* Setelah *user* melakukan *login*, *user* dapat melihat data barang hilang yang telah diinputkan oleh *user* dengan menggunakan fungsi getBarangHilang pada kelas barang, setelah itu *user* dapat melakukan edit masing-masing barang hilang apabila pemilik barang meminta untuk mengubah data barang atau data pemilik yang telah diinput. Dalam proses edit data sistem menggunakan fungsi update pada kelas barang untuk mengedit dan mengupdate data.



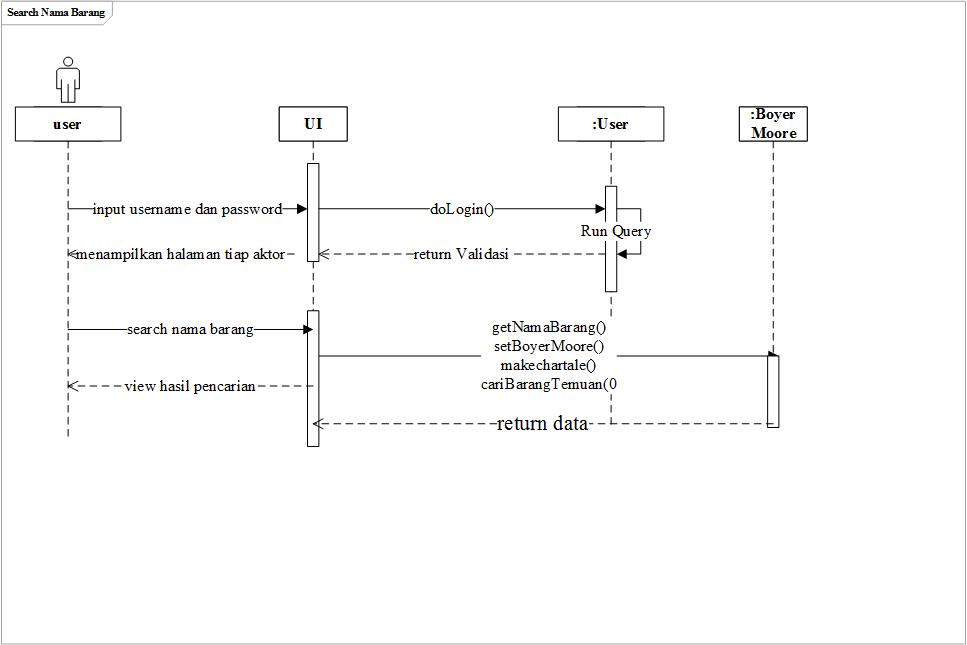
**Gambar 3. 5 Sequence diagram tambah barang temuan**

Gambar 3.5 menjelaskan kelas apa saja yang terlibat serta pesan apa saja yang terjadi ketika *user* melakukan penambahan data pengguna aplikasi. Ketika *user* membuka menu tambah barang temuan, maka sistem akan menampilkan formulir untuk mengisi data lengkap barang temuan dan data pelapor. Setelah *user* melakukan input data secara lengkap, maka sistem akan memanggil fungsi createfound() pada kelas barang dan menyimpan data yang akan dimasukkan ke dalam *database*. Kemudian memberikan pemberitahuan dengan menampilkan notifikasi pada tampilan yang akan langsung me-*refresh* halaman barang temuan.

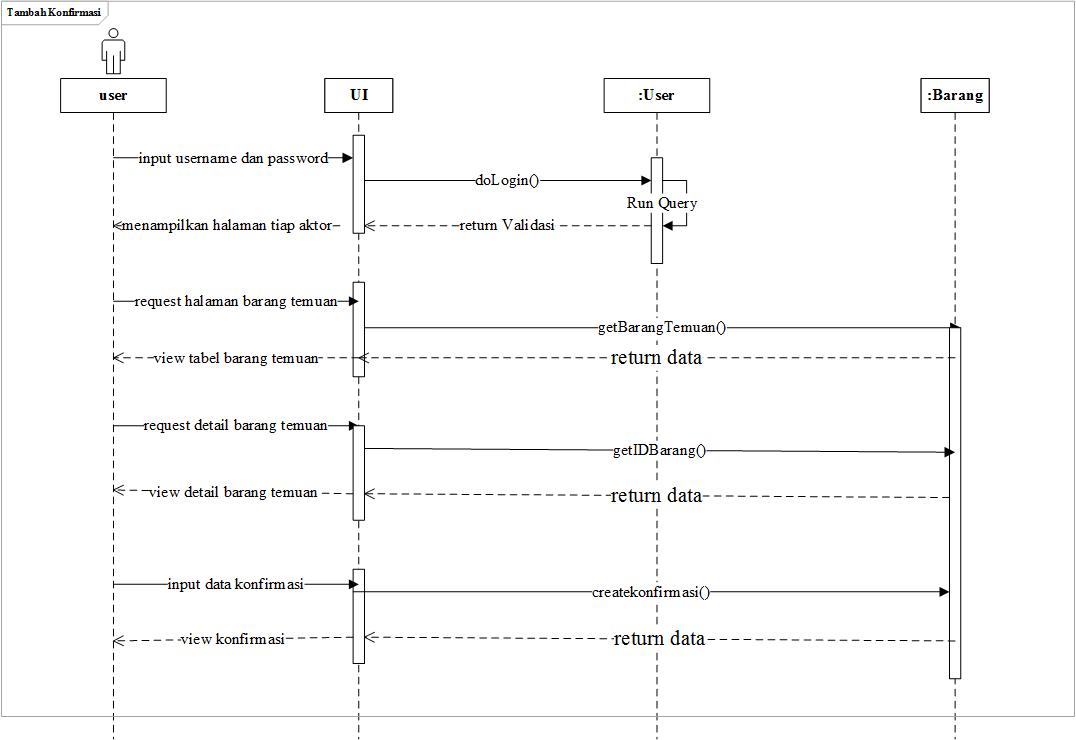


**Gambar 3. 6 Sequence diagram Tambah Barang Hilang**

Gambar 3.6 menjelaskan kelas apa saja yang terlibat serta pesan apa saja yang terjadi ketika *user* melakukan penambahan data pengguna aplikasi. Ketika *user* membuka menu tambah barang hilang, maka sistem akan menampilkan formulir untuk mengisi data lengkap barang hilang dan data pelapor. Setelah *user* melakukan input data secara lengkap, maka sistem akan memanggil fungsi createlost() pada kelas barang dan menyimpan data yang akan dimasukkan ke dalam *database*. Kemudian memberikan pemberitahuan dengan menampilkan notifikasi pada tampilan yang akan langsung me-*refresh* halaman barang hilang.

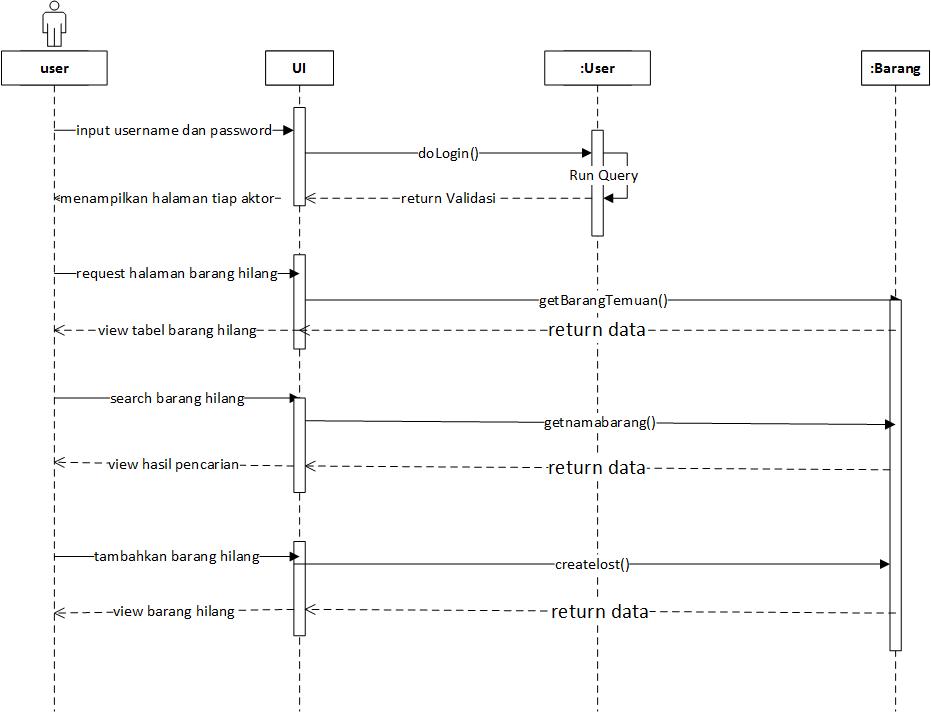


**Gambar 3. 7 Sequence diagram search barang**



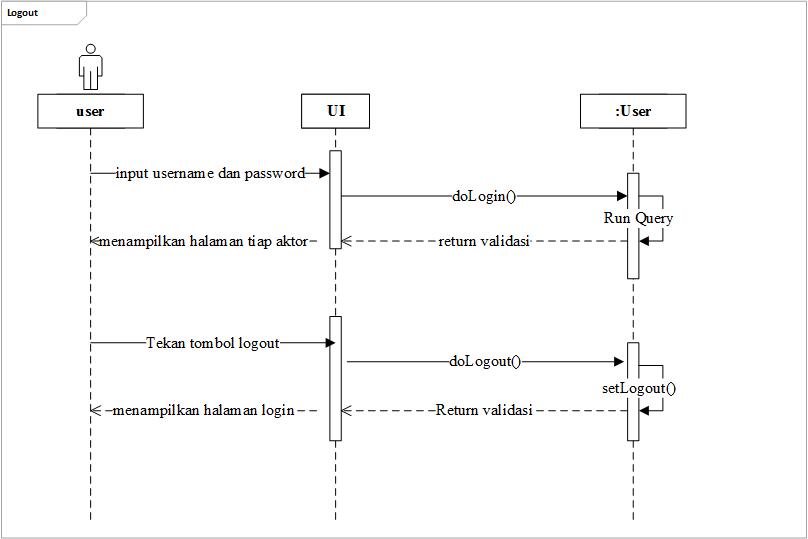
**Gambar 3. 8 Sequence diagram tambah konfirmasi barang**

Gambar 3.8 menjelaskan kelas apa saja yang terlibat serta pesan apa saja yang terjadi ketika *user* melakukan penambahan data konfirmasi barang . Ketika *user* membuka menu konfirmasi barang, maka sistem akan menampilkan formulir untuk mengisi data lengkap konfirmasi barang. Setelah *user* melakukan input data secara lengkap, maka sistem akan memanggil fungsi createkonfirmasi() pada kelas barang dan menyimpan data yang akan dimasukkan ke dalam *database*. Kemudian memberikan pemberitahuan dengan menampilkan notifikasi pada tampilan yang akan langsung me-*refresh* halaman utama. Pada fitur konfirmasi ini, *user* akan meminta pemilik atau pelapor untuk mengecek email karena pemberitahuan telah mengambil barang akan dikirimkan via email.



**Gambar 3. 9 Sequence diagram lihat konfirmasi barang**

Gambar 3.9 menjelaskan kelas apa saja yang terlibat serta pesan apa saja yang terjadi ketika *user* melihat data konfirmasi barang. Ketika *user* membuka menu detail konfirmasi, maka sistem akan menampilkan data konfirmasi barang yang telah dipilih. Data yang telah dipilih ini memanggul fungsi getIDKonfirmasi() yang berisi data konfirmasi yang telah terhubung.

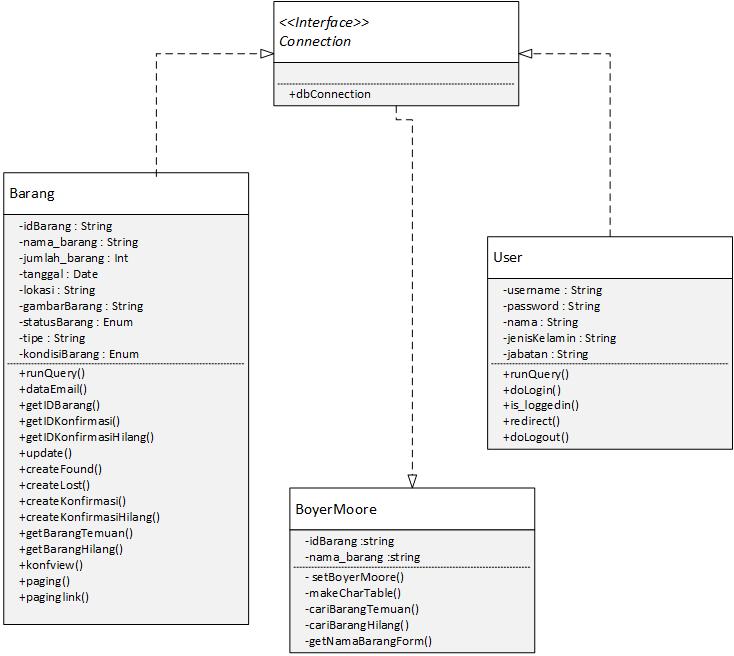


**Gambar 3. 10 Sequence diagram Logout**

Gambar 3.10 menjelaskan aktivitas *logout user.* Ketika perintah *logout* dilakukan, maka sistem akan memanggil fungsi doLogout() pada kelas *User* dan melakukan perintah destroy session pada fungsi setLogout() pada kelas *User*, kemudian mengirimkan hasil validasi ke tampilan untuk diberitahukan kepada pengguna aplikasi dan kembali ke halaman *login* aplikasi.

1. *Class Diagram*

*Class diagram* menggambarkan kelas yang dibuat dengan hubungannya terhadap kelas lainnya. Penentuan kelas dikelompokkan berdasarkan kemiripan *behavior.* Berikut adalah rancangan *class diagram* dari aplikasi *Lost and Found*  Universitas Bakrie yang dibuat.



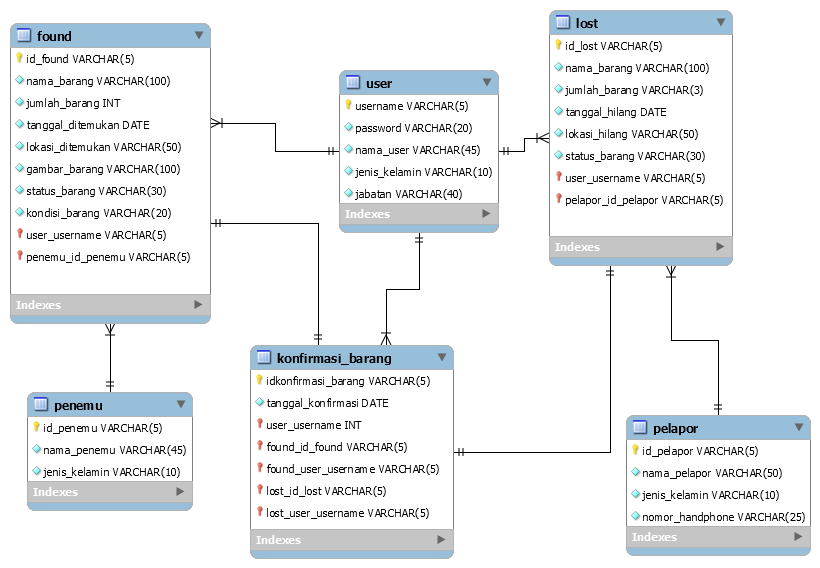
**Gambar 3. 11 Class Diagram Aplikasi Lost and Found**

Pada Gambar 3.11 menjelaskan bahwa terdapat 3 kelas yang akan digunakan pada pengembangan sistem informasi *Lost and Found*  Universitas Bakrie, antara lain:

* *Connection,* merupakan *class interface* yang akan menghubungkan koneksi aplikasi ke *database*. Kelas ini hanya memiliki metode yang bertugas untuk menghubungkan koneksi.
* *User,* merupakan kelas yang mengelola proses data dari pengguna aplikasi. Kelas ini akan mengelola semua tugas yang berhubungan dengan pengguna aplikasi.
* Barang, merupakan kelas yang mengelola proses data yang berhubungan dengan barang temuan dan barang hilang. Kelas ini dikelompokkan berdasarkan *behavior* yang dimiliki yaitu bertugas mengelola data barang hilang dan barang temuan dalam aplikasi.
* Boyer-Moore, merupakan kelas yang menjalankan fungsi-fungsi dari algoritma *boyer-moore.* Kelas ini dikelompokkan berdasarkan *behavior* yang dimiliki yaitu bertugas melakukan pencarian data barang hilang dan barang temuan.

1. *Database design*

Perancangan *database* adalah untuk menentukan isi dan pengaturan data yang dibutuhkan dalam suatu perancangan sistem. Berikut adalah rancangan *database* untuk sistem *Lost and Found*  Universitas Bakrie:



**Gambar 3. 12 Data Model Lost and Found**

* + 1. *Testing*

Proses pengujian dilakukan dengan MPE yang akan menguji efektivitas dari Algoritma *Boyer-Moore*. Adapun tahapan analisis yang akan dilakukan adalah:

1. Menentukan alternatif

Analisa ini menggunakan alternatif Algoritma *Brute Force* sebagai pembanding dari Algoritma *Boyer-Moore.*

1. Menentukan kriteria

Dalam menentukan kriteria yang akan dipakai, dapat dijelaskan pada tabel 3.2

**Tabel 3. 11 Penentuan Kriteria** [2]

|  |  |
| --- | --- |
| **Kriteria** | **Keterangan** |
| Jumlah Iterasi Algoritma | Perhitungan jumlah iterasi/perulangan *(Looping)* yang terjadi pada saat algoritma melakukan usaha pencocokan *string* |
| Jumlah Huruf Pada *pattern* | Jumlah huruf yang dicocokkan oleh algoritma |

1. Menentukan bobot kriteria

Penentuan bobot merupakan salah satu komponen paling penting yang berpengaruh pada hasil analisa.

1. Pemberian nilai pada setiap kriteria

Tahap ini adalah tahap dimana setiap kriteria yang telah terbentuk diberi nilai.

1. Menghitung skor

Setelah nilai pada setiap kriteria dimasukkan, maka tahapan selanjutnya yang akan dilakukan adalah melakukan perhitungan dengan rumus Metode Perbandingan Eksponensial (1).

1. Menentukan prioritas keputusan

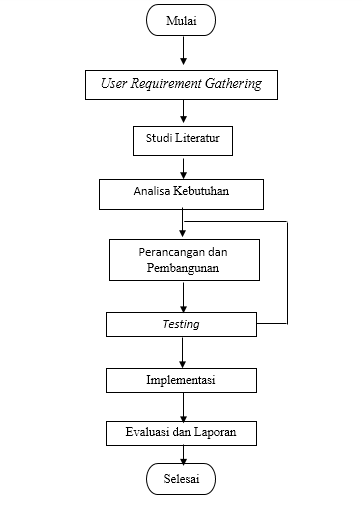
Pada prioritas keputusan akan terlihat total nilai dari alternatif terendah yang memperoleh nilai pertama, karena semakin tinggi total nilai yang diperoleh maka akan semakin tinggi jumlah usaha yang dilakukan oleh algoritma tersebut.

* + 1. Implementasi

Pada tahap terakhir dilakukan pengimplementasian sistem informasi dan melakukan pengujian akhir sistem

1. **Kerangka Penelitian**

Pada penelitian ini memiliki tahapan atau aktivitas yang dilakukan sebagai berikut.



**Gambar 3. 13 Kerangka Penelitian**

1. **Jenis Penelitian**

Jenis penelitian yang dilakukan adalah implementasi algoritma *Boyer-Moore* yang akan diterapkan pada aplikasi sistem *Lost and Found* di Universitas Bakrie berbasis *website.* Rancang bangun yang dilakukan diawali dengan identifikasi kebutuhan dan batasan kebutuhan pengguna, dalam hal ini akan dilakukan pengamatan, wawancara dengan narasumber dan studi literatur.

1. **Objek Penelitian**

Objek penelitian yang dilakukan adalah aplikasi *Lost and Found* yang akan diterapkan di Universitas Bakrie. Aplikasi ini berfungsi untuk memudahkan bagian *security* dalam memproses berita kehilangan dan pencarian hasil barang temuan yang telah disimpan oleh bagian *security.* Aplikasi ini berbasis *web* dengan menerapkan algoritma *Boyer-Moore* pada proses pencarian *string.*

1. **Metode Pengumpulan Data**

Pengumpulan data dan analisa data dalam proses penelitian ini dilakukan untuk melengkapi metode penelitian. Adapun pengumpulan data yang dilakukan adalah sebagai berikut.

1. Pengamatan

Pada proses pengamatan, dilakukan sebagai tahap awal dalam penentuan penelitian yang akan dilakukan. Dalam hal ini melihat proses kerja secara langsung dan membuat hipotesa masalah-masalah yang terjadi pada pegalaman kehilangan barang dan penemuan barang di Universitas Bakrie. Berikut ini merupakan d efinisi lokasi penelitian yang berlangsung:

Nama Institusi : Universitas Bakrie Jakarta

Bidang : Lembaga Pendidikan

Alamat : Jl. HR.Rasuna Said Kav C-22

Gedung Pasar Festival Lt GF/22, Setiabudi, Jakarta Selatan

Telp : 021 5276543

Waktu Wawancara : 12 Februari 2016

Hasil pengamatan yang diperoleh adalah form penemuan barang dan formlaporan kehilangan terlampir pada lampiran.

1. Wawancara

Wawancara dilakukan kepada koordinator *security* Universitas Bakrie yang menangani kehilangan barang maupun temuan barang. Berikut ini merupakan rincian wawancara:

Narasumber : Universitas Bakrie Jakarta

Profesi : Koordinator *Security* Universitas Bakrie

Alamat : Jl. HR.Rasuna Said Kav C-22

Gedung Pasar Festival Lt GF/22, Setiabudi, Jakarta Selatan

Telp : 021 5276543

Waktu Wawancara : 12 Februari 2016

Hasil dari wawancara serta daftar pernyataan terlampir pada lampiran.

1. Studi Literatur

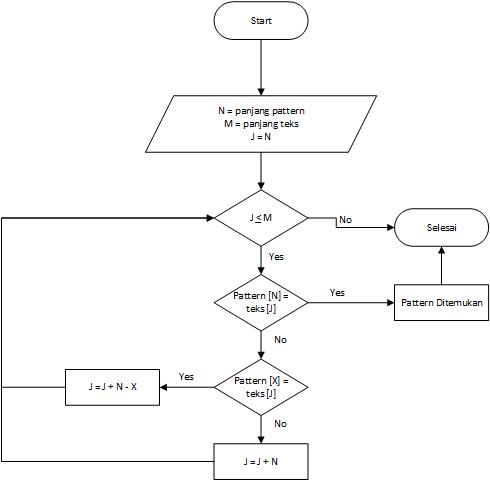
Dalam melakukan penelitian ini, dilakukan kajian pustaka dengan mempelajari beberapa buku teks, jurnal penelitian¸ *e-book,* tugas akhir serta materi-materi di internet yang mendukung dalam proses penelitian. Hasil dari tahap ini merupakan sebuah *literature review* yang dapat digunakan untuk menentukan landasan teori yang tepat untuk penelitian. *Literature Review* juga digunakan untuk pemilihan metode sistem pencarian *string¸* model siklus pengembangan perangkat lunak, bahasa pemrograman, dan pengujian aplikasi dalam penelitian.

1. **Implementasi Algoritma *Boyer-Moore***

Algoritma *Boyer-Moore* dianggap sebagai algoritma pencocokan *string* (*string matching)* yang paling efisien pada kebanyakan aplikasi, sebagai contoh pada *text editor* dan *command substitution.* Hal ini dikarenakan algoritma ini bekerja sangat cepat pada kasus dimana alfabet berukuran sedang dan *pattern* yang akan dicari relatif panjang.

Algoritma *Boyer-Moore* menelusuri karakter-karakter pada *pattern* dari kanan ke kiri, dimulai dari karakter yang berada pada posisi paling kanan. Selama proses pencocokan antara *pattern* P dengan *text* T, ketidakcocokan antara karakter *text* T[i] = c dengan karakter *pattern* P[j] yang bersesuaian ditangani dengan skenario sebagai berikut.

Jika c tidak terdapat pada P, maka geser P secara keseluruhan melewati T[i]. Atau jika tidak – c terdapat dalam P – geser P sampai kemunculan karakter c pada P dapat bersesuaian dengan T[i].



**Gambar 3. 14 Flowchart Algoritma Boyer-Moore** [24]

Teknik ini mampu menghindari perbandingan-perbandingan yang tidak dibutuhkan dengan menggeser *pattern* relatif terhadap *text.*

Setiap karakter c pada alfabet memiliki nilai kemunculan (*last*(c)) sebagai berikut [25].

|  |  |  |
| --- | --- | --- |
| *Last* (c) = | Index (posisi) kemunculan terakhir karakter c pada *pattern*  P | Jika c ada dalam P |
|  |  |  |
|  | -1 | Jika c tidak ada dalam P |

Nilai kemunculan ini menentukan seberapa jauh pergeseran *pattern* P dapat dilakukan jika karakter C di dalam *text* tidak cocok dengan *pattern.* Berikut ini adalah contoh dari nilai kemunculan suatu karakter.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| T: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  | T | G | T | C | A | A | A | G | T | T | C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| P: | 0 | 1 | 2 | 3 | 4 | 5 |
|  | T | G | T | C | A | G |

Maka, didapatkan nilai kemunculan sebagai berikut.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| c | A | C | T | G | S |
| *Last*(c) | 4 | 3 | 2 | 5 | -1 |

Implementasi algoritma *Boyer-Moore* yang diterapkan pada tugas akhir ini, didasarkan pada *pseudocode* berikut.

*Input* : Text dengan n karakter dan *Pattern* dengan m karakter

Output : Index dari sub*string* awal dari T yag cocok dengan P

for x € ∑

*last* [x] 🡨 -1

for y 🡨 m downto 1

*last* [P[y]] 🡨 y

i 🡨 m-1

j 🡨 m-1

Repeat

If P[j] = T[i] then

If j=0 then

return i // we have a match

else

i 🡨 i -1

j 🡨 j -1

else

i 🡨 i + m – Min (j, 1 + *last*[T[i]])

j 🡨 m -1

until i > n -1

Return “no match”

Komputasi dari fungsi *last* itu sendiri membutuhkan waktu O(m + |∑|). Sedangkan pada kasus terbaik dari algoritma tersebut, untuk teks dengan panjang n dan pola yang akan dicari dengan panjang m, dibutuhkan waktu n/m. Hal ini dikarenakan pada kasus terbaik, hanya satu karakter di dalam m yang perlu di cek. Hal ini juga menjelaskan bahwa semakin panjang pola yang akan dicari, maka akan semakin cepat algoritma tersebut menemukannya.

Pada kasus terburuk algoritma ini membutuhkan waktu m\*n untuk dapat menemukan hasil yang cocok. Kasus buruk ini terjadi ketika *string* yang dicari terdiri dari pengulangan sebuah karakter, dan *string* yang menjadi target terdiri dari m-1 pengulangan dari karakter itu yang didahului dengan suatu karakter yang berbeda. Pada skenario ini, harus dilakukan pengecekan sebanyak n-m+1, dimana setiap pengecekan membutuhkan m perbandingan. Oleh karena itu, *running time* algoritma *Boyer-Moore* pada kasus terburuk adalah O(nm + |∑|).

1. **Rencana Kegiatan Penelitian**

Penelitian ini dilakukan dari bulan Februari 2016. Alokasi waktu penelitian yang dibutuhkan untuk melakukan penelitian ini adalah 6 bulan. Adapun rencana kegiatan penelitian terlampir pada lampiran.

# **BAB IV**

**IMPLEMENTASI DAN PENGUJIAN**

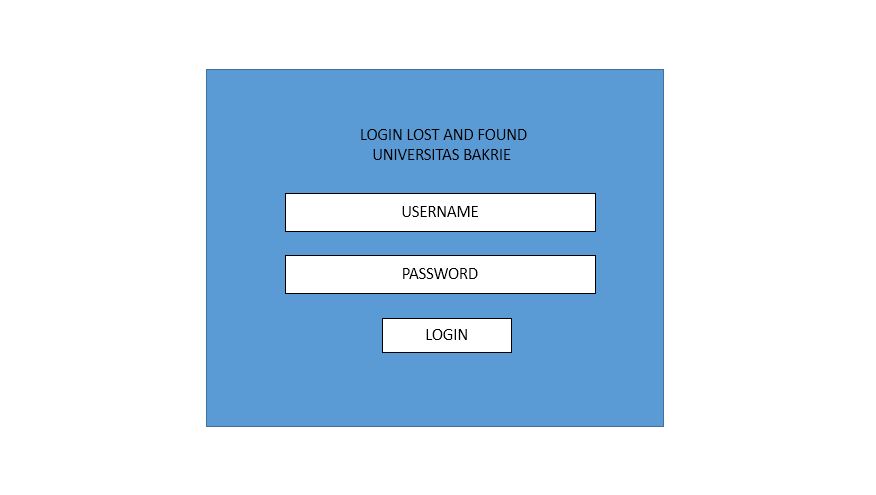
Bab ini membahas mengenai implementasi fitur search dengan *auto-complete* pada *search* barang aplikasi “*Lost and Found* Universitas Bakrie” menggunakan algoritma *Boyer-Moore.* Perancangan dan implementasi algoritma *Boyer-Moore* didasarkan pada analisis yang telah dilakukan pada bab sebelumnya. Bab ini juga akan membahas mengenai pengujian algoritma serta analisis dari hasil penerapan algoritma *Boyer-Moore* pada aplikasi *Lost and Found* Universitas Bakrie.

1. ***Design* *User Interface***

Perancangan aplikasi sangat dibutuhkan dalam sebuah pengembangan software. Adapun model perancangan aplikasi adalah sebagai berikut :

1. *User Interface*

Berikut adalah rancangan *Graphic User Interface* (GUI) dari aplikasi *Lost and Found* yang akan dibuat.



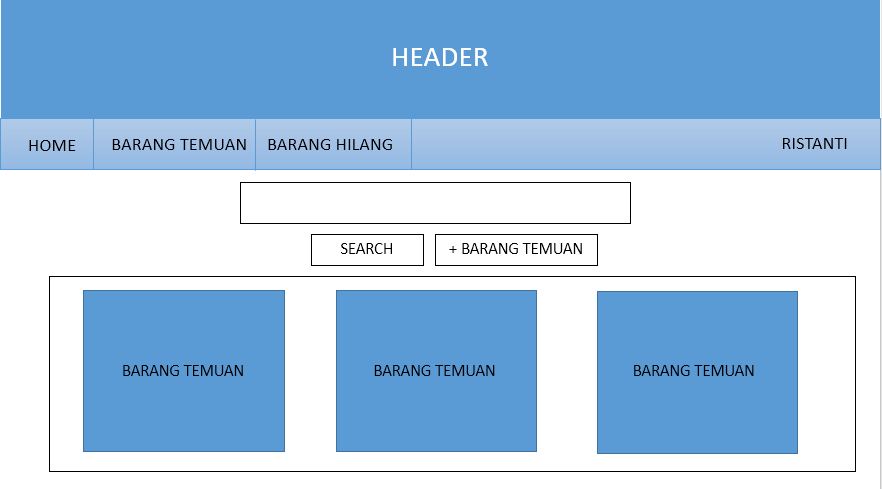
**Gambar 4. 1 GUI Halaman Awal User**

Gambar 4.1 menampilkan halaman ketika *user* membuka aplikasi saat pertama kali dan klik menu ‘LOGIN’. Terdapat beberapa menu navigasi yang dapat dipilih.



