

SKRIPSI

PENGEMBANGAN APLIKASI DESKTOP PEMERIKSAAN
TAUTAN RUSAK PADA SITUS WEB



Ade Rimbo Spencher

NPM: 6182001060

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2025

UNDERGRADUATE THESIS

**DEVELOPMENT OF A DESKTOP APPLICATION FOR
BROKEN LINK CHECKING IN WEBSITES**



Ade Rimbo Spencher

NPM: 6182001060

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2025**

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 ISI	ix
DAFTAR GAMBAR	xi
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	1
1.3 Tujuan	1
1.4 Batasan Masalah	1
1.5 Metodologi	1
1.6 Sistematika Pembahasan	1
2 LANDASAN TEORI	3
2.1 HTTP [1]	3
2.2 URI [2]	3
2.3 HTML [3]	3
2.4 Web Crawling [4]	3
2.5 JavaFX [5]	3
2.6 Jsoup	3
2.7 Java HTTP Client API	3
3 ANALISIS	5
3.1 Analisis Masalah	5
3.2 Analisis Sistem Serupa	5
3.3 Analisis Kebutuhan	5
3.4 Analisis Teknologi yang Digunakan	5
4 PERANCANGAN	7
4.1 Perancangan Kelas	7
4.2 Perancangan Alur	11
4.3 Perancangan Antarmuka	13
5 IMPLEMENTASI DAN PENGUJIAN	17
5.1 Implementasi	17
5.2 Pengujian	17
6 KESIMPULAN DAN SARAN	19
6.1 Kesimpulan	19
6.2 Saran	19
DAFTAR REFERENSI	21
A KODE PROGRAM	23

DAFTAR GAMBAR

4.1	Diagram kelas aplikasi	11
4.2	Sequence diagram proses pemeriksaan tautan	13
4.3	Rancangan Halaman Utama	15
4.4	Rancangan Jendela Detail Broken Link	16

¹

BAB 1

²

PENDAHULUAN

³ **1.1 Latar Belakang**

⁴ **1.2 Rumusan Masalah**

⁵ **1.3 Tujuan**

⁶ **1.4 Batasan Masalah**

⁷ **1.5 Metodologi**

⁸ **1.6 Sistematika Pembahasan**

¹

BAB 2

²

LANDASAN TEORI

- ³ Bab ini membahas teori dan teknologi yang mendasari pengembangan perangkat lunak pemeriksa tautan rusak, mencakup protokol HTTP, URI, konsep dasar situs web, *web crawling*, serta pustaka dan teknologi yang digunakan seperti Jsoup, Java HTTP Client API, dan JavaFX.

⁶ **2.1 HTTP [1]**

⁷ **2.2 URI [2]**

⁸ **2.3 HTML [3]**

⁹ **2.4 Web Crawling [4]**

¹⁰ **2.5 JavaFX [5]**

¹¹ **2.6 Jsoup**

¹² **2.7 Java HTTP Client API**

¹

BAB 3

²

ANALISIS

³ Bab ini membahas analisis yang menjadi dasar perancangan dan pengembangan aplikasi desktop
⁴ pemeriksa tautan rusak pada situs web. Analisis mencakup permasalahan utama yang melatarbelakangi penelitian, tinjauan terhadap sistem serupa, serta perumusan kebutuhan sistem baik fungsional
⁵ maupun non-fungsional. Selain itu, dibahas pula analisis terhadap teknologi yang digunakan untuk
⁶ mendukung implementasi aplikasi.
⁷

⁸ **3.1 Analisis Masalah**

⁹ **3.2 Analisis Sistem Serupa**

¹⁰ **3.3 Analisis Kebutuhan**

¹¹ **3.4 Analisis Teknologi yang Digunakan**

1

BAB 4

2

PERANCANGAN

3 Bab ini membahas perancangan aplikasi desktop pemeriksa tautan rusak pada situs web berdasarkan
4 hasil analisis yang telah dilakukan pada Bab 3. Perancangan mencakup rancangan kelas, perancangan
5 alur dan perancangan antarmuka. Dengan adanya perancangan ini, implementasi dapat dilakukan
6 secara lebih terarah dan sesuai dengan kebutuhan yang telah ditentukan.

7 **4.1 Perancangan Kelas**

8 Struktur kelas pada aplikasi *Broken Link Checker* dibagi ke dalam tiga *package* utama, yaitu *Web*
9 *Crawling*, *URL Fetcher*, dan *GUI Manager*. *Package Web Crawling* berisi kelas-kelas inti yang
10 mengatur proses *crawling*, pengelolaan *frontier*, serta representasi tautan. *Package URL Fetcher*
11 menyediakan mekanisme untuk mengambil konten dari situs web menggunakan pustaka eksternal
12 seperti Jsoup dan HttpClient. Sementara itu, *package GUI Manager* menangani pengelolaan
13 antarmuka pengguna dengan JavaFX, termasuk kontrol proses pemeriksaan dan penyajian hasil.
14 Diagram kelas untuk aplikasi ini dapat dilihat pada Gambar 4.1.

15 Penjelasan setiap kelas pada diagram adalah sebagai berikut:

16 **1. Kelas Link**

17 Kelas *Link* merupakan kelas dasar yang merepresentasikan sebuah tautan dalam sistem. Kelas
18 ini menjadi induk bagi *WebpageLink* dan *BrokenLink*, sehingga relasinya berupa pewarisan.

- 19 • **url** : Atribut ini bertipe data *String* dan digunakan untuk menyimpan alamat dari
20 tautan yang diperiksa.
- 21 • **statusCode** : Atribut ini bertipe data *int* dan berfungsi untuk menyimpan kode status
22 HTTP hasil pemeriksaan tautan.
- 23 • **accessTime** : Atribut ini bertipe data *Instant* dan merekam waktu terakhir kali tautan
24 tersebut diakses.

25 **2. Kelas WebpageLink**

26 Kelas *WebpageLink* merepresentