SKRIPSI

APLIKASI PEMERIKSA KESALAHAAN DOKUMEN SKRIPSI INFORMATIKA UNPAR

Marcell Trixie Alexander

NPM: 2014730003

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2019

UNDERGRADUATE THESIS

GENERAL ERROR CHECKER APPLICATION FOR INFORMATICS ENGINEERING UNPAR THESIS DOCUMENT

Marcell Trixie Alexander

NPM: 2014730003

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2019

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

Kata-kata kunci: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Keywords: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

Pada bab ini dijelaskan mengenai latar belakang penulisan skripsi, rumusan masalah, tujuan penulisan skripsi, batasan masalah, metodologi penelitian, dan sistematika penulisan skripsi ini.

1.1 Latar Belakang

Skripsi merupakan karangan ilmiah yang wajib ditulis oleh mahasiswa sebagai bagian dari persyaratan akhir pendidikan akademiknya. Namun dalam penulisannya, peserta skripsi sering melakukan kesalahan kecil yang tidak dapat diabaikan. Kesalahan sering terjadi dalam penggunaan imbuhan, kata keterangan, penulisan kata dan sebagainya. Hal-hal seperti ini seharusnya dapat diperiksa dan diminimalisir oleh diri sendiri. Pada saat bimbingan, waktu dosen pembimbing lebih baik dimanfaatkan untuk membahas konten dibanding memeriksa kesalahan-kesalahan tersebut.

Dari masalah tersebut dapat dibuat sebuah aplikasi untuk melakukan pemeriksaan pada dokumen skripsi. Kesalahan yang akan diperiksa berasal dari survei yang dilakukan kepada dosen-dosen Informatika Unpar. Hasil dari survei tersebut akan diseleksi untuk diimplementasikan ke dalam aplikasi. Aplikasi sederhana ini dapat dimanfaatkan oleh mahasiswa Informatika Unpar secara mandiri. Aplikasi ini dijalankan melalui melalui terminal *command Windows*. Aplikasi menerima masukan berupa file *PDF* skripsi dan menampilkan laporan yang berisi kesalahan-kesalahan yang ditemukan pada dokumen skripsi.

1.2 Rumusan Masalah

Berdasarkan deskripsi topik yang sudah ditulis, dapat dirumuskan masalah sebagai berikut:

1. Bagaimana cara memeriksa kesalahan yang ada pada dokumen skripsi?
2. Bagaimana cara membuat perangkat lunak yang dapat memeriksa kesalahan pada dokumen skripsi?

1.3 Tujuan

Tujuan dari skripsi ini adalah sebagai berikut:

1. Dapat memeriksa kesalahan yang ada pada dokumen skripsi
2. Dapat membangun perangkat lunak untuk memeriksa kesalahan yang ada pada dokumen skripsi

1.4 Batasan Masalah

Batasan masalah skripsi ini adalah sebagai berikut:

1. Pemeriksaan menggunakan *pattern matching* tanpa analisis gramatikal
2. Tidak ada pemeriksaan kosakata

1.5 Metodologi Penelitian

Metodologi penelitian yang digunakan pada skripsi ini adalah sebagai berikut:

1. Melakukan survei kepada dosen-dosen Informatika mengenai kesalahan-kesalahan penulisan yang ditemui dalam dokumen skripsi
2. Melakukan studi literatur *Regular Expression* untuk mendeteksi kesalahan-kesalahan dalam file *PDF* skripsi
3. Mempelajari *library PDF Parser* untuk mengestraksi file *PDF* skripsi yang akan diperiksa
4. Melakukan perancangan perangkat lunak
5. Melakukan implementasi perancangan perangkat lunak
6. Melakukan pengujian terhadap perancangan perangkat lunak
7. Menulis dokumen skripsi

1.6 Sistematika Pembahasan

Sistematika penulisan pada skripsi ini terdiri dari 6 bab, yaitu:

1. Bab 1 Pendahuluan
   Bab 1 akan membahas latar belakang dibuatnya perangkat lunak untuk memeriksa kesalahan dokumen skripsi. Pada bab ini dibahas juga rumusan masalah, tujuan skripsi, batasan masalah dan metodologi penelitian yang digunakan pada skripsi.
2. Bab 2 Landasan Teori
   Bab 2 yang merupakan landasan teori akan berisi teori-teori yang menjadi dasar-dasar dalam penulisan skripsi ini. Teori yang akan dibahas pada bab 2, yaitu *Regular Expression* dan *library PDF Parser*.
3. Bab 3 Analisis Masalah
   Bab 3 berisi analisis masalah yang muncul dalam menyelesaikan masalah tersebut. Pada bab ini akan dianalisa masalah yang ditemukan pada saat melakukan pengamatan beberapa sidang skripsi semester Ganjil 2018/2019 dan survei secara personal kepada dosen-dosen Informatika Unpar. Hasil dari setiap survei akan dipilih mana saja yang dapat diimplementasikan dalam perangkat lunak.

4. Bab 4 Perancangan
   Bab 4 berisi rancangan perangkat lunak yang akan dibuat, seperti perancangan kelas dan algoritma yang digunakan dalam pembangunan perangkat lunak. Perangkat lunak akan dibuat dengan menggunakan bahasa pemrogramam *PHP*.

5. Bab 5 Implementasi dan Pengujian
   Bab 5 pada skripsi ini membahas implementasi perangkat lunak dan pengujian yang dilakukan terhadap perangkat lunak tersebut. Bab ini juga menjelaskan tentang spesifikasi perangkat lunak dan pengujian yang dilakukan pada skripsi ini.

6. Bab 6 Kesimpulan dan Saran
   Bab 6 berisi kesimpulan dari penulisan skripsi ini. Bab ini juga berisi saran untuk pengembangan perangkat lunak agar lebih baik lagi.

BAB 2

LANDASAN TEORI

Pada bab ini akan dibahas mengenai landasan teori yang membahas *regular expression*, *library PDF Parser* dan kamus bahasa Indonesia *LibreOffice*.

2.1 *Regular Expression*

Regular expression (*regex*) [1] adalah jenis pola teks tertentu yang dapat digunakan pada banyak aplikasi modern dan bahasa pemrograman. *Regex* biasanya dimanfaatkan untuk memverifikasi kecocokan antara input dengan pola teks, untuk menemukan teks yang cocok dengan pola dalam teks yang lebih besar, untuk mengganti teks yang cocok dengan pola dengan teks lain atau menyusun ulang bit dari teks yang cocok dan untuk membagi sebuah blok teks menjadi beberapa subteks.

Regex sudah banyak digunakan dalam pencocokan pola, misalnya untuk validasi beberapa string seperti *username* dan *password*, alamat *e-mail*, alamat *IP* ataupun nomor telepon. Pemanfaatan *regex* dengan baik, dapat menyederhanakan banyak tugas pemrograman dan pemrosesan teks dalam kehidupan sehari-hari. Istilah *regex* berasal dari teori matematika dan komputer sains, yang mencerminkan sifat ekspresi dalam matematika yang disebut keteraturan. Ekspresi tersebut dapat diimplementasikan dalam perangkat lunak, dengan menggunakan *Deterministic Finite Automaton* (DFA). DFA adalah *finite state machine* yang tidak menggunakan *backtracking*.

Regex dapat digunakan dalam berbagai bahasa pemrograman, salah satunya yaitu, *Perl Compatible Regular Expression* (PCRE). PCRE [2] adalah serangkaian fungsi yang menerapkan pencocokan pola *regex* dengan menggunakan sintaks dan semantik yang sama dengan bahasa pemrograman *Perl 5*, meskipun ada beberapa sedikit perbedaan. Pada saat ini, implementasi yang digunakan sesuai dengan *Perl* versi 5.005.

2.1.1 Metakarakter

Metakarakter pada *regex* dibedakan menjadi 2 jenis berdasarkan dari posisinya, yaitu metakarakter *outside square brackets* dan metakarakter *outside square brackets*. Meskipun ada beberapa simbol metakarakter yang sama, namun fungsinya agak berbeda. Pada metakarakter *outside square brackets* terdapat 14 simbol, sedangkan metakarakter *inside square brackets* terdapat 3 simbol. Rincian dari ke-2 metakarakter tersebut akan dijelaskan sebagai berikut:

Tabel 2.1: Tabel metakarakter *outside square brackets*

Simbol	Nama
\	Backslash
^	Circumflex Anchor
$	Dollar Anchor
.	Dot
[ ]	Square Bracket
|	Vertical Bar
( )	Parenthesis
?	Question Mark
*	Asterisk
+	Plus
{ }	Curly Bracket

Pada tabel 2.1 telah disebutkan simbol-simbol yang digunakan pada metakarakter *outside square brackets*. Berikut ini adalah penjelasan fungsi dari setiap metakarakter:

1. *Backslash*
   Backslash yang berada di luar kelas karakter memiliki beberapa fungsi, yaitu:
   - Membuat karakter lepas
     Apabila diikuti oleh karakter non-alfanumerik, metakarakter ini dapat menghilangkan makna khusus yang dimiliki oleh karakter tersebut. Misalnya pada saat ingin mencocokan karakter "*", dengan menggunakan metakarakter *backslash* dapat dituliskan dengan "*". Jadi karakter *asterisk* akan terbaca sebagai karakter bukan sebagai metakarakter.
   - Menggunakan karakter yang tidak dapat ditulis dalam pola, seperti \a (*alarm*), \cx (*control-x*, \e (*escape*) dan lain-lain.
   - Menentukan jenis karakter dalam pola, seperti \d (angka desimal), \D (non-angka desimal), \w (karakter kata), \W (non-karakter kata) dan lain-lain.
   - Menggunakan pernyataan sederhana tertentu, seperti \b (*word boundary*), \B (*non-word boundary* dan lain-lain.
2. *Anchor*
   Anchor merupakan metakarakter yang terdiri dari simbol *circumflex* (^) dan *dollar* ($). *Circumflex* digunakan sebagai tanda awal yang memulai suatu teks string, sedangkan *dollar* digunakan sebagai tanda akhir dari suatu teks pola.
3. *Dot*
   Dot akan cocok dengan karakter apapun, keculai karakter *newline*.

4. *Square brackets*
   Square brackets terdiri dari pasangan "[" dan "]", memiliki fungsi untuk mendefinisikan kelas karakter. Kelas karakter akan berada di antara 2 karakter tersebut. Contohnya kelas karakter numerik [0-9] sama dengan [0123456789].

5. *Vertical bar*
   Vertical bar berfungsi untuk memisahkan beberapa kondisi pola alternatif yang berbeda. Sebagai contohnya untuk pola A | B, maka hasilnya akan mencocokan "A" atau "B".

6. *Parenthesis*
   Parenthesis terdiri dari pasangan "(" dan ")", memiliki fungsi untuk mengelompokan subpola dalam *regex*.

7. *Quantifiers*
   Quantifiers berfungsi untuk menunjukkan berapa banyak instance karakter, set karakter, atau kelas karakter yang harus dicocokkan. Pada metakarakter ini terdapat 3 jenis, yaitu:

   - *question mark* (?), dengan jumlah kuantifier { 0,1 } (0 hingga 1).
   - *asterisk* (*), dengan jumlah kuantifier { 0, } (0 atau lebih).
   - *plus* (+), dengan jumlah kuantifier { 1, } (1 atau lebih).

8. *Curly Brackets*
   Curly Brackets terdiri dari pasangan "" dan "", memiliki fungsi untuk mendefinisikan angka minimal dan maksimum dari kuantifiers yang akan digunakan. Misalnya pola a1,5, akan cocok dengan "a", "aa", "aaa", "aaaa" atau "aaaaa".

Metakarakter *inside square brackets* merupakan bagian dari kelas karakter. Pada bagian kelas karakter ini hanya terdapat 3 macam metakarakter saja. Berikut adalah metakarakter yang digunakan:

Tabel 2.2: Tabel metakarakter *inside square brackets*

Simbol	Nama
\	Backslash
^	Circumflex
-	Hyphen

Pada tabel 2.2 telah disebutkan macam-macam metakarakter *inside square brackets*. Dari ke-3 metakarakter tersebut ada beberapa yang memiliki simbol yang sama dengan metakarakter *outside square brackets*, namun fungsinya berbeda. Berikut adalah penjelasan dari fungsi metakarakter dari tabel 2.2

1. *Backslash*
   Metakarakter ini fungsinya sama dengan *backslash* yang ada pada *outside square brackets*.

Namun dari ke-4 fungsi tersebut hanya 3 fungsi saja yang digunakan, yaitu membuat karakter lepas, Menggunakan karakter yang tidak dapat ditulis dalam pola dan menentukan jenis karakter dalam pola.

2. *Circumflex*
   Metakarakter ini memiliki fungsi yang berbeda dengan yang digunakan pada *outside square brackets*. Fungsinya untuk membuat negasi, namun hanya berlaku untuk karakter pertamanya saja. Contohnya [^0] akan cocok dengan semua karakter kecuali karakter 0.

3. *Hyphen*
   Metakarakter ini berfungsi untuk menentukan jangkauan dari sebuah karakter dalam kelas karakter, seperti [0-9] yang menandakan jangkauan karakter dari angka 0 hingga 9.

2.1.2 Kelas Karakter

Kelas karakter adalah karakter yang memiliki atribut yang spesifik yang dikelompokan dalam sebuah kelas. Karakter tersebut dapat berbeda di setiap negara. Kelas karakter hanya valid digunakan pada *regex* didalam tanda kurung siku pada *bracket expression*. Perl mendukung notasi POSIX yang digunakan untuk kelas karakter. Dalam penggunaannya, kelas-kelas tersebut ditulis diantara "[:" dan ":]". PCRE juga mendukung penggunaan notasi ini. Sebagai contoh untuk kelas alfanumerik, penulisannya yaitu [:alnum:]. Berikut ini akan dijelaskan macam-macam kelas karakter yang digunakan:

Tabel 2.3: Tabel kelas karakter

Kelas	Deskripsi	Keterangan
alnum	Alfanumerik	Kelas yang berisi dengan karakter alfanumerik, meliputi angka dan huruf. Karakter-karakter yang termasuk yaitu, huruf a-Z atau A-Z dan angka 0-9. Simbol atau karakter khusus tidak termasuk dalam kelas ini.
alpha	Huruf	Kelas yang berisi dengan karakter huruf. Karakter-karakter yang termasuk yaitu, huruf a-Z atau A-Z. Simbol atau karakter khusus tidak termasuk dalam kelas ini.
ascii	Kode karakter	Kelas yang merepresentasikan kode karakter dari 0-127.
blank	Spasi dan Tab	Karakter-karakter yang termasuk yaitu, TAB dan spasi.

Tabel 2.4: Tabel kelas karakter

Kelas	Deskripsi	Keterangan
cntrl	Karakter kontrol	Karakter kontrol adalah karakter yang tidak merepresentasikan simbol tetapi merepresentasikan character encoding.
digit	Angka desimal	Kelas yang berisi dengan karakter numerik. Karakter-karakter yang termasuk yaitu, angka 0-9.
graph	Karakter cetak (kecuali spasi)	Kelas yang berisi karakter yang dapat dicetak dan tampak. Contoh karakter yang dapat dicetak namun tidak tampak adalah karakter spasi dan TAB. Jadi pada kelas ini karakter spasi dan tab tidak termasuk.
lower	Huruf kecil	Kelas yang berisi karakter dengan huruf kecil.
print	Karakter cetak (termasuk spasi)	Kelas yang berisi karakter yang dapat dicetak. Karakter yang termasuk kelas ini adalah spasi.
punct	Karakter cetak (kecuali alfanumerik)	Kelas yang berisi karakter yang dapat dicetak, namun tidak termasuk alfanumerik.
space	Ruang putih	Karakter yang termasuk kelas ini adalah spasi dan TAB.
upper	Huruf kapital	Kelas yang berisi karakter dengan huruf kapital.
word	Karakter ”word”	Kelas yang berisi karakter kata.
xdigit	Heksadesimal	Kelas yang merepresentasikan bilangan hexadesimal, a-f atau A-F dan 0-9.

2.2 PDF Parser

PDF Parser [3] adalah sebuah *library* yang digunakan untuk mengekstrak data yang ada dalam sebuah file PDF. *Library* ini digunakan untuk kebutuhan pengguna yang menggunakan bahasa pemrograman PHP. *PDF Parser* dapat digunakan pada *PHP* dengan versi 5.3 ke atas. Terdapat

beberapa fitur yang dimiliki oleh *library* ini. Berikut adalah fitur yang sudah dapat digunakan:

- Memuat / mengurai objek dan header
- Menampilkan data meta, seperti nama penulis, deskripsi dan sebagainya
- Menampilkan isi dari teks PDF
- Dapat digunakan untuk kompresi PDF
- Mendukung *MAC OS Roman charset encoding*
- Menangani *encoding* heksa dan oktal pada teks

Dari fitur yang sudah ada, masih ada yang belum bisa ditangani oleh *library* ini. *PDF Parser* belum dapat mengekstrak dokumen yang diamankan. Selain itu, *library* ini tidak dapat mendeteksi jenis teks yang dicetak miring, tebal dan bergaris bawah. Hingga saat ini, pengembangan *library PDF parser* masih terus berjalan.

PDF Parser memiliki beberapa kelas yang menjalankan fungsional dari *library ini.* Pada subbab berikut akan dijelaskan beberapa kelas dari *library PDF Parser.*

2.2.1 Kelas Parser

Kelas ini berfungsi untuk mengatur ekstraksi file PDF. *Method-method* yang terdapat pada kelas ini adalah sebagai berikut:

- $object
  Atribut ini adalah sebuah array, yang akan menyimpan objek-objek dari file PDF yang akan diekstrak.
- public function _construct()
  Berfungsi untuk konstruktor kelas Parser.
- public function parseFile($filename)
  Method ini berfungsi untuk mengekstrak isi file dokumen PDF.
  Parameter: nama file yang akan diekstrak.
  Kembalian: konten dari file yang akan diekstrak.
- public function parseContent($content)
  Method ini berfungsi untuk mengekstrak konten pada dokumen.
  Parameter: konten dari file PDF.
  Kembalian: dokumen yang akan diekstrak.
- protected function parseTrailer($structure, $document)
  Method ini berfungsi untuk Parameter: struktur dan dokumen. Parameter: struktur dan dokumen
  Kembalian: header dengan parameter trailer dan dokumen.

- protected function parseObject($id, $structure, $document)
  Method ini berfungsi untuk. Parameter: id, struktur dan dokumen

- protected function parseHeader($structure, $document)
  Method ini berfungsi untuk mengekstrak header pada dokumen.
  Parameter: struktur dan dokumen
  Kembalian: header dengan parameter trailer dan dokumen.

- protected function parseHeaderElement($type, $value, $document)
  Method ini berfungsi untuk mengekstrak elemen-elemen yang ada pada header.
  Parameter: tipe, value dan dokumen

2.2.2 Page

Kelas ini merepresentasikan halaman-halaman yang ada pada dokumen PDF. *Method-method* yang terdapat pada kelas ini adalah sebagai berikut:

- public function getFonts()
  Method ini berfungsi untuk mendapatkan font yang digunakan dalam PDF dalam bentuk array.
  Kembalian: mengembalikan array font.

- public function getFont($id)
  Method ini memiliki fungsi yang sama dengan method getFonts(). Namun method ini hanya mengembalikan nilai pada indeks tertentu saja, berdasarkan id yang didapatkan dari parameter.
  Parameter: id dengan tipe data String.
  Kembalian: megembalikan font.

- public function getXObjects()
  Method ini berfungsi untuk.
  Kembalian: mengembalikan sebuah array PDFObject.

- public function getXObject($id)
  Method ini berfungsi untuk.
  Parameter: id dengan tipe data String.
  Kembalian: mengembalikan PDFObject.

- public function getText(Page $page = null)
  Method ini berfungsi untuk mendapatkan isi teks dari file PDF yang telah melalui proses ekstrak dalam bentuk string.
  Parameter: atribut *page* dengan tipe data kelas Page.
  Kembalian: mengembalikan isi teks.

- public function getTextArray(Page $page = null)
  Method ini berfungsi untuk mendapatkan isi teks dari file PDF yang telah melalui proses ekstrak dalam bentuk array.

Parameter: atribut *page* dengan tipe data kelas Page.

Kembalian: mengembalikan isi teks dalam sebuah array.

2.3 Kamus Indonesia *LibreOffice*

LibreOffice [4] adalah sebuah paket aplikasi perkantoran berfitur lengkap yang tersedia secara gratis. LibreOffice menggunakan *Open Document Format* (ODF) sebagai format aslinya untuk menyimpan dokumen. ODF menjadi format standar terbuka yang sedang diadopsi sebagai format file yang dibutuhkan untuk penerbitan dan penerimaan dokumen. LibreOffice juga dapat membuka dan menyimpan dokumen dalam format lainnya. Salah satunya format yang digunakan oleh beberapa versi dari Microsoft Office.

LibreOffice memiliki ekstensi untuk kamus Indonesia. Ekstensi ini sudah mengalami beberapa perkembangan, mulai dari versi 1.0 yang rilis pada tanggal 19 Mei 2012. Versi 1.0 merupakan hasil unggahan kembali dari ekstensi *OpenOffice* yang terakhir diperbaharui pada tahun 2009. Pada tanggal 17 Mei 2014 versi 1.1 dirilis, pada versi ini LibreOffice versi 4.0 dapat menggunakan ekstensi ini. Selanjutnya, pada tanggal 15 Juli 2014 versi 2.0 dirilis. Pada versi yang paling baru ini digunakan metode baru dengan sirkumfiks yang kemudian mengubah daftar kata, sehingga memuat semua lema dari Kamus Besar Indonesia.

Ekstensi kamus Indonesia ini dapat diunduh secara gratis oleh para pengguna melalui halaman resmi dari *LibreOffice*. Ekstensi yang dapat diunduh tersebut memiliki nama *id_id.oxt*. Dalam ekstensi tersebut, terdapat beberapa file yang berisi informasi tentang kamus Indonesia, yaitu *id_ID.aff* dan *id_ID.dic*.

2.3.1 File *id_ID.dic*

File ini berisi kata-kata yang akan digunakan dalam kamus bahasa Indonesia. File ini dapat dibuka dengan menggunakan aplikasi *notepad* atau teks editor lainnya. Setelah dibuka menggunakan teks editor, akan terlihat 31129 baris yang berisi kata-kata yang ada dalam kamus bahasa Indonesia *LibreOffice*. Untuk menjelaskan hal tersebut, akan diambil beberapa potong baris dari file tersebut.

Listing 2.1: Potongan kode untuk file *id_ID.dic*

```
1  31128
2  a
3  ab
4  aba
5  aba-aba
6  abad/i0
7  abadi/DkMkO0k0nl
8  abadiah
9  abah/DkMk
10 abah-abah
```

Pada listing 2.1, baris pertama pada file ini menyatakan banyaknya kata yang ada pada kamus. Kata yang ada dalam kamus tersebut berjumlah 31128 buah. Baris ke-2 hingga ke-31128 merupakan kata-kata yang ada pada kamus. Namun ada beberapa kata tersebut ada yang terlihat berbeda, seperti pada baris 6, 7 dan 9. Di sebelah kiri dari kata tersebut terdapat garis miring yang disertai

sebuah kode. Kode tersebut berarti bahwa kata tersebut dapat ditambahkan oleh imbuhan tertentu. Bagian ini akan dijelaskan lebih lanjut pada pembahasan file *id_ID.aff*.

2.3.2 File *id_ID.aff*

File ini berisi daftar awalan dan akhiran untuk kamus Indonesia. Awalan akan disimbolkan dengan huruf kapital, sedangkan akhiran akan disimbolkan dengan huruf kecil. Selain ke-2 hal tesebut, ada juga yang merupakan gabungan dari awalan dan akhiran. Berikut adalah penjelasan dari awalan dan akhiran:

1. *Prefiks*
   Prefiks atau awalan adalah imbuhan yang ditambahkan pada bagian awal sebuah kata dasar atau bentuk dasar, misalnya ber-, pe-, me- dan lain-lain. Penulisan kode awalan pada file ini akan disimbolkan dengan huruf kapital.

Listing 2.2: Potongan kode untuk prefiks

```
1 PFX B0 Y 2 # ber-
2 PFX B0 0 ber    [^r]
3 PFX B0 0 be      r
```

Listing 2.2 merupakan salah satu contoh dari awalan yang terdapat pada kamus bahasa Indonesia. Berikut ini adalah penjelasan dari baris kode tersebut:

- PFX
  Menandakan bahwa kode tersebut merupakan awalan
- B0
  Huruf B menunjukan awalan yang dimulai dengan huruf B, dan angka 0 menunjukan bentuk awalan asli
- 2
  Menandakan bahwa ada 2 jenis awalan yang dapat digunakan, yaitu ber- dan be-
- -ber
  Menandakan bahwa kode tersebut merupakan awalan ber-

2. *Sufiks*
   Sufiks atau akhiran adalah imbuhan yang ditambahkan pada bagian belakang kata dasar, misalnya -an, -kan, -i dan lain-lain. Penulisan kode akhiran pada file ini akan disimbolkan dengan huruf kecil.

Listing 2.3: Potongan kode untuk sufiks

```
1 SFX k0 Y  3  # -kan
2 SFX k0 0  kan    .
3 SFX k0 0  kanlah
4 SFX k0 0  kankah
```

Listing 2.3 merupakan salah satu contoh dari akhiran yang terdapat pada kamus bahasa Indonesia. Berikut ini adalah penjelasan dari baris kode tersebut:

- SFX
  Menandakan bahwa kode tersebut merupakan akhiran
- k0
  Huruf k menunjukan akhiran yang dimulai dengan huruf k, dan angka 0 menunjukan bentuk akhiran asli
- 3
  Menandakan bahwa ada 3 jenis akhiran yang dapat digunakan, yaitu -kan, -kanlah dan -kankah
- -kan
  Menandakan bahwa kode tersebut merupakan akhiran -kan

3. *Konfiks*
   Konfiks adalah imbuhan tunggal yang terbentuk dari perpaduan awalan dan akhiran.

Listing 2.4: Potongan kode untuk konfiks

```
1 PFX KT Y 2 # keter- (keter-an)
2 PFX KT 0 keter/A1 [^r]
3 PFX KT 0 kete/A1 r
4
5 SFX Sa Y 1 # -an (se-an) + o0
6 SFX Sa 0 an/S1o0A1 .
```

Listing 2.4 merupakan salah satu contoh dari konfiks yang terdapat pada kamus bahasa Indonesia. Berikut ini adalah penjelasan dari baris kode tersebut:

- Kolom pertama biasanya digunakan sebagai tanda untuk mengetahui apakah kode tersebut termasuk awalan atau akhiran. Namun pada konfiks tidak ada simbol khusus yang menandakan bahwa kode tersebut adalah sebuah konfiks. Pada baris 1 tertulis ”PFX”, yang berarti konfiks ini berakar pada awalan tertentu. Pada baris 5 tertulis ”SFX”, yang berarti konfiks ini berakar pada akhiran tertentu.
- Kolom ke-2 menunjukan simbol dari awalan atau akhiran yang digunakan. Pada baris 1, konfiks itu berasal dari awalan keter-, dapat dilihat bahwa huruf K berarti ”ke” dan huruf T adalah ”ter”. Pada baris 5, huruf S berarti awalan yang berasal dari huruf S dan huruf a berarti akhiran yang berasal dari huruf a.
- Kolom ke-4 menunjukan berapa banyak jumlah konfiks yang dapat digunakan. Pada baris 1 terdapat 2 jenis, sedangkan pada baris 5 terdapat 1 jenis saja.
- Kolom ke-5 menunjukan jenis konfiks yang digunakan. Baris 1-3 merupakan contoh dari konfiks keter-an, sedangkan baris 5-6 merupakan contoh dari konfiks se-an.

BAB 3

ANALISIS MASALAH

Pada bab ini akan dibahas survei kesalahan umum dan keputusan implementasi hasil survei.

3.1 Survei Kesalahan Umum

Pada bagian ini akan dijelaskan tentang survei yang dilakukan untuk mengumpulkan informasi yang dibutuhkan dalam pengembangan perangkat lunak. Informasi yang dicari adalah tentang kesalahan-kesalahan umum yang sering terjadi pada penulisan dokumen skripsi. Untuk mengumpulkan informasi tersebut, metode yang dipilih adalah melakukan survei. Dalam pelaksanaannya, survei dibagi menjadi dua, yaitu pengamatan beberapa sidang skripsi dan wawancara secara personal kepada dosen-dosen Informatika Unpar.

3.1.1 Pengamatan Sidang

Pengamatan dilakukan pada sidang skripsi semester Ganjil 2018/2019, yang berlangsung pada bulan Mei 2019. Tidak semua sidang skripsi yang berlangsung diamati, melainkan dari 42 sidang skripsi hanya diambil 7 sidang skripsi saja. Hal tersebut dilakukan dengan pertimbangan dari ke-7 sidang skripsi tersebut diuji oleh dosen Informatika yang berbeda-beda. Namun ada beberapa dosen Informatika yang tidak masuk dalam pengamatan, karena tidak dapat menghadiri sidang yang diuji oleh dosen tersebut. Data dari sidang yang akan diamati akan disajikan pada tabel 3.1 dan 3.2:

Tabel 3.1: Tabel informasi sidang skripsi yang diamati

Tanggal	Mahasiswa	Judul Skripsi	Penguji
15-05-2019	Osfaldo Mickael Oktavianus Naibaho	Sistem Informasi Penjualan Barang Pada Apotek	-Vania Natali, S.Kom, M.T. -Elisati Hulu, M.T.
16-05-2019	Ricky Wahyudi	Temu Kembali Gambar Menggunakan Fitur Surf dan Warna	-Dr.rer.nat. Cecilia Esti Nugraheni, ST, MT -Dr. Ir. Veronica Sri Moertini, MT.

Tabel 3.2: Tabel informasi sidang skripsi yang diamati

Tanggal	Mahasiswa	Judul Skripsi	Penguji
17-05-2019	Billy Adiwijaya	Pembangkit Timelapse Pengembangan Proyek Perangkat Lunak	-Kristopher David Harjono M.T. -Elisati Hulu M.T.
20-05-2019	Ihsan Fajari	Sistem Informasi Rekomendasi Pariwisata di Tasikmalaya	-Dra. Rosa de Lima Endang Padmowati, MT -Dr. Ir. Veronica Sri Moertini, MT.
22-05-2019	Muhammad Adrian Putra Zubir	Sistem Informasi Penyediaan Barang Pada Apotek	-Dra. Rosa de Lima Endang Padmowati, MT -Pascal Alfadian Nugroho, S.Kom, M.Comp.
23-05-2019	Ellena Angelica	Kolektor Pengumuman Informatika	-Natalia S.Si, M.Si -Dr. Ir. Veronica Sri Moertini, MT.
24-05-2019	Evelyn Wijaya	Open Source Snake 360	-Chandra Wijaya S.T., M.T. -Raymond Chandra Putra, S.T., M.T.

Dari ke-7 pengamatan tersebut, terdapat beberapa kesalahan-kesalahan yang terjadi dalam penulisan dokumen skripsi. Hasil dari pengamatan tersebut akan dijelaskan pada tabel 3.3 dan 3.4:

Tabel 3.3: Tabel hasil pengamatan sidang skripsi

Kode	Jenis kesalahan	Keterangan
PS-01	Penulisan kata	Kesalahan dalam penulisan kata merupakan salah satu kesalahan yang sering terjadi. Pada umumnya lebih dikenal dengan istilah *typo*.
PS-02	Penggunaan imbuhan di- dan kata depan di	Penulisan imbuhan di- disatukan antara imbuhan dengan kata dasarnya. Untuk kata depan, penulisannya dipisah antara kata depan dengan kata berikutnya. Pada umumnya diikuti oleh keterangan tempat atau waktu.

Tabel 3.4: Tabel hasil pengamatan sidang skripsi

Kode	Jenis kesalahan	Keterangan
PS-03	Pemberian spasi sebelum dan setelah tanda baca	Salah satu hal kecil yang sering mengganggu adalah penggunaan spasi sebelum dan setelah tanda baca. Tanda baca yang paling sering dipakai, seperti titik, koma, tanya, dan seru harus diberi spasi setelahnya. Spasi juga digunakan sebelum menggunakan tanda kurung buka. Ada beberapa kesalahan yang masih ditemukan seperti, memberi spasi sebelum tanda tanya ataupun memberi spasi sebelum dan setelah garis miring.
PS-04	Pemberian spasi sebelum dan setelah tanda baca	Masalah ini sering ditemukan dalam penulisan dokumen skripsi, biasanya terjadi pada saat menyisipkan gambar atau tabel. Susunan atau ukuran gambar yang tidak tepat dapat mengakibatkan terciptanya ruang kosong yang besar.
PS-05	Awal kalimat tidak menggunakan huruf kapital	Setiap huruf pertama pada kata pertama dalam sebuah kalimat harus ditulis dengan huruf kapital.
PS-06	Pemberian spasi sebelum dan setelah tanda baca	Setiap kata dalam sebuah kalimat dipisahkan dengan jarak 1 spasi agar kalimat dapat dibaca dan dimengerti dengan baik.
PS-07	Pemberian spasi sebelum dan setelah tanda baca	Pada PDF Latex, biasanya kesalahan ini karena mahasiswa tidak memberikan tag kepada gambar tersebut. Hal ini mengakibatkan posisi gambar tidak terletak pada tempat yang seharusnya.
PS-08	Pemberian spasi sebelum dan setelah tanda baca	Dalam penulisan dokumen skripsi, setiap gambar dan tabel perlu diberikan keterangan.
PS-09	Pemberian spasi sebelum dan setelah tanda baca	Dalam sebuah bab, biasanya jumlah subbab lebih dari 1. Kesalahan yang sering dilakukan oleh mahasiswa yaitu, hanya terdapat 1 subbab saja pada 1 bab. Apabila dalam bab tersebut hanya terdapat 1 subbab, lebih baik tidak perlu dibuat subbab.

3.1.2 Wawancara Personal

Survei tahap selanjutnya yaitu melakukan dengan melakukan wawancara secara personal. Narasumber dari wawancara ini adalah dosen-dosen Informatika Unpar. Namun tidak semua dosen Informatika diminta untuk menjadi narasumber. Hasil dari wawancara tersebut akan dijelaskan pada tabel 3.5 hingga tabel 3.7:

Tabel 3.5: Tabel hasil wawancara dosen

Tanggal	Narasumber	Hasil Wawancara	Penjelasan
9-07-2019	Keenan Adiwijaya Leeman S.T.	KAL-01 Cetak miring untuk bahasa asing	Penggunaan kata dalam bahasa asing harus ditulis menggunakan cetak miring. Mahasiswa sering lupa untuk menulis cetak miring bahasa asing.
		KAL-02 Kalimat pengantar untuk setiap subbab	Setiap penulisan bab dan subbab selalu diikuti dengan kalimat pengantar untuk memulai bab dan subbab tersebut. Kesalahan yang sering terjadi, yaitu mahasiswa seringkali lupa untuk menuliskan kalimat pengantar tersebut.
		KAL-03 Kelengkapan data skripsi	Data skripsi harus diisi dengan lengkap sebagai bentuk identitas, seperti nama mahasiswa, NPM, dosen pembimbing, judul skripsi dan sebagainya. Hal-hal seperti seringkali lupa diisi karena terlalu fokus dalam mengerjakan konten-konten dalam skripsi.
9-07-2019	Chandra Wijaya S.T., M.T.	CHW-01 Letak keterangan untuk gambar dan tabel	Kesalahan yang sering terjadi adalah letak dari penulisan keterangan tersebut. Keterangan pada gambar posisinya ada di bawah gambar, sedangkan keterangan pada tabel posisinya ada di atas tabel.

Tabel 3.6: Tabel hasil wawancara dosen

Tanggal	Narasumber	Hasil Wawancara	Penjelasan
		CHW-02 Penggunaan bahasa yang benar	KBBI menjadi kaidah dalam penulisan bahasa Indonesia. Mahasiswa terkadang salah memilih kata yang hendak ditulis dalam dokumen, padahal kata tersebut tidak sesuai dengan KBBI.
15-07-2019	Husnul Hakim, S.Kom., M.T.	HUH-01 Rujukan untuk gambar dan tabel	Setiap gambar dan tabel yang dimasukan ke dalam dokumen skripsi, perlu dirujuk dalam sebuah paragraf. Mahasiswa sering lupa atau terlewat untuk merujuk gambar dan tabel tersebut.
		HUH-02 Penulisan pseudocode	Dalam penulisan pseudocode hal-hal yang perlu diperhatikan antara lain nama method, masukan serta keluaran pada method dan no baris pada pseudocode.
		HUH-03 Penulisan kata hubung	Kesalahan penggunaan konjungsi akan berakibat tidak jelasnya makna kalimat karena hubungan antar frasa dan antar klausa tidak jelas.
16-07-2019	Vania Natali, S.Kom, M.T.	VAN-01 Tahun skripsi pada cover skripsi	Penulisan tahun skripsi harus sama dengan tahun dimana mahasiswa mengambil skripsi tersebut. Kesalahan yang pernah terjadi, yaitu mahasiswa salah menuliskan tahun skripsi. Meskipun terlihat sepele, namun hal ini perlu diperhatikan.
		VAN-02 Konsistensi penggunaan kata	Mahasiswa harus konsisten dalam penulisan kata, misalnya kata*user* dan pengguna. Mahasiswa harus memilih antara memakai *user* atau pengguna.

Tabel 3.7: Tabel hasil wawancara dosen

Tanggal	Narasumber	Hasil Wawancara	Penjelasan
		VAN-02 Penggunaan kata ganti orang	Dalam penulisan dokumen skripsi, tidak boleh ada kata ganti orang. Jika karya non-ilmiah lebih santai karena memakai gaya bahasa non-formal, maka berbeda dengan karya ilmiah. Karya ilmiah memiliki aturan baku dan menggunakan bahasa formal.
16-07-2019	Natalia S.Si, M.Si	NAT-01 Penulisan daftar referensi	Kesalahan yang sering terjadi, yaitu tidak ditemukannya referensi yang akan digunakan. Pada teks yang akan dirujuk, akan terdapat tanda [?], seharusnya tanda tanya tersebut diisi oleh nomor dari referensi.

3.2 Keputusan Implementasi Hasil Survei

Pada bagian ini akan dijelaskan tentang keputusan implementasi dari hasil survei. Setiap hasil survei yang didapatkan melalui pengamatan sidang skripsi dan wawancara dosen, telah diberikan sebuah kode untuk digunakan dalam proses implementasi. Namun, tidak semua dari hasil survei tersebut dapat diimplementasikan menggunakan *regex*. Metode yang digunakan untuk mendeteksi kesalahan yaitu dengan *pattern matching*, sehingga hal-hal yang bersifat kontekstual tidak dapat dicek dengan *regex*. Berikut ini adalah hasil keputusan yang telah diambil pada setiap hasil survei di atas:

Tabel 3.8: Tabel keputusan implementasi

Kode	Survei	Keputusan	Alasan
PS-01	Penulisan Kata	Diimplementasi	Dapat diselesaikan menggunakan *regular expression*
PS-02	Penggunaan imbuhan di- dan kata depan di-	Tidak diimplementasi	Tidak dapat diselesaikan menggunakan *regular expression*, karena pada kamus Indonesia *LibreOffice* tidak ada fitur untuk membedakan kata sebagai keterangan atau bukan.

Tabel 3.9: Tabel keputusan implementasi

Kode	Survei	Keputusan	Alasan
PS-03	Pemberian spasi sebelum dan setelah tanda baca	Diimplementasi	Dapat diselesaikan menggunakan *regular expression*
PS-04	Terdapat ruang kosong yang besar	Tidak diimplementasi	Tidak dapat diselesaikan menggunakan *regular expression*, karena hasil ekstrak dari PDF menggunakan *PDF Parser* tidak mendeteksi adanya baris kosong.
PS-05	Awal kalimat tidak menggunakan huruf kapital	Diimplementasi	Dapat diselesaikan menggunakan *regular expression*
PS-06	Tidak ada spasi antar kata	Tidak diimplementasi	Dapat diselesaikan menggunakan *regular expression*. Namun survei ini tidak diimplementasi, karena penyelesaiannya sama dengan survei PS-01, sehingga akan disatukan implementasinya.
PS-07	Gambar tidak sesuai tempatnya	Tidak diimplementasi	Tidak dapat diselesaikan menggunakan *regular expression*, karena hasil ekstrak PDF dari *PDF Parser* tidak dapat mendeteksi gambar.
PS-08	Tidak ada keterangan untuk gambar dan tabel	Tidak diimplementasi	Tidak dapat diselesaikan menggunakan *regular expression*, karena hasil ekstrak PDF dari *PDF Parser* tidak dapat mendeteksi gambar dan tabel.
PS-09	Jumlah subbab dalam 1 bab tidak boleh hanya 1	Diimplementasi	Dapat diselesaikan menggunakan *regular expression*
KAL-01	Cetak miring untuk bahasa asing	Tidak diimplementasi	Tidak dapat diselesaikan menggunakan *regular expression*, karena membutuhkan kamus bahasa Inggris. Selain itu *PDF Parser* tidak dapat mencocokan teks yang cetak miring.
KAL-02	Kalimat pengantar untuk setiap subbab	Diimplementasi	Dapat diselesaikan menggunakan *regular expression*.
KAL-03	Kelengkapan data skripsi	Diimplementasi	Dapat diselesaikan menggunakan *regular expression*

Tabel 3.10: Tabel keputusan implementasi

Kode	Survei	Keputusan	Alasan
CHW-01	Letak keterangan untuk gambar dan tabel	Tidak diimplementasi	Tidak dapat diselesaikan menggunakan *regular expression*, karena hasil ekstrak PDF dari *PDF Parser* tidak dapat mendeteksi gambar dan tabel.
CHW-02	Penggunaan bahasa yang benar	Tidak diimplementasi	Dapat diselesaikan menggunakan *regular expression*. Namun survei ini tidak diimplementasi, karena penyelesaiannya sama dengan survei PS-01, sehingga akan disatukan implementasinya.
HUH-01	Rujukan untuk gambar dan tabel	Tidak diimplementasi	Tidak dapat diselesaikan menggunakan *regular expression*, karena hasil ekstrak PDF dari *PDF Parser* tidak dapat mendeteksi gambar dan tabel.
HUH-02	Penulisan pseudocode	Tidak diimplementasi	Tidak dapat diselesaikan menggunakan *regular expression*, karena hasil ekstrak PDF dari *PDF Parser* tidak dapat mendeteksi pseudocode.
HUH-03	Penulisan kata hubung	Tidak diimplementasi	Tidak dapat diselesaikan menggunakan *regular expression*, karena *regex* tidak dapat memeriksa kata hubung yang digunakan sudah tepat atau belum berdasarkan fungsi dari kata hubung tersebut.
VAN-01	Tahun skripsi pada cover skripsi	Tidak diimplementasi	Dapat diselesaikan menggunakan *regular expression*. Namun survei ini tidak diimplementasi, karena penyelesaiannya sama dengan survei KAL-03, sehingga akan disatukan implementasinya.
VAN-02	Konsistensi penggunaan kata	Tidak diimplementasi	Tidak dapat diselesaikan menggunakan *regular expression*, karena tidak dapat membuat padanan kata untuk memeriksa kosistensi penggunaan kata.

Tabel 3.11: Tabel keputusan implementasi

Kode	Survei	Keputusan	Alasan
VAN-03	Penggunaan kata ganti orang	Diimplementasi	Dapat diselesaikan menggunakan *regular expression*
NAT-01	Penulisan daftar referensi	Diimplementasi	Dapat diselesaikan menggunakan *regular expression*

Seperti yang sudah dijabarkan pada tabel 3.8 hingga tabel 3.11, 8 dari 21 hasil survei akan diimplementasikan menjadi fitur dalam perangkat lunak. Keputusan tersebut diambil berdasarkan dapat / tidaknya kesalahan tersebut diperiksa menggunakan *pattern matching regex* dan tingkat kesulitan untuk memeriksa kesalahan tersebut.

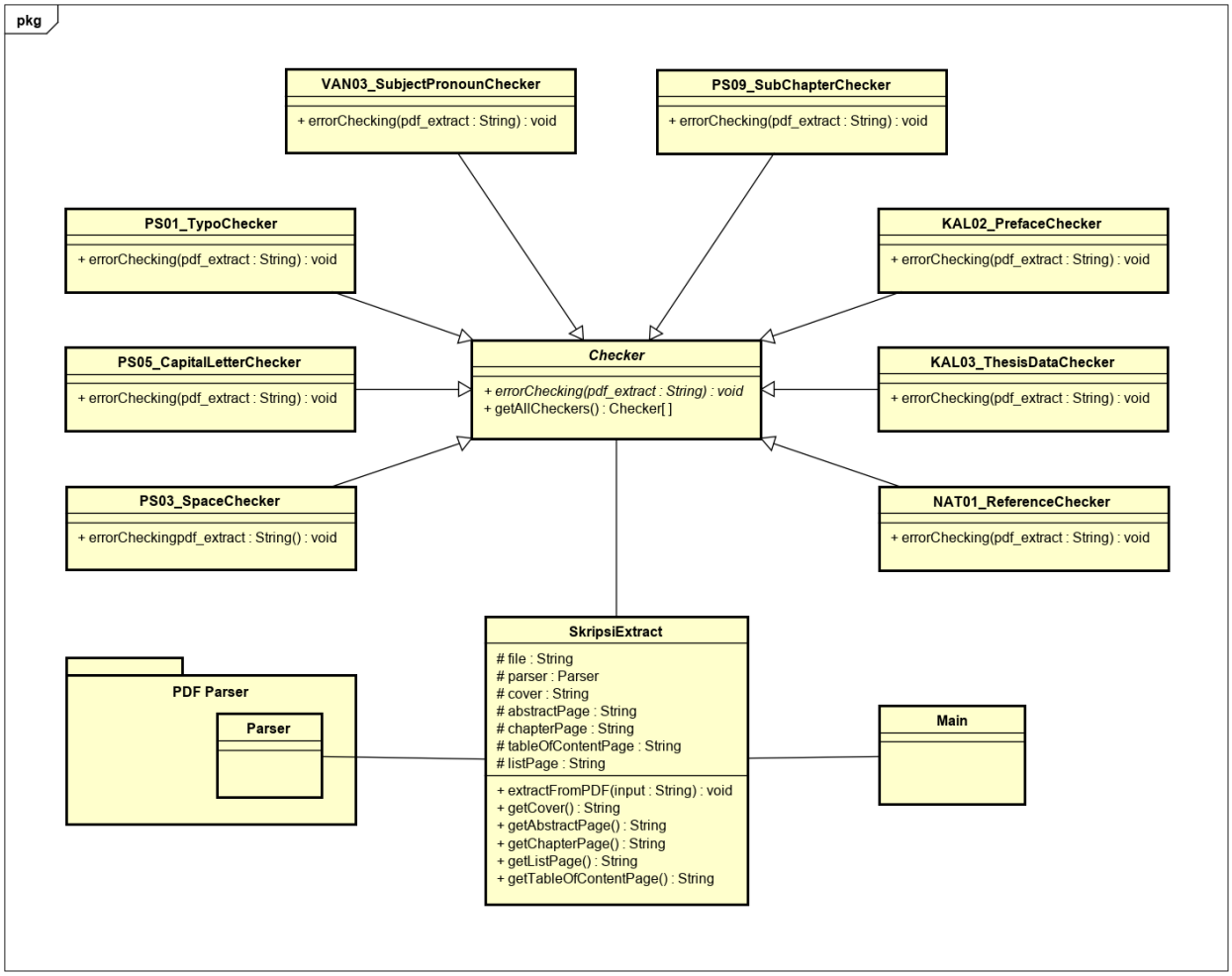
BAB 4

PERANCANGAN

Pada bab ini dibahas mengenai perancangan perangkat lunak yang dibangun, meliputi perancangan kelas dan algoritma pengecekan dokumen skripsi.

4.1 Perancangan Kelas

Pada bagian ini akan dijelaskan rancangan kelas yang akan digunakan pada perangkat lunak. Rancangan kelas tersebut akan ditunjukan oleh diagram kelas di bawah ini:



Gambar 4.1: Diagram kelas Aplikasi Pemeriksa Kesalahan Dokumen Skripsi

Pada gambar 4.1 telah ditunjukan bahwa perangkat lunak memiliki sebelas kelas dan sebuah *library PDF Parser*. Rincian dari setiap kelas tersebut akan dijelaskan sebagai berikut:

1. Kelas Checker
   Kelas ini merupakan kelas *Parent* dari semua *checker* yang akan diimplementasi pada perangkat lunak. Berikut adalah *method* yang terdapat pada kelas ini:

   - errorChecking($pdf_extract)
     Method ini merupakan method abstrak, yang akan diturunkan kepada seluruh anak kelasnya. Method ini berfungsi untuk memeriksa kesalahan pada dokumen skripsi sesuai dengan peran yang diberikan pada kelas tersebut. Method ini menerima masukan pdf_extract dengan tipe data kelas *SkripsiExtract*. Parameter tersebut dapat digunakan oleh masing-masing kelas *Checker* untuk memanggil *method getter* yang diperlukan. Tidak semua pemeriksa memerlukan seluruh isi halaman dari dokumen skripsi.
   - getAllChecker()
     Method ini berfungsi untuk melakukan instansiasi seluruh anak kelas *Checker*. Method ini mengembalikan hasil instansiasi anak kelas *Checker*.

2. Kelas KAL02_PrefaceChecker
   Kelas ini bertanggungjawab untuk memeriksa ada atau tidaknya kata pengantar sebelum memulai bab atau subbab. Berikut adalah *method* yang terdapat pada kelas ini:

   - errorChecking($pdf_extract)
     Method ini berfungsi untuk memeriksa kesalahan pada dokumen skripsi. Method ini menerima masukan pdf_extract dengan tipe data kelas *SkripsiExtract*. Bagian yang diperiksa pada *checker* ini hanya bab 1 hingga 6.

3. Kelas KAL03_ThesisDataChecker
   Kelas ini bertanggungjawab untuk memeriksa kelengkapan data skripsi yang ditulis dalam bahasa Indonesia maupun bahasa Inggris. Berikut adalah *method* yang terdapat pada kelas ini:

   - errorChecking($pdf_extract)
     Method ini berfungsi untuk memeriksa kesalahan pada dokumen skripsi. Method ini menerima masukan pdf_extract dengan tipe data kelas *SkripsiExtract*. Bagian yang diperiksa pada *checker* ini adalah halaman cover bahasa Indonesia dan bahasa Inggris.

4. Kelas NAT01_ReferenceChecker
   Kelas ini bertanggungjawab untuk memeriksa referensi yang akan dirujuk dalam dokumen. Berikut adalah *method* yang terdapat pada kelas ini:

   - errorChecking($pdf_extract)
     Method ini berfungsi untuk memeriksa kesalahan pada dokumen skripsi. Method ini menerima masukan pdf_extract dengan tipe data kelas *SkripsiExtract*. Bagian yang diperiksa pada *checker* ini adalah bab 1 hingga 6.

5. Kelas PS01_TypoChecker
   Kelas ini bertanggungjawab untuk memeriksa kesalahan penulisan kata. Berikut adalah *method* yang terdapat pada kelas ini:

   - errorChecking($pdf_extract)
     Method ini berfungsi untuk memeriksa kesalahan pada dokumen skripsi. Method ini menerima masukan pdf_extract dengan tipe data kelas *SkripsiExtract*. Bagian yang diperiksa pada *checker* ini adalah bab 1 hingga 6.

6. Kelas PS03_SpaceChecker
   Kelas ini bertanggungjawab untuk memeriksa penggunaan spasi sebelum dan setelah tanda baca. Berikut adalah *method* yang terdapat pada kelas ini:

   - errorChecking($pdf_extract)
     Method ini berfungsi untuk memeriksa kesalahan pada dokumen skripsi. Method ini menerima masukan pdf_extract dengan tipe data kelas *SkripsiExtract*. Bagian yang diperiksa pada *checker* ini adalah bab 1 hingga 6.

7. Kelas PS05_CapitalLetterChecker
   Kelas ini bertanggungjawab untuk memeriksa penggunaan huruf kapital pada awal kalimat. Berikut adalah *method* yang terdapat pada kelas ini:

   - errorChecking($pdf_extract)
     Method ini berfungsi untuk memeriksa kesalahan pada dokumen skripsi. Method ini menerima masukan pdf_extract dengan tipe data kelas *SkripsiExtract*. Bagian yang diperiksa pada *checker* ini adalah bab 1 hingga 6.

8. Kelas PS09_SubChapterChecker
   Kelas ini bertanggungjawab untuk memeriksa subbab yang ada dalam sebuah bab. Berikut adalah *method* yang terdapat pada kelas ini:

   - errorChecking($pdf_extract)
     Method ini berfungsi untuk memeriksa kesalahan pada dokumen skripsi. Method ini menerima masukan pdf_extract dengan tipe data kelas *SkripsiExtract*. Bagian yang diperiksa pada *checker* ini adalah bab 1 hingga 6.

9. Kelas VAN03_SubjectProunounChecker
   Kelas ini bertanggungjawab untuk memeriksa penggunaan kata ganti orang pada dokumen skripsi. Berikut adalah *method* yang terdapat pada kelas ini:

   - errorChecking($pdf_extract)
     Method ini berfungsi untuk memeriksa kesalahan pada dokumen skripsi. Method ini menerima masukan pdf_extract dengan tipe data kelas *SkripsiExtract*. Bagian yang diperiksa pada *checker* ini adalah bab 1 hingga 6.

10. Kelas SkripsiExtract
    Kelas ini bertanggungjawab untuk melakukan ekstak file PDF skripi. Berikut adalah atribut yang terdapat pada kelas ini:

    - file
      Atribut ini berfungsi untuk menyimpan nama file dari dokumen skripsi.
    - parser
      Atribut ini berfungsi sebagai instansiasi kelas Parser dari *Library PDF Parser*.
    - cover
      Atribut ini berfungsi untuk menyimpan hasil ekstrak dari halaman cover bahasa Indonesia dan bahasa Inggris.
    - abstractPage
      Atribut ini berfungsi untuk menyimpan hasil ekstrak dari halaman abstrak bahasa Indonesia dan bahasa Inggris.
    - chapterPage
      Atribut ini berfungsi untuk menyimpan hasil ekstrak mulai dari bab 1 hingga bab 6.
    - listPage
      Atribut ini berfungsi untuk menyimpan hasil ekstrak pada halaman daftar isi, daftar tabel, daftar gambar dan daftar referensi.

    Kelas ini juga memiliki *method* yang berkaitan dengan proses ekstrak dokumen skripsi dan penyimpanan hasil ekstraknya. Berikut adalah *method* yang terdapat pada kelas ini:

    - extractFromPDF($input)
      Method ini berfungsi untuk melakukan proses ekstrak file PDF skripsi. Hasil dari ekstrak tersebut akan disimpan ke dalam beberapa bagian, seperti cover, halaman abstrak, dan sebagainya.
    - getCoverPage()
      Method ini berfungsi untuk mendapatkan halaman cover skripsi dalam bahasa Indonesia dan bahasa Inggris.
    - getAbstractPage()
      Method ini berfungsi untuk mendapatkan halaman abstrak dalam bahasa Indonesia dan bahasa Inggris.
    - getListPage()
      Method ini berfungsi untuk mendapatkan halaman daftar isi, daftar gambar, daftar tabel dan daftar referensi.
    - getContent()
      Method ini berfungsi untuk mendapatkan halaman konten skripsi dari bab 1 sampai bab 6.

11. Kelas Main
    Kelas ini digunakan untuk menjalankan perangkat lunak. Kelas ini tidak memiliki atribut dan

method, hanya terdiri dari beberapa baris kode yang memanggil *method* untuk mengekstrak dan memeriksa file PDF skripsi.

4.2 Perancangan Perangkat Lunak

Pada bagian ini akan dijelaskan perancangan untuk mengekstrak dokumen skripsi dan pola pengecekan kesalahan yang akan digunakan pada perangkat lunak.

4.2.1 Algoritma untuk Mengekstrak Dokumen

Dokumen skripsi yang akan diperiksa dalam perangkat lunak merupakan file yang memiliki ekstensi *PDF*. Sebelum dilakukan pemeriksaan kesalahan, dokumen tersebut perlu diekstrak terlebih dulu. Pada skripsi ini, aplikasi akan menggunakan *library PDFParser* untuk mengekstrak dokumen skripsi. Pada *library* tersebut, terdapat 2 metode yang dapat digunakan untuk mengekstrak dokumen.

Listing 4.1: Potongan kode untuk mengesktrak seluruh halaman dokumen

```
1 include 'vendor/autoload.php';
2
3 $parser = new \Smalot\PdfParser\Parser();
4 $pdf    = $parser->parseFile('document.pdf');
5 $text   = $pdf->getText();
```

Listing 4.2: Potongan kode untuk mengesktrak halaman dokumen secara spesifik

```
1 include 'vendor/autoload.php';
2
3 $parser = new \Smalot\PdfParser\Parser();
4 $pdf    = $parser->parseFile('document.pdf');
5 $pages  = $pdf->getPages();
```

Listing 4.1 dan listing 4.2 merupakan potongan kode yang terdapat pada dokumentasi *library PDF Parser*. Kode dari kedua listing tersebut hampir mirip, yang menjadi pembeda adalah pada baris ke-5. Listing 4.1 menggunakan *method getText()* dan hasil dari ekstraknya disimpan menjadi sebuah kesatuan dari halaman awal hingga halaman akhir. Listing 4.2 menggunakan *method getPages()* dan hasil dari ekstraknya disimpan dalam sebuah array String sejumlah halaman yang ada pada dokumen.

Pada skripsi ini akan digunakan metode yang mengesktrak halaman-halaman secara spesifik. Hasil ekstrak akan disimpan dalam beberapa bagian, berdasarkan konten-konten yang ada. Konten-konten yang dimaksud seperti, halaman cover, abstrak, daftar isi, daftar tabel, daftar gambar dan seterusnya. Ada beberapa alasan yang mendorong untuk melakukan pemisahan konten-konten tersebut, salah satunya yaitu agar pemeriksa dapat memeriksa bagian-bagian spesifik yang menjadi tanggungjawab dari pemeriksa tersebut. Contohnya, untuk memeriksa kelengkapan data skripsi tidak perlu memeriksa seluruh isi dari dokumen. Hal yang paling dibutuhkan adalah halaman covernya saja, sehingga proses pemeriksaan menjadi lebih efektif.

4.2.2 Pola Pengecekan Kesalahan

Hasil survei kesalahan-kesalahan dalam penulisan dokumen skripsi sudah disaring dan akan diimplementasikan dalam perangkat lunak. Pengecekan kesalahan tersebut akan dilakukan dengan menggunakan teknik *pattern matching regex*. Hasil ekstrak dari PDF skripsi akan dipotong dengan delimiter karakter titik dan spasi 1 buah ke dalam sebuah array. Hal ini dilakukan agar pemeriksaan dapat lebih mudah dilakukan apabila teks sudah dipecah menjadi sebuah kalimat utuh. Berikut adalah rincian dari hasil survei yang dipilih beserta penyelesaiannya:

1. Penulisan kata (PS-01)
   Untuk mendeteksi kesalahan penulisan kata, akan digunakan ekstensi kamus bahasa Indonesia *LibreOffice*. Dengan digunakan kamus tersebut, kesalahan penulisan suatu kata dapat diminimalisir. Pada skripsi ini, pemeriksaan kata-kata yang menggunakan imbuhan tidak ditangani pada masalah ini. Terdapat kode yang harus diterjemahkan terlebih dahulu untuk dapat mengetahui imbuhan yang dapat digunakan dalam sebuah kata. Pola pengecekan yang digunakan akan dijelaskan pada pseudocode 1.

Algorithm 1 Typo checker function

```
function ExtractFromPDF($pdf_exctract)
    Input: An object from the SkripsiExtract class to call the getter method
    Output: An array containing error reports
    temp ← Split dictionary based on newline
    dictionary ← Array for store dictionary
    for i = 0 to size of temp do
        Remove affix code from each index dictionary array
    word ← Call getter from class SkripsiExtract and split with non word character
    typos ← Array for store typos
    for each word in array do
        row ← Add index with 1
        if value not contain in dictionary and value not contain in typos then
            Fill typos with value
    result ← Array for store all errors report
    if size of typos greater than 0 then
        Add all errors report into result
    return result
```

Pada pseudocode 1, pola akan memeriksa setiap kata yang ada pada indeks array. Namun untuk pemeriksaan kata ini hanya untuk kata yang menggunakan bahasa Indonesia saja. Pola akan mencocokan kata yang akan diperiksa dengan kamus bahasa Indonesia *LibreOffice*. Berikut adalah contoh dari kesalahan dan laporan yang dikeluarkan:

- Contoh kesalahan
  Istilah *regex* berasal dari toeri matematika dan komputer sains, yang mencerminkan sifat ekspresi dalam matematika yang disebut keteraturan
- Laporan kesalahan
  Pada baris ini ditemukan penulisan kata yang tidak sesuai dengan kamus

2. Pemberian spasi sebelum dan sesudah tanda baca (PS-03)
Kesalahan ini akan terdeteksi apabila tidak ada karakter spasi sebelum ataupun sesudah tanda baca. Pola pengecekan yang digunakan akan dijelaskan pada *pseudocode* 2.

Algorithm 2 Space checker function

```
1:  function ExtractFromPDF($pdf_exctract)
2:      Input: An object from the SkripsiExtract class to call the getter method
3:      Output: An array containing error reports
4:      result ← Array for store all errors report
5:      sentence ← Call getter from class SkripsiExtract
6:      for each sentence in sentence do
7:          row ← Add index with 1
8:          pattern ← Define regex /([A-Za-z0-9]*[,.!?][A-Za-z0-9])/
9:          if pattern contain in array then
10:             fill result with errors report
11:     return result
```

Pada pseudocode 2, pola akan memeriksa setiap kalimat yang ada pada indeks array. Kesalahan akan terdeteksi apabila setelah tanda baca titik, koma, seru atau tanya tidak terdapat spasi yang memisahkan tanda baca dengan kata baru. Berikut adalah contoh dari kesalahan dan laporan yang dikeluarkan:

- Contoh kesalahan
  Aplikasi ini dijalankan melalui melalui terminal command Windows.laporan dari hasil kesalahan tersebut akan ditampilkan melalui terminal command Windows.
- Laporan kesalahan
  Perhatikan spasi sebelum atau setelah tanda baca.

3. Awal kalimat tidak menggunakan huruf kapital (PS-05)
Kesalahan ini akan terdeteksi apabila setelah tanda baca pada akhir kalimat, karakter pertama setelah spasi menggunakan huruf kecil. Pola pengecekan yang digunakan akan dijelaskan pada *pseudocode* 3.

Algorithm 3 Capital letter checker function

```
1:  function ExtractFromPDF($pdf_exctract)
2:      Input: An object from the SkripsiExtract class to call the getter method
3:      Output: An array containing error reports
4:      result ← Array for store all errors report
5:      sentence ← Call getter from class SkripsiExtract
6:      for each sentence in sentence do
7:          row ← Add index with 1
8:          pattern ← Define regex /^(\s[a-z][a-z0-9].*)/
9:          if pattern contain in array then
10:             fill result with errors report
11:     return result
```

Pada pseudocode 3, pola akan memeriksa setiap kalimat yang ada pada indeks array. Kesalahan akan terdeteksi apabila karakter pertama dalam sebuah kalimat menggunakan huruf kecil. Berikut adalah contoh dari kesalahan dan laporan yang dikeluarkan:

- Contoh kesalahan
  aplikasi sederhana ini, dapat dimanfaatkan oleh mahasiswa Informatika Unpar secara mandiri
- Laporan kesalahan
  Huruf pertama pada kalimat ini tidak menggunakan huruf kapital

4. Jumlah subbab dalam 1 bab tidak boleh hanya 1 (PS-09)
   Kesalahan ini dapat diselesaikan dengan mencari jumlah subbab yang ada dalam sebuah bab. Pola pengecekan yang digunakan akan dijelaskan pada *pseudocode* 4.

Algorithm 4 Subchapter checker function

```
1: function ExtractFromPDF($pdf_exctract)
2:     Input: An object from the SkripsiExtract class to call the getter method
3:     Output: An array containing error reports
4:     result ← Array for store all errors report
5:     sentence ← Call getter from class SkripsiExtract
6:     for each sentence in sentence do
7:         row ← Add index with 1
8:         pattern ← Define regex /^/
9:         if pattern contain in array then
10:            fill result with errors report
11:    return result
```

Pada pseudocode 4, pola akan memeriksa setiap kalimat yang ada pada indeks array. Kesalahan akan terdeteksi apabila karakter pertama dalam sebuah kalimat menggunakan huruf kecil. Berikut adalah contoh dari kesalahan dan laporan yang dikeluarkan:

- Contoh kesalahan
  Bab 4
  PERANCANGAN

  4.1 Perancangan Kelas
- Laporan kesalahan
  Pada bab ini hanya terdapat 1 subbab, lebih baik tidak perlu menggunakan subbab

5. Kalimat pengantar untuk setiap subbab (KAL-02)
   Kesalahan ini dapat dideteksi dengan melihat ada atau tidaknya kalimat setelah subbab dibuat. Pola pengecekan yang digunakan akan dijelaskan pada *pseudocode* 5.

Algorithm 5 Preface checker function

```
function ExtractFromPDF($pdf_exctract)
    Input: An object from the SkripsiExtract class to call the getter method
    Output: An array containing error reports
    result ← Array for store all errors report
    sentence ← Call getter from class SkripsiExtract
    for each sentence in sentence do
        row ← Add index with 1
        pattern ← Define regex /^[A-Z0-9][A-Za-z] /
        if pattern contain in array then
            fill result with errors report
    return result
```

Pada pseudocode 5, pola akan memeriksa setiap kalimat yang ada pada indeks array. Kesalahan akan terdeteksi apabila tidak terdapat kalimat yang mengawali bab tersebut. Berikut adalah contoh dari kesalahan dan laporan yang dikeluarkan:

- Contoh kesalahan
  Bab 4
  PERANCANGAN

  4.1 Perancangan Kelas
  4.2 Perancangan Algoritma
- Laporan kesalahan
  Berilah kata pengantar untuk bab atau subbab

6. Kelengkapan data skripsi (KAL-03)
   Kesalahan ini dapat terlihat pada halaman cover skripsi. Mahasiswa yang belum mengisi data skripsi, pada file PDFnya akan ditampilkan tulisan template pada data skripsinya.Pola pengecekan yang digunakan akan dijelaskan pada *pseudocode* 6.

Algorithm 6 Thesis data checker function

```
function ExtractFromPDF($pdf_exctract)
    Input: An object from the SkripsiExtract class to call the getter method
    Output: An array containing error reports
    result ← Array for store all errors report
    sentence ← Call getter from class SkripsiExtract
    for each sentence in sentence do
        row ← Add index with 1
        pattern ← Define regex /JUDUL BAHASA INDONESIA|JUDUL BAHASA ING-
    GRIS|Nama Lengkap|10 digit NPM UNPAR|tahun/
        if pattern contain in array then
            fill result with errors report
    return result
```

Pada pseudocode 6, pola akan memeriksa halaman cover bahasa Indonesia dan bahasa Inggris.

Kesalahan akan terdeteksi pada halaman cover masih terdapat kalimat yang terdapat dalam template. Berikut adalah contoh dari kesalahan dan laporan yang dikeluarkan:

- Contoh kesalahan
  «SKRIPSI/TUGAS AKHIR»
  «Judul Bahasa Indonesia»
  Marcell Trixie Alexander
  «10 digit NPM UNPAR»
- Laporan kesalahan
  Ada data skripsi yang belum dilengkapi, data dapat diisi pada file data.tex

7. Penggunaan kata ganti orang (VAN-03)
   Kesalahan ini dapat diatasi dengan memasukan kata-kata yang termasuk dalam kata ganti orang menjadi kata-kata yang tidak dapat digunakan. Pola pengecekan yang digunakan akan dijelaskan pada *pseudocode* 7.

Algorithm 7 Subject Pronoun checker function

```
1: function ExtractFromPDF($pdf_exctract)
2:     Input: An object from the SkripsiExtract class to call the getter method
3:     Output: An array containing error reports
4:     result ← Array for store all errors report
5:     sentence ← Split PDF parsing results based on dots and space
6:     for each sentence in sentence do
7:         row ← Add index with 1
8:         pattern ← Define regex /saya|kamu|dia/i
9:         if pattern contain in array then
10:            fill result with errors report
11:    return result
```

Pada pseudocode 7, pola akan memeriksa setiap kalimat yang ada pada indeks array. Kesalahan akan terdeteksi pada kalimat terdapat kata ganti orang. Berikut adalah contoh dari kesalahan dan laporan yang dikeluarkan:

- Contoh kesalahan
  Saya akan membuat aplikasi ini menggunakan bahasa pemrograman PHP
- Laporan kesalahan
  Kalimat tersebut mengandung kata ganti orang

8. Penulisan daftar referensi (NAT-01)
   Kesalahan dalam penulisan daftar referensi dapat dilihat dengan munculnya tanda ”[?]”, yang menandakan bahwa referensi tersebut tidak dirujuk dengan baik. Pola pengecekan yang digunakan akan dijelaskan pada *pseudocode* 8.

Algorithm 8 Reference checker function

```
function ExtractFromPDF($pdf_exctract)
    Input: An object from the SkripsiExtract class to call the getter method
    Output: An array containing error reports
    result ← Array for store all errors report
    sentence ← Call getter from class SkripsiExtract
    for each sentence in sentence do
        row ← Add index with 1
        pattern ← Define regex /[?]/
        if pattern contain in array then
            fill result with errors report
    return result
```

Pada pseudocode 8, pola akan memeriksa setiap kalimat yang ada pada indeks array. Kesalahan akan terdeteksi kalimat terdapat simbol "[?]", yang menandakan referensi tidak dirujuk dengan baik. Berikut adalah contoh dari kesalahan dan laporan yang dikeluarkan:

- Contoh kesalahan
  Regular expression [?] adalah jenis pola teks tertentu yang dapat digunakan pada banyak aplikasi modern dan bahasa pemrograman
- Laporan kesalahan
  Referensi tidak dirujuk dengan baik, lakukan perintah PDFLatex->BibTex->PDFLatex->PDFLatex untuk memperbaikinya

BAB 5

IMPLEMENTASI DAN PENGUJIAN

Pada bab ini dibahas mengenai implementasi perangkat lunak dan pengujian yang dilakukan terhadap perangkat lunak tersebut. Lingkungan implementasi, yang meliputi perangkat keras dan perangkat lunak, serta hasil implementasi akan dijelaskan pada bab ini. Selain Pengujian yang dilakukan pada skripsi ini, yang meliputi pengujian fungsional dan eksperimental akan dijelaskan pada bab ini.

5.1 Implementasi

Pada bagian ini akan dijelaskan mengenai lingkungan yang digunakan untuk membangun perangkat lunak beserta hasil implementasinya.

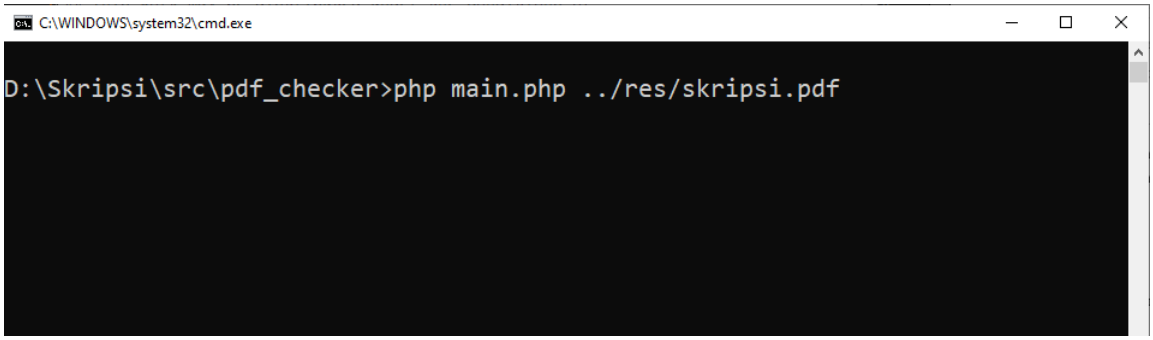
5.1.1 Lingkungan Implementasi

Berikut spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam pembangunan pada skripsi ini:

1. Spesifikasi Perangkat Keras
    - Perangkat: Laptop
    - Processor: AMD Bristol Ridge Quad Core FX-9830P 3GHz
    - RAM: 8GB
    - GPU: Radeon RX 460
    - Storage: Harddisk 1TB
2. Spesifikasi Perangkat Lunak
    - Sistem Operasi Windows 10 64-bit
    - PHP 7.3.5 (cli)
    - Composer versi 1.8.5
    - Sublime Text versi 3.2.1

5.1.2 Hasil Implementasi

Perangkat lunak dikembangkan menggunakan bahasa pemrograman PHP. Perangkat lunak yang dibuat tidak menggunakan *Graphical User Interface*, melainkan berbasis *Command Line Interface*. Perangkat lunak akan menerima input berupa file PDF skripsi yang disimpan pada folder yang telah disediakan, dan mengeluarkan laporan kesalahan pada terminal.



Gambar 5.1: Perintah yang digunakan untuk menjalankan perangkat lunak

Gambar 5.1 merupakan perintah yang perlu dituliskan pada terminal, untuk menjalankan perangkat lunak. Kelas Main menjadi kelas yang digunakan untuk menjalankan seluruh proses yang berjalan dalam perangkat lunak. File PDF skripsi yang akan diperiksa harus berada di folder yang telah disediakan, yaitu pada folder Skripsi\src\res.

5.2 Pengujian

Pada bagian ini akan dijelaskan mengenai pengujian yang dilakukan pada perangkat lunak. Akan dilakukan 2 bentuk pengujian pada skripsi ini, yaitu pengujian fungsional dan pengujian eksperimental.

5.2.1 Pengujian Fungsional

Pengujian fungsional bertujuan untuk menguji fungsionalitas perangkat lunak. Perangkat lunak memiliki 8 fitur yang telah diimplementasikan. Fitur-fitur tersebut akan diuji untuk melihat kebenaran dan kesesuaian fitur tersebut dengan yang diharapkan.

5.2.2 Pengujian Eksperimental

BAB 6

KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan dari pembangunan aplikasi dan saran untuk pengembangan aplikasi ini.

6.1 Kesimpulan

6.2 Saran

DAFTAR REFERENSI

[1] Goyvaerts, J. dan Levithan, S. (2012) *Regular Expressions Cookbook*, 2nd edition. O?Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

[2] PHP Perl compatible regular expression. https://www.php.net/manual/en/book.pcre.php. 24 Juli 2019.

[3] Parser, P. Pdf parser. https://www.pdfparser.org/. 7 Mei 2019.

[4] LibreOffice (2018) Getting started with libreoffice 6.0. https://documentation.libreoffice.org/en/english-documentation/writer/. 24 Juli 2019.

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

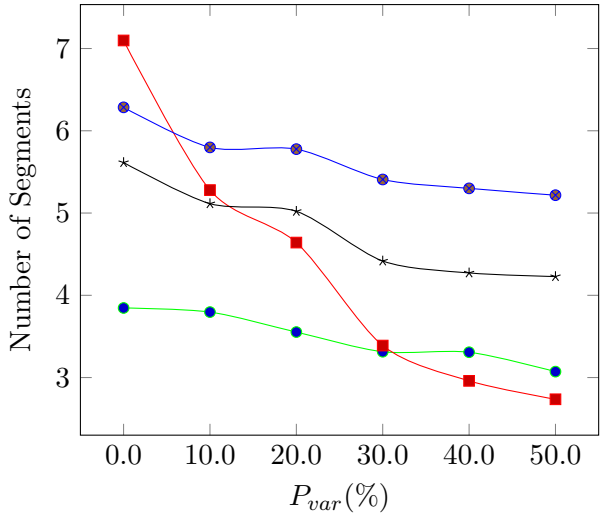
Listing A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

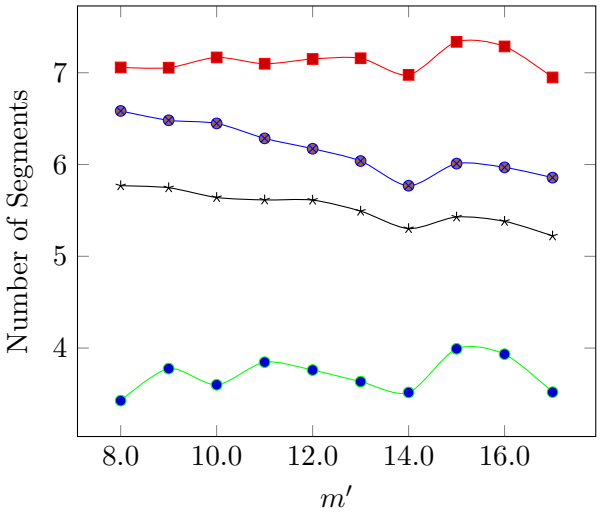
LAMPIRAN B

HASIL EKSPERIMEN

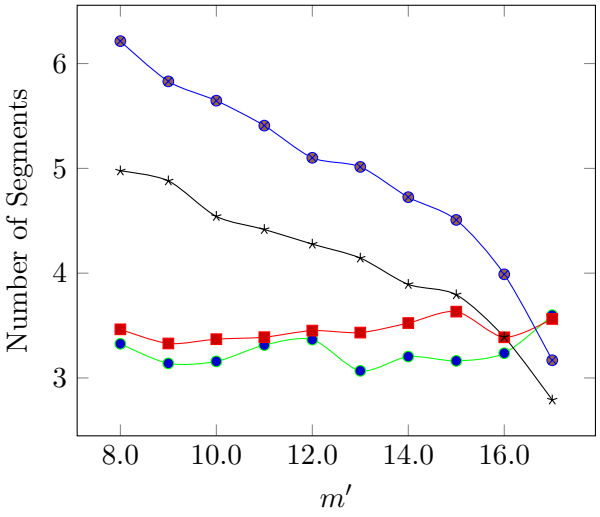
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



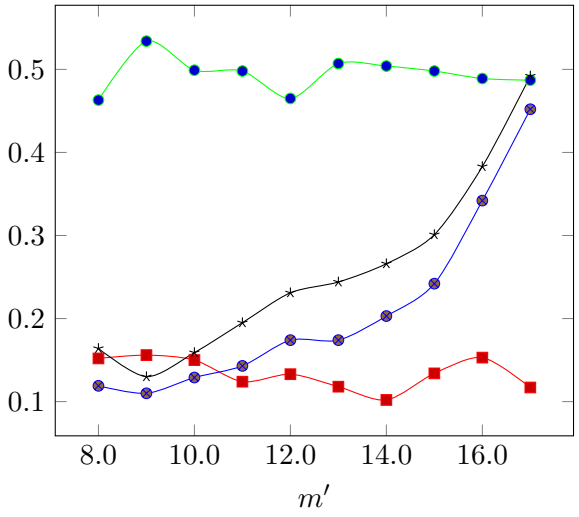
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4