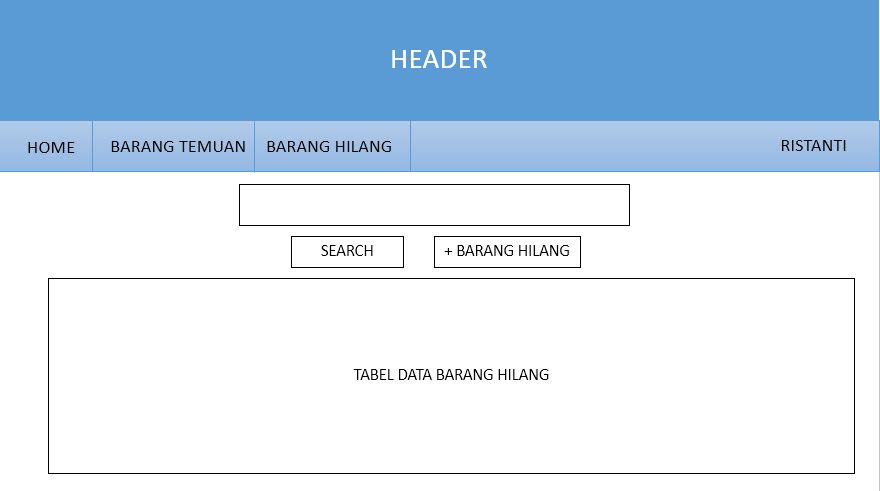
**Gambar 4. 2 Halaman Home**

Gambar 4.2 menggambarkan halaman *home,* ketika *user* sudah melakukan *login*.



**Gambar 4. 3 Halaman Barang Temuan**

Gambar 4.3 menggambarkan halaman awal setelah *user* melakukan ‘klik’ Barang Temuan pada navigasi.



**Gambar 4. 4 Tampilan Barang Hilang**

Gambar 4.4 menggambarkan tampilan barang hilang ketika *user* melakukan ‘klik’ barang hilang pada navigasi.



**Gambar 4. 5 Insert Data Barang**

Gambar 4.5 Menggambarkan pengisian form barang temuan yang kemudian disimpan ke dalam *database*.



**Gambar 4. 6 Konfirmasi Barang**

Gambar 4.6 menggambarkan form konfirmasi barang apabila telah cocok dengan barang yang dilaporkan hilang.

1. **Implementasi Sistem**

Implementasi sistem merupakan kumpulan dari elemen-elemen yang telah dirancang dalam bentuk pemrograman untuk menghasilkan suatu tujuan yang dibuat berdasarkan kebutuhan yang telah dibuat. Berikut adalah spesifikasi *hardware* dan *software* yang digunakan dalam tahapan implementasi.

1. Informasi *Hardware*

Informasi *hardware* yang digunakan dalam pengembangan aplikasi ini adalah sebagai berikut:

Nama *device* : Bisa untuk sebagian besar PC

*Operating system* : Windows 7 Home Premium

*Processor* : Intel(R) Core (TM) i5-3210M CPU @ 2,5GHz 2.50 GHz

*Memory* : 4.00 GB RAM

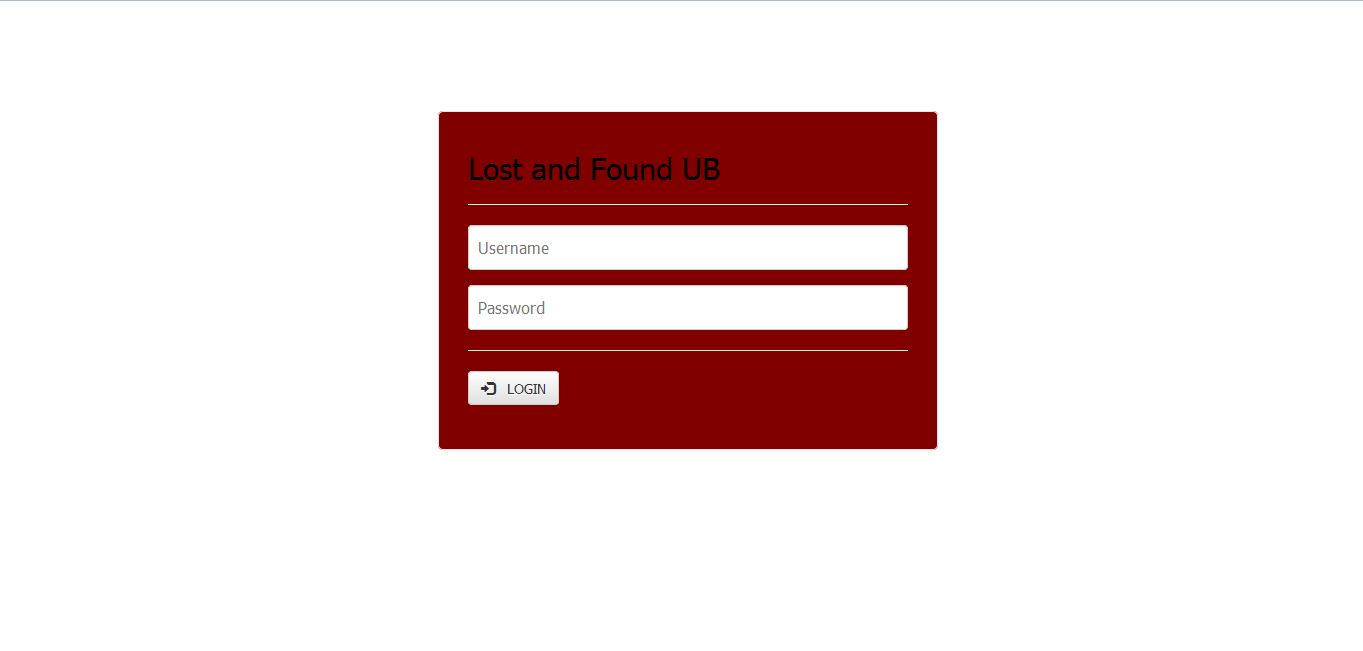
1. Informasi *Software*

Informasi *Software* untuk pengembangan aplikasi adalah sebagai berikut:

* XAMPP *version* 3.2.2 sebagai *web server, database server,* dan *application server*
* Mozilla Firefox *version* 47.0.1 sebagai *web browser*
* Sublime Text Copyright 2006-2014 Sublime HQ Pty Ltd untuk membangun aplikasi

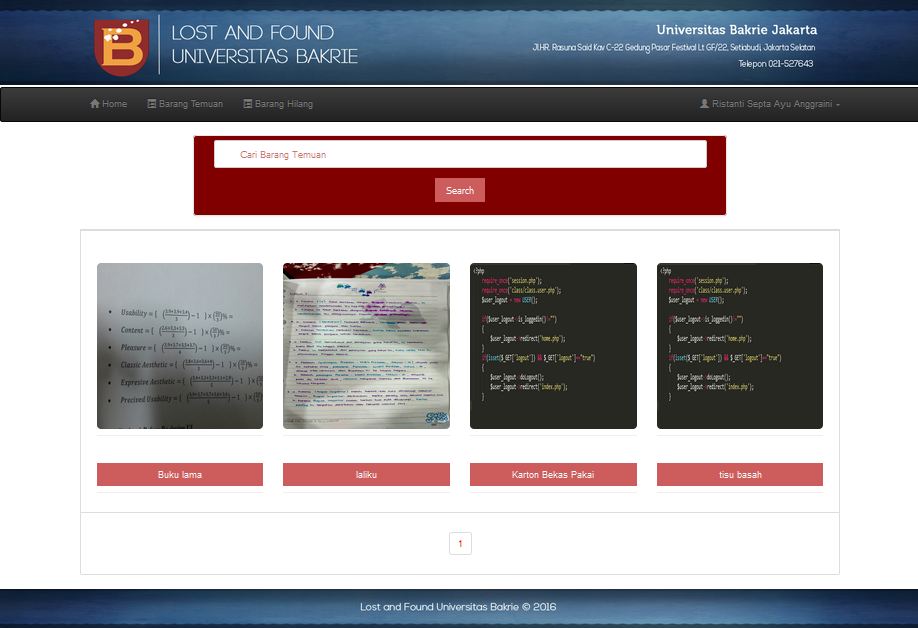
1. **Implementasi Perancangan Antarmuka**

Berdasarkan hasil perancangan *user interface* pada tahap sebelumnya, implementasi dari *user interface* dilakukan dengan menggunakan *front-end framework Hyper Text Markup Language* (HTML) *Cascading Style Sheet* (CSS) yaitu *Bootstrap.* Berikut tampilan hasil rancangan yang telah dibuat:



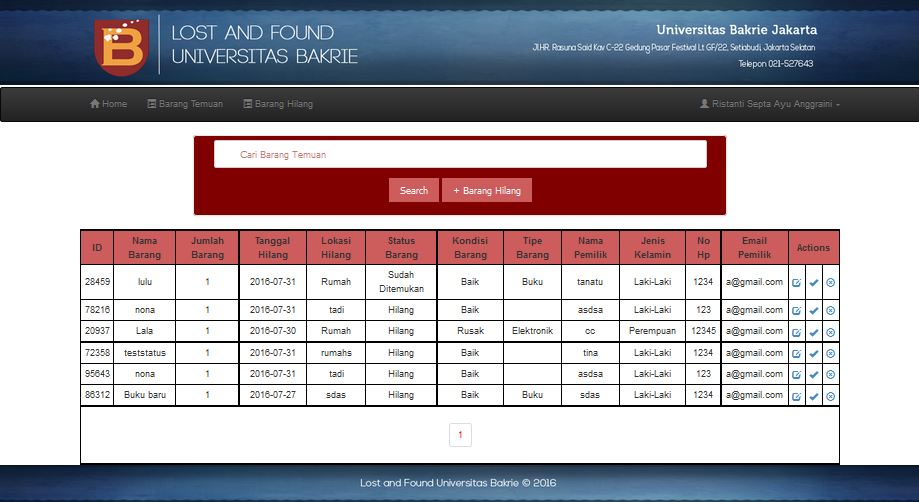
**Gambar 4. 7 Halaman Login**

Gambar 4.7 di atas menggambarkan halaman awal bagi *user* untuk dapat mengakses aplikasi “*Lost and Found* ”. *User* harus memasukkan “nomor induk” dan “*password*” dengan benar sehingga *user* dapat akses masuk ke dalam aplikasi.

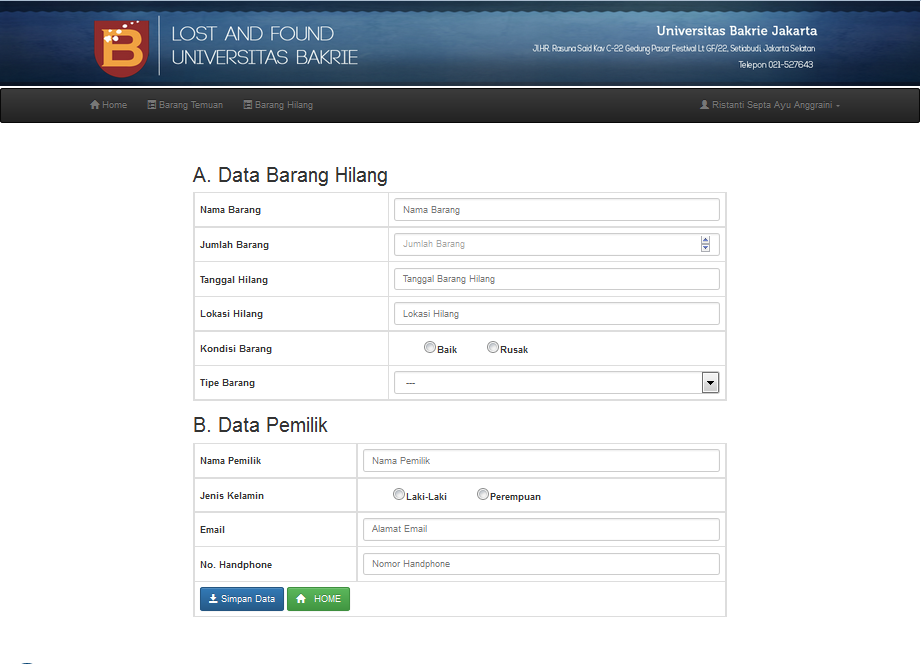


**Gambar 4. 8 Halaman Awal Barang Temuan**

*User* dapat melihat semua data barang yang telah di-*submit* dengan keterangan barang temuan. Gambar 4.8 menampilkan data barang temuan dalam bentuk tabel dan gambar sehingga dapat memudahkan *user* untuk melihat barang yang akan dicari.

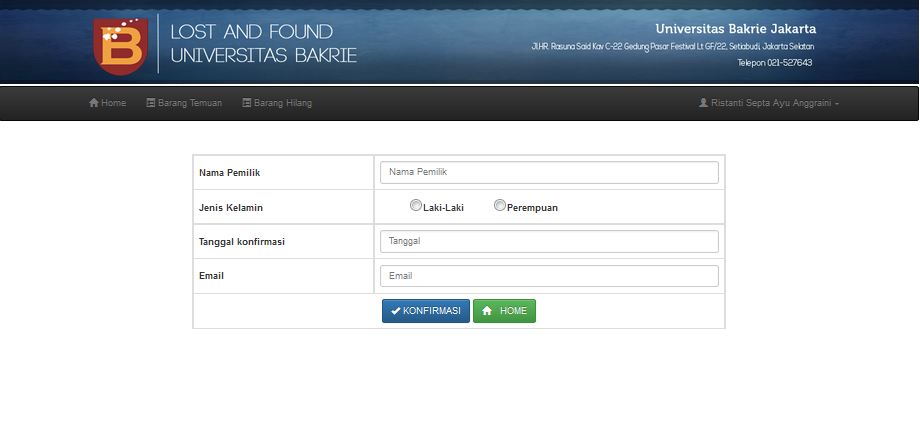
****

**Gambar 4. 9 Tampilan Menu Barang Hilang**

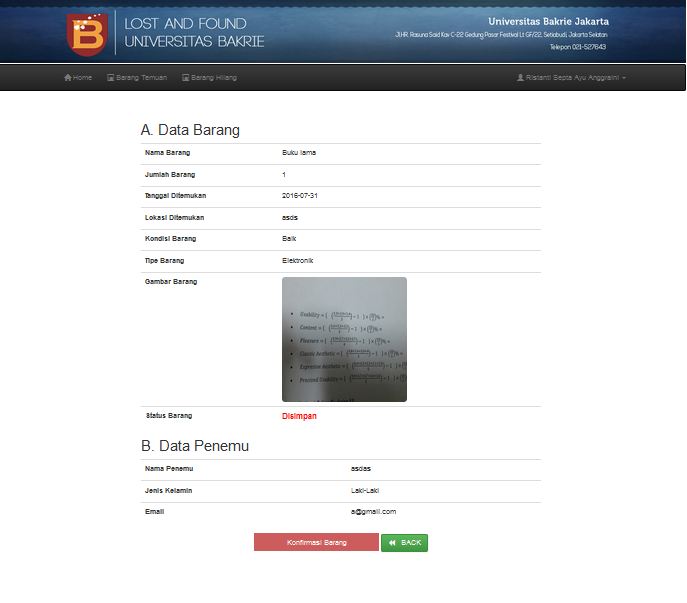


**Gambar 4. 10 Tambahkan data barang temuan atau hilang**

Gambar menampilkan halaman *insert* data barang hilang maupun temuan ke dalam *database*.

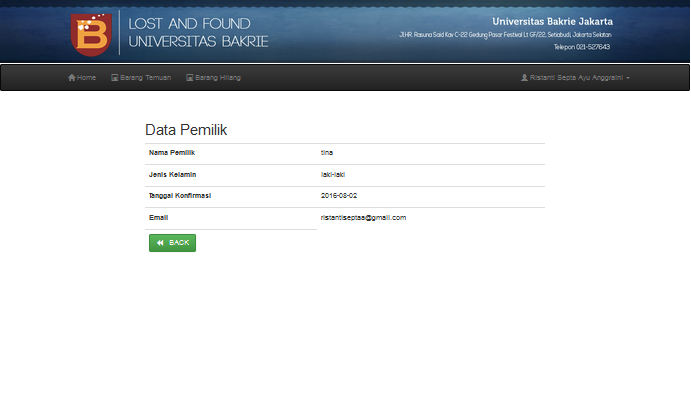
**Gambar 4. 11 Formulir konfirmasi barang**

Gambar 4.11 menampilkan formulir konfirmasi barang yang akan diisi sesuai data *user* yang benar.



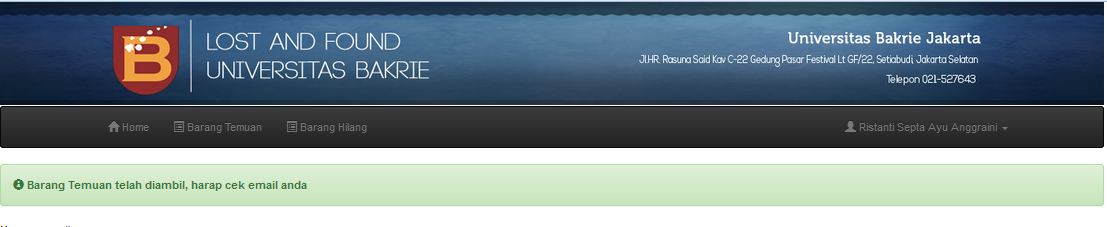
**Gambar 4. 12 Tampilan details barang temuan**

Gambar 4.12 menampilkan data barang yang belum di ambil oleh pemilik, perbedaannya adalah dengan adanya status barang pada details barang tersebut. Status barang yang belum diambil diberikan “Disimpan” sedangkan yang telah diambil adalah “Sudah Diambil”.



**Gambar 4. 13 Tampilan data pemilik**

Gambar 4.13 menampilkan data pemilik yang telah mengambil barang temuan.



**Gambar 4. 14 Notifikasi barang telah diambil**

Gambar 4.14 menampilkan hasil dari konfirmasi barang yang telah diambil dan disarankan pemilik mengecek email yang telah didaftarkan untuk pemberitahuan lebih lengkap.



**Gambar 4. 15 Pesan Konfirmasi barang hilang via email**

Pada gambar 4.15 merupakan contoh email dari bagian *security* untuk konfirmasi bahwa barang telah ditemukan dan dapat diambil di bagian *security* dengan batas waktu yang telah ditentukan.



**Gambar 4. 16 Konfirmasi pengambilan barang via email**

Pada gambar 4.16 merupakan contoh email dari bagian *security* untuk konfirmasi pengambilan barang.

1. **Implementasi Data**

Data yang digunakan dalam penelitian ini adalah data barang temuan yang diambil sampelnya dari salah satu jenis barang. Kumpulan data tersebut dimasukkan dalam *database* mysql menggunakan perangkat bantuan phpmyadmin. *File* data ini bernama “dbpdo.sql”, file ini di-*import* ke dalam *database* “dbpdo”. Data yang digunakan sebagai sampel dalam implementasi *search* ini adalah data dari barang-barang yang sering dilaporkan hilang oleh pihak *security*. Jenis ini dipilih sebagai tes data karena memiliki jumlah informasi terbesar dibandingkan dengan jenis lainnya.

1. **Implementasi Algoritma *Boyer-Moore* pada Fitur *Auto-Complete***

Fitur *auto-complete* digunakan pada pengisian nama barang untuk proses pencarian barang. Dengan adanya fitur *auto-complete,* pengguna tidak perlu memasukkan nama lengkap barang untuk mencari jika barang tersebut memang ada pada basis data yang ada.

Algoritma *Boyer-Moore* berperan sebagai mencari *pattern* (P) yang dicari pada *full-text* (S) nama barang yang ada di tabel tb\_found dan tb\_lost. Dalam pengaplikasiannya, saat pengguna memasukkan tiga karakter pada *search* form, program JavaScript akan melakukan panggilan fungsi PHP dengan Asynchronous Javascript and XML (AJAX) seperti yang digambarkan pada Gambar 4.17. Program ini bergantung pada *library* jQueryuntuk menghasilkan bentuk *auto-complete* yang menerima daftar *string* yang akan ditampilkan dalam format *JavaScript Object Notation* (JSON).

$(function() {

$("#topic\_title").autocomplete({ //Library jQuery autocomplete

source: "autocomplete.php", //Program PHP yang dipanggil

minLength: 2, //berapa huruf minimal untuk mulai operasi

html: true,

open: function(event, ui) {

$(".ui-autocomplete").css("z-index", 1000); //CSS

}

});

**Gambar 4. 17 Program JavaScript untuk Fitur Auto-Complete**

Program JavaScript paga Gambar 4.18 menggunakan metode GET untuk melakukan *request* ke program PHP pada *server side.* Program PHP autocomplete.php akan menjalankan panggilan fungsi untuk menjalankan fungsi algoritma *Boyer-Moore* yang ada pada program boyermoore.php. Gambar 4.18 memberikan gambaran mengenai kode pada autocomplete.php

$keyword = $\_GET['term'];

//$data = searchForKeyword($keyword);

$data = cari($keyword); //ada di *database*.php

//$data->array semua barang yang ditemukan

echo json\_encode($data);

**Gambar 4. 18 Kode autocomplete.php**

Program boyermoore.php bertugas untuk mengambil data nama barang dari *database* dan mengolahnya dengan algoritma *Boyer-Moore* untuk menemukan *pattern* (P) tertentu sesuai yang diinginkan oleh pengguna. Gambar 4.19 menggambarkan bagaimana program mengakses *database* dan mengubahnya menjadi *string* yang memiliki pola yang ditunjukkan pada Gambar 4.20.

function cari($key) {

$crud = new crud();

$hasil = '';

$table = 'tb\_found';

$res= $crud->getNamaBarangFrom($table);

$size = sizeof($res);

for ($i = 0; $i < $size; $i++) {

foreach ($res[$i] as $a => $b) {

$hasil .= ($b . "\*");

}

}

$cari = BoyerMoore($hasil, $key);

return $cari;

}

**Gambar 4. 19 Fungsi cari() dalam boyermoore.php**

Nama barang 1\*nama barang 2\* nama barang 3\*...\*

**Gambar 4. 20 Bentuk String yang akan diolah**

Sebelum melakukan pengolahan dengan *Boyer-Moore*, program PHP pada boyermoore.php khususnya fungsi makechartable() membuat sebuah tabel dalam bentuk array dari *pattern* (P) dengan pola array(2) { [‘karakter pertama}=> int(i) [karakter kedua]=> int(0) ... }. Gambar 4.19 menunjukkan potongan kode pada fungsi makechartable().

function makeCharTable($string) {

$len = strlen($string);

$table = array();

for ($i=0; $i < $len; $i++) {

$table[strtolower($string[$i])] = $len - $i - 1;

}

return $table;

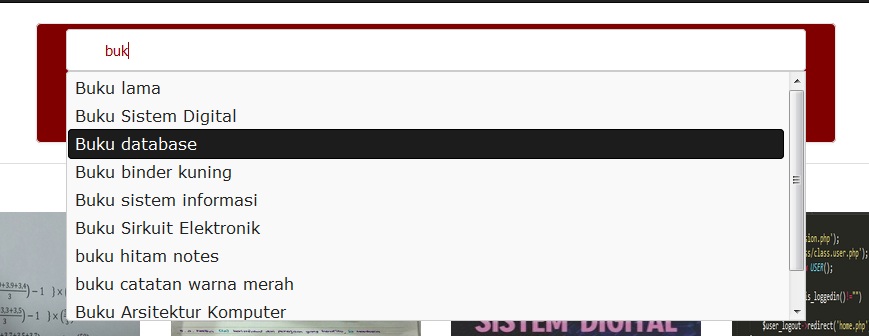
}

**Gambar 4. 21 Fungsi makechartable()**

Fungsi paling krusial yang berperan sebagai pencari *pattern* (P) pada *string* adalah fungsi Boyermoore($text, $*pattern*) yang ditunjukkan pada Lampiran. Selain menggunakan *Boyer-Moore* untuk menentukan dimana indeks *string* saat *pattern* ditemukan, fungsi ini juga menggandakan seluruh nama barang sehingga walaupun *pattern* ditemukan di tengah nama barang, seperti menemukan *pattern* “ge” dalam “Buku Management, nama lengkap dari barang akan diambil dan hasilnya dapat dilihat dalam fitu *auto-complete.*

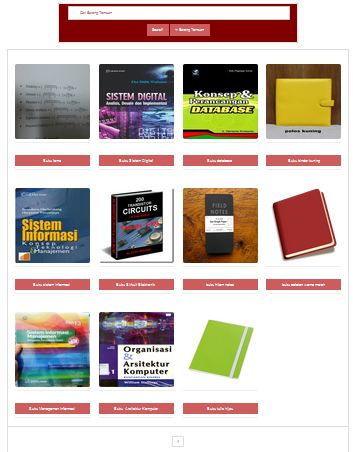
1. **Hasil Pencarin Berdasarkan Kata Kunci**

Bagian ini menjelaskan mengenai hasil pencarian *string* yang dilakukan *user* pada *field search.* Kata kunci yang dimasukkan akan ditampilkan dalam beberapa kondisi. Kondisi pertama adalah hasil pencarian yang terdapat pada *database* dan ditampilkan oleh fitur *auto-complete.* Hasil yang ditampilkan merupakan semua data yang ada di *database* dan merupakan *string* dari yang telah di-*input-*kan. Kondisi kedua adalah hasil pencarian yang tidak terdapat pada *database* dan tidak dapat ditampilkan oleh fitur *auto-complete* karena tidak sesuai dengan *string* yang ada di *database.*



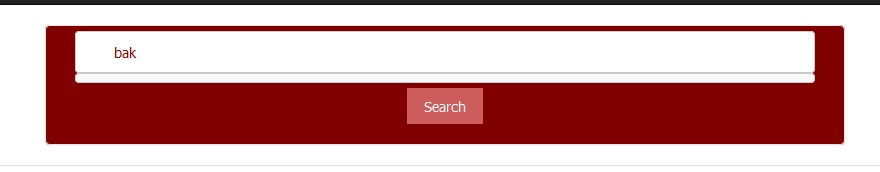
**Gambar 4. 22 Search dengan fitur auto-complete kondisi pertama**

Pada Gambar 4. 22 merupakan hasil pencarian data nama barang dengan menggunakan string “buk” dan *auto-complete* dapat menampilkan semua hasil dari *string* “buk” yang ada di *database.*



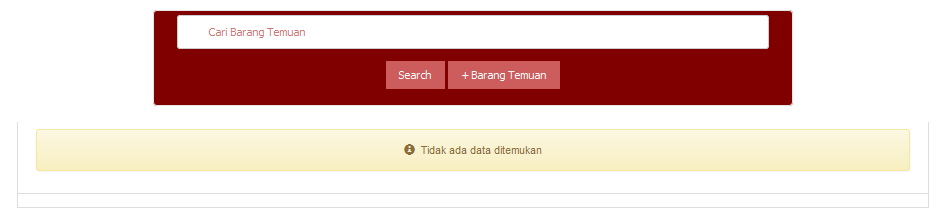
**Gambar 4. 23 Hasil search pada kondisi pertama**

Pada Gambar 4.23 menampilkan hasil dari pencarian kata “buk” pada kondisi pertama, terdapat 10 data barang yang tersimpan dengan kata yang mengandung *string* “buk”.



**Gambar 4. 24 Search dengan fitur autocomple kondisi kedua**

Pada Gambar 4. 24 merupakan hasil pencarian data nama barang dengan menggunakan string “bak” dan *auto-complete* tidakdapat menampilkan semua hasil dari *string* “bak” yang dari *database.*



**Gambar 4. 25 Hasil dari pencarian string kondisi kedua**

Pada Gambar 4.25 menampilkan hasil dari pencarian kata yang mengandung “bak” dan sistem tidak dapat menemukan nama barang yang cocok dengan yang di-*input*-kan pada kolom *search.*

1. **Pengujian Algoritma**

Berdasarkan rencana pengujian algoritma bagian 3.1.5 di atas, pengujian akan dilakukan dengan memakai Metode Perbandingan Eksponensial. Penulis mengadopsi *draft* pengujian mengacu pada penelitian [26] yang menerapkan Metode Perbandingan Eksponensial sebagai metode pengujian penentuang ranking. Berikut adalah penjelasan pengujian algoritma *Boyer-Moore* berdasarkan MPE :

* 1. Menentukan *Pattern* pada Text

Algoritma *Boyer-Moore* merupakan algoritma pencocokan *string* yang terdiri dari dua komponen yaitu *pattern* dan teks. Dalam penentuan *pattern* dan teks untuk analisa data diambil dari analisa pada tabel 4.1.

**Tabel 4. 1 Penentuan Pattern dan Teks setelah jumlah hurufnya disamakan**

| **Proses Ke-** | ***Pattern*** | **Teks Setelah Dipotong** | **Teks di *Database*** |
| --- | --- | --- | --- |
| 1 | h | h | Handphone Samsung |
| h | Headset Putih |
| h | Buku Hitam Notes |
| h | Charger Iphone |
| h | Hardisk |
| h | Buku Tulis Hijau |
| b | Buku Arsitektur Komputer |
| d | Dompet Kulit |
| b | Buku Sistem Digital |
| b | Buku *Database* |
| 2 | ha | ha | Handphone Samsung |
| ho | Handphone Samsung |
| he | Headset Putih |
| hi | Buku Hitam Notes |
| ha | Charger Iphone |
| ha | Hardisk |
| hi | Buku Tulis Hijau |
| bu | Buku Arsitektur Komputer |
| do | Dompet Kulit |
| bu | Buku Sistem Digital |
| bu | Buku *Database* |
| 3 | har | han | Handphone Samsung |
| hon | Handphone Samsung |
| hea | Headset Putih |
| hit | Buku Hitam Notes |
| har | Charger Iphone |
| har | Hardisk |
| hij | Buku Tulis Hijau |
| buk | Buku Arsitektur Komputer |
| dom | Dompet Kulit |
| buk | Buku Sistem Digital |
| buk | Buku *Database* |
| 4 | hard | hand | Handphone Samsung |
| head | Handphone Samsung |
| hita | Headset Putih |
| harg | Buku Hitam Notes |
| hone | Charger Iphone |
| hard | Hardisk |
| hija | Buku Tulis Hijau |
| buku | Buku Arsitektur Komputer |
| domp | Dompet Kulit |
| buku | Buku Sistem Digital |
| buku | Buku *Database* |

* 1. Proses Pencarian Algoritma

Setelah *pattern* dan teks terbentuk maka proses selanjutnya adalah pencocokan karakter. Dalam proses pengujian ini, digunakan sebuah pembanding algoritma yaitu algoritma *brute-force* yang melakukan pencarian dari *string* paling kiri ke kanan. Berikut adalah tabel yang menggambarkan ilustrasi pencocokan karakter yang dilakukan oleh Algoritma Brute Force.

**Tabel 4. 2 Simulasi Cara Kerja Algoritma Brute Force**

| **Proses ke-** | **Iterasi Ke-** | ***Pattern*** | **Teks** | **Teks di *Database*** | **Pencocokan Brute Force** | **Hasil Sugesti** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | h | h | Handphone Samsung | h = h | Handphone Samsung |
| 2 | h | Headset Putih | h = h | Headset Putih |
| 3 | h | Buku Hitam Notes | h = h | Buku Hitam Notes |
| 4 | h | Charger Iphone | h = h | Charger Iphone |
| 5 | h | Hardisk | h = h | Hardisk |
| 6 | h | Buku Tulis Hijau | h = h | Buku Tulis Hijau |
| 7 | b | Buku Arsitektur Komputer | h ≠ b | - |
| 8 | d | Dompet Kulit | h ≠ d | - |
| 9 | b | Buku Sistem Digital | h ≠ b | - |
| 10 | b | Buku *Database* | h ≠ b | - |
| 2 | 1 | ha | ha | Handphone Samsung | h = h | - |
| 2 | a = a | Handphone Samsung |
| 3 | ho | Handphone Samsung | h = h | - |
| 4 | a ≠ o | - |
| 5 | he | Headset Putih | h = h | - |
| 6 | h ≠ e | - |
| 7 | hi | Buku Hitam Notes | h = h | - |
| 8 | h ≠ h |  |
| 9 | ha | Charger Iphone | h = h | - |
| 10 | a = a | Charger Iphone |
| 11 | ha | Hardisk | h = h | - |
| 12 | a = a | Hardisk |
| 13 | hi | Buku Tulis Hijau | h = h | - |
| 14 | a ≠ i | - |
| 15 | bu | Buku Arsitektur Komputer | h ≠ b | - |
| 16 | do | Dompet Kulit | h ≠ b | - |
| 17 | bu | Buku Sistem Digital | h ≠ b | - |
| 18 | bu | Buku *Database* | h ≠ b | - |
| 3 | 1 | har | han | Handphone Samsung | h = h | - |
| 2 | a = a |  |
| 3 | r ≠ n |  |
| 4 | hon | Handphone Samsung | h = h | - |
| 5 | a ≠ o |  |
| 6 | hea | Headset Putih | h = h | - |
| 7 | a ≠ e |  |
| 8 | hit | Buku Hitam Notes | h = h | - |
| 9 | a ≠ i |  |
| 10 | har | Charger Iphone | h = h | - |
| 11 | a= a | - |
| 12 | r = r | Charger Iphone |
| 13 | har | Hardisk | h = h | - |
| 14 | a = a | - |
| 15 | r = r | Hardisk |
| 16 | hij | Buku Tulis Hijau | h = h | - |
| 17 | a ≠ i |  |
| 18 | r ≠ j |  |
| 19 | buk | Buku Arsitektur Komputer | h ≠ b | - |
| 20 | dom | Dompet Kulit | h ≠ d | - |
| 21 | buk | Buku Sistem Digital | h ≠ b | - |
| 22 | buk | Buku *Database* | h ≠ b | - |
| 4 | 1 | hard | hand | Handphone Samsung | h = h | - |
| 2 | a = a | - |
| 3 | r ≠ n | - |
| 4 | head | Headset Putih | h = h | - |
| 5 | a ≠ e | - |
| 6 | hita | Buku Hitam Notes | h = h | - |
| 7 | a ≠ i | - |
| 8 | harg | Charger Iphone | h = h | - |
| 9 | a = a | - |
| 10 | r = r | - |
| 11 | d ≠ g | - |
| 12 | hone | Charger Iphone | h = h | - |
| 13 | a ≠ o | - |
| 14 | hard | Hardisk | h = h | - |
| 15 | a = a | - |
| 16 | r = r | - |
| 17 | d = d | Hardisk |
| 18 | hija | Buku Tulis Hijau | h = h | - |
| 19 | a ≠ i | - |
| 20 | buku | Buku Arsitektur Komputer | h ≠ b | - |
| 21 | domp | Dompet Kulit | h ≠ d | - |
| 22 | buku | Buku Sistem Digital | h ≠ b | - |
| 23 | buku | Buku *Database* | h≠ b | - |

**Tabel 4. 3 Simulasi Cara Kerja Algoritma Boyer-Moore**

| **Proses ke** | **Iterasi Ke-** | ***Pattern*** | **Teks** | **Teks di *Database*** | **Pencocokan *Boyer-Moore*** | **Hasil Sugesti** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | h | h | Handphone Samsung | h = h | Handphone Samsung |
| 2 | h | Headset Putih | h = h | Headset Putih |
| 3 | h | Buku Hitam Notes | h = h | Buku Hitam Notes |
| 4 | h | Charger Iphone | h = h | Charger Iphone |
| 5 | h | Hardisk | h = h | Hardisk |
| 6 | h | Buku Tulis Hijau | h = h | Buku Tulis Hijau |
| 7 | b | Buku Arsitektur Komputer | h ≠ b | - |
| 8 | d | Dompet Kulit | h ≠ d | - |
| 9 | b | Buku Sistem Digital | h ≠ b | - |
| 10 | b | Buku *Database* | h ≠ b | - |
| 2 | 1 | ha | ha | Handphone Samsung | a = a | - |
| 2 | h = h | Handphone Samsung |
| 3 | ho | Handphone Samsung | a ≠ o | - |
| 4 | he | Headset Putih | a ≠ e | - |
| 5 | hi | Buku Hitam Notes | a ≠ i | - |
| 6 | ha | Charger Iphone | a = a | - |
| 7 | h = h | Charger Iphone |
| 8 | ha | Hardisk | a = a | - |
| 9 | h = h | Hardisk |
| 10 | hi | Buku Tulis Hijau | a ≠ i | - |
| 11 | bu | Buku Arsitektur Komputer | a ≠ u | - |
| 12 | do | Dompet Kulit | a ≠ o | - |
| 13 | bu | Buku Sistem Digital | a ≠ u | - |
| 14 | bu | Buku *Database* | a ≠ u | - |
| 3 | 1 | har | han | Handphone Samsung | r ≠ n | - |
| 2 | hon | Handphone Samsung | r ≠ n | - |
| 3 | hea | Headset Putih | r ≠ a | - |
| 4 | hit | Buku Hitam Notes | r ≠ t | - |
| 5 | har | Charger Iphone | r = r | - |
| 6 | a = a | - |
| 7 | h = h | Charger Iphone |
| 8 | har | Hardisk | r = r | - |
| 9 | a = a | - |
| 10 | h = h | Hardisk |
| 11 | hij | Buku Tulis Hijau | j ≠ r | - |
| 12 | buk | Buku Arsitektur Komputer | r ≠ k | - |
| 13 | dom | Dompet Kulit | r ≠ m | - |
| 14 | buk | Buku Sistem Digital | r ≠ k | - |
| 15 | buk | Buku *Database* | r ≠ k | - |
| 4 | 1 | hard | hand | Handphone Samsung | d = d | - |
| 2 | r ≠ n | - |
| 3 | head | Headset Putih | d = d | - |
| 4 | r ≠ a | - |
| 5 | hita | Buku Hitam Notes | d ≠ a | - |
| 6 | harg | Charger Iphone | d ≠ g | - |
| 7 | hone | Charger Iphone | d ≠ e | - |
| 8 | hard | Hardisk | d = d | - |
| 9 | r = r | - |
| 10 | a = a | - |
| 11 | h = h | Hardisk |
| 12 | hija | Buku Tulis Hijau | d ≠ a | - |
| 13 | buku | Buku Arsitektur Komputer | d ≠ u | - |
| 14 | domp | Dompet Kulit | d ≠ p | - |
| 15 | buku | Buku Sistem Digital | d ≠ u | - |
| 16 | buku | Buku *Database* | d ≠ u | - |

* 1. Menentukan Bobot Kriteria

Penentuan bobot kriteria dijelaskan pada tabel berikut:

**Tabel 4. 4 Pembobotan Kriteria**

| **Kriteria** | **Persentase pengaruh kriteria** | **Bobot *range* (0-1)** | **Keterangan** |
| --- | --- | --- | --- |
| Jumlah iterasi algoritma | 80% | 0,8 | Tingkat pengaruh iterasi algoritma terhadap kecepatan sangat tinggi karena semakin banyak perulangan/iterasi maka akan semakin lambat suatu algoritma menyelesaikan masalah |
| Jumlah huruf pada *pattern* | 20% | 0,2 | Jumlah huruf pada *pattern* merupakan kriteria yang juga mempengaruhi kecepatan, namun tidak lebih berpengaruh dari pada relasi |

* 1. Pemberian Nilai Pada Setiap Kriteria

Tahap ini adalah tahap dimana setiap kriteria yang telah terbentuk diberi nilai. Untuk dapat memberikan nilai, berikut adalah contoh hasil simulasi *auto-complete* Universitas Bakrie yang diambil dari pembahasan analisa pada bagian 4.7.2.

**Tabel 4. 5 Pemberian Nilai Pada Setiap Kriteria**

| **Alternatif** | **Proses ke-** | **Jumlah Iterasi Algoritma** | **Jumlah Huruf Pada *Pattern*** |
| --- | --- | --- | --- |
| *Boyer-Moore* | 1 | 10 | 1 |
| 2 | 14 | 2 |
| 3 | 15 | 3 |
| 4 | 16 | 4 |
| Brute Force | 1 | 10 | 1 |
| 2 | 18 | 2 |
| 3 | 22 | 3 |
| 4 | 23 | 4 |

* 1. Menghitung Skor

Nilai pada kriteria dimasukkan dalam nilai bobot yang telah ditentukan. Tahapan selanjutnya adalah dengan melakukan perhitungan menggunakan rumus Metode Perbandingan Eksponensial.

**Tabel 4. 6 Simulasi Perhitungan Analisa Menggunakan Perhitungan Perbandingan Eksponensial**

| **Proses Ke-** | **Kriteria** | | | | | | **Total Nilai BF** | **Total Nilai BM** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Jumlah Iterasi** | | | **Jumlah Huruf** | | |
| **B** | **BF** | **BM** | **B** | **BF** | **BM** |
| **N** | **N** | **N** | **N** |
| 1 | 0,8 | 10 | 10 | 0,2 | 1 | 1 | 7,310 | 7,310 |
| 2 | 0,8 | 18 | 14 | 0,2 | 2 | 2 | 11,246 | 9,407 |
| 3 | 0,8 | 22 | 15 | 0,2 | 3 | 3 | 13,102 | 9,973 |
| 4 | 0,8 | 23 | 16 | 0,2 | 4 | 4 | 13,605 | 10,509 |
| **TOTAL** | | | | | | | **45,262** | **37,199** |

Keterangan

1. BF : Algoritma *Brute Force*
2. BM : Algoritma *Boyer-Moore*
3. B : Nilai Bobot
4. N : Nilai Dari Kriteria
5. Total Nilai :

Contoh perhitungan :

Nilai pada proses 2.

Nilai BF =

=

= 11,249

Nilai BM =

=

= 9407

**Gambar 4. 26 Grafik Perhitungan Skor**

* 1. Menentukan Prioritas Keputusan

Setelah Total Nilai dari setiap alternatif dihitung maka tahapan selanjutnya adalah tahapan akhir yaitu menentukan prioritas keputusan berdasarkan total nilai dari setiap alternatif. Secara rinci dapat dilihat sebagai berikut:

**Tabel 4. 7 Prioritas Keputusan**

|  |  |  |
| --- | --- | --- |
| **Alternatif** | **Total Nilai** | **Rangking** |
| Algoritma *Boyer-Moore* | **37,199** | **1** |
| Algoritma Brute Force | **45,262** | **2** |

Pada tabel 4.7 terlihat bahwa total nilai dari alternatif terendah yang memperoleh rangking pertama, hal ini dikarenakan semakin tinggi jumlah usaha yang dilakukan oleh algoritma tersebut. Berdasarkan analisa tersebut maka Algoritma *Boyer-Moore* yang menjadi algoritma tercepat dalam pencarian menggunakan *auto-complete* serta permasalah dapat terpecahkan.

# **BAB V**

**SIMPULAN DAN SARAN**

1. **Simpulan**

Berdasarkan hasil analisis, perancangan, implementasi, dan penguian yang telah dilakukan, dapat diambil kesimpulan sebagai berikut.

1. Algoritma *Boyer-Moore* dapat diterapkan pada fitur *auto-complete* aplikasi *Lost and Found* berbasis *web*. Algoritma *Boyer-Moore* juga dapat berjalan dengan baik pada fitur *auto-complete* pencarian barang dengan melakukan pencarian dari karakter paling kanan ke karakter paling kiri dengan menggunakan pergeseran *pattern good-suffix shift rule* dan *bad character rule*.
2. Dari hasil pengujian menggunakan Metode Perbandingan Eksponensial, grafik perhitungan skor menunjukkan bahwa jumlah iterasi yang dilakukan oleh algoritma *Boyer-Moore* lebih sedikit daripada algoritma *Brute Force* yang dijadikan sebagai algoritma pembanding untuk fitur pencarian menggunakan fitur *auto-complete.*  Hasil pencarian algoritma *Boyer-Moore* juga bergantung pada dua hal, yaitu panjang *pattern* yang diinputkan dan kata kunci yang di-*input-*kan oleh *user.* Apabila *user* memasukkan jumlah *pattern* sedikit maka peluang untuk sistem menampilkan hasil pencarian lebih lama, semakin banyak jumlah *pattern* yang dimasukkan maka semakin kecil sistem mendapatkan sugesti kemiripannya.
3. **Saran**

Berdasarkan hasil penulisan yang telah dilakukan, maka beberapa saran untuk pengembangan penulisan selanjutnya adalah sebagai berikut.

1. Untuk meningkatkan akurasi dari hasil pencarian *auto-complete* menggunakan algoritma *Boyer-Moore,* maka diharapkan *user* melakukan beberapa hal dalam proses pencarian, yang pertama adalah *user* memasukkan data dengan nama barang yang spesifik disertai dengan tipe barang agar menghindari kesamaan barang yang disimpan. Kedua, diharapkan *user* melakukan pencarian dengan memasukkan nama barang yang spesifik (*pattern* yang panjang) supaya hasil pencarian lebih akurat.
2. Aplikasi *Lost and Found* ini dikembangkan untuk meningkatkan jumlah keakurasian dalam pencarian data menggunakan *auto-complete* dengan menambahkan metode pencocokan *string* lainnya*.*

# **DAFTAR PUSTAKA**

|  |  |
| --- | --- |
| [1] | Suprawoto, “Akurat, Cepat, Mudah, dan Merata Sebuah Praktik Pengelolaan Informasi Publik,” *Bali: Konferensi Perpustakaan Digital Indonesia ke-1,* pp. 1-11, 2008. |
| [2] | A. Januardi, “Analisa Perbandingan Algoritma Brute Force dan Boyer Moore dalam Pencarian Word Suggestion Menggunakan Metode Perbandingan Eksponensial,” *Pelita Informatika Budi Darma ISSN: 2301-9425,* vol. IV, no. 1, pp. 18 - 24, 2013. |
| [3] | G. H. Kumara, “Visualisasi Beberapa Algoritma Pencocokan String dengan Java,” *Jurusan Teknik Informatika, Fakultas Elektro Informatika, Institut Teknologi Bandung,* pp. 1-14, 2009. |
| [4] | D. A. Wibisono, D. Priharsari dan A. Muttaqin, “Rancang Bangun Sistem Informasi Pencarian Benda Hilang 'Lost and Found' Berbasis Website di Universitas Brawijaya,” pp. 1-8, 2012. |
| [5] | V. Sagita dan M. I. Prasetyowati, “Studi Perbandingan Implementasi Algoritma Boyer-Moore, Turbo Boyer-Moore, dan Tuned Boyer-Moore dalam Pencarian String,” *ULTIMATICS, ISSN 2085-4552,* vol. IV, no. 1, pp. 31-37, 2013. |
| [6] | Supriyanto, “Perancangan dan Pembuatan Sistem Informasi Kehilangan Berbasis Web,” pp. 1-12, 2014. |
| [7] | J. A. O'Brien dan G. M. Marakas, Management Information System 10e, New York : McGraw-Hill/Irwin , 2010. |
| [8] | M. Z. Halim, “Rancang Bangun Sistem Informasi Penyewaan Pada Rental Mobil Berbasis Web dan Menggunakan SMS Gateway,” *Laporan Skripsi ,* pp. 7-8, 2013 . |
| [9] | D. Utomo, E. W. Harjo dan Handoko, “Perbandingan Algoritma String Searching Brute Force, Knuth Morris Pratt, Boyer Moore, dan Karp Rabin pada Teks Alkitab Bahasa Indonesia,” *Techne Jurnal Ilmiah Elektronika ,* vol. 7, no. 1, pp. 1-13, 2011. |
| [10] | K. W. Argakusumah dan S. Hansun, “Implementasi Algoritma Boyer-Moore pada Aplikasi Kamus Kedokteran Berbasis Android,” *Ultimatics Vol. VI, No. 2 ISSN 2085-4552,* p. 71, 2014. |
| [11] | C. Chiquita, “Penerapan Algoritma Boyer-Moore Dynamic Programming untuk Layanan Auto-Complete dan Auro Correct,” *Makalah IF3051 Strategi Algoritma,* pp. 1-6, 2012. |
| [12] | S. Kristanto, A. Rachmat dan R. G. Santosa, “Implementasi Algoritma Boyer-Moore Pada Permainan Word Search Puzzle,” *Conference Paper,* pp. 1-9, 2013. |
| [13] | R. S. Pressman, Software Engineering, A Practitioner's Approach, Seventh Edition, New York : McGraw-Hill , 2010. |
| [14] | P. E. Mountaines, “Pengembangan Aplikasi Berbasis Web untuk Menampilkan Absensi dan Nilai Akhir Peserta Didik,” *Tugas Akhir ,* p. 3, 2013 . |
| [15] | A. Mujumdar, G. Masiwal dan P. Chawan, “Analysis of various Software Process Models,” *International Journal of Engineering Research and Applications (IJERA) ISSN: 2248 - 9622 vol.2, Issue 3,* pp. 2015 - 2021, 2012. |
| [16] | A. M. French, “Web Development Life Cycle: A New Methodology for Developing Web Applications,” *Journal of Internet Banking and Commerce, vol 16 no.2 ,* p. 5, 2011. |
| [17] | Kamatchi, J. Iyer dan S. Singh, “Software Engineering: Web Development Life Cycle,” *International Journal of Engineering Research & Technology (IJERT) ISSN:2278-0181,* vol. 2, no. 3, pp. 1-4, 2013. |
| [18] | S. Al-Fedaghi, “Developing Web Applications,” *International Journal of Software Engineering and Its Applications Vol. 5 No.2,* vol. 5, no. 2, pp. 57-68, 2011. |
| [19] | R. Nagila, “Comparison of Web Development Technologies - ASP.NET & PHP,” *Master Thesis in Web Development Malarden University ,* p. 12, 2013. |
| [20] | S. P. Chandran dan M. Angepat, “Comparison between ASP.NET and PHP - Implementation of a Real Estate Web Application,” *Master Thesis in Software Engineering, Malardalens hogskola Eskilstuna Vasteras,* pp. 34-35, 2011. |
| [21] | J. L. Whitten dan L. D. Bentley, System Analysis and Design Methods 7/E, New York : McGraw-Hill, 2007. |
| [22] | N. M. A. Munassar dan A. Govardan, “Comparison Study Between Traditional and Object-Oriented Approaches to Develop All Projects in Software Engineering,” *International Journal of Computer Science and Information Technologies ,* vol. 3, no. 1, pp. 3022-3028, 2012. |
| [23] | Marimin, Teknik dan Aplikasi Pengambilan Keputusan dengan Kriteria Majemuk, Jakarta: Grasindo, 2015. |
| [24] | H. R. Pratiwi, D. Syarif dan A. Wibowo, “Prototype Aplikasi SMS Content Filtering Menggunakan Metode String Matching (Studi Kasus : Content Iklan),” *Jurnal Teknik Informatika ,* vol. 1, pp. 1-8, 2012. |
| [25] | Kent, “Boyer-Moore Algorithm,” 4 April 2015. [Online]. Available: http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/StringMatch/boyerMoore.htm. |
| [26] | D. Mahardika, “Sistem Pendukung Keputusan Promosi Kenaikan Jabatan Manager dengan Metode Perbandingan Eksponensial Pada PT Texmaco Perkasa Engineering Kendal,” pp. 1-14, 2012. |

# **LAMPIRAN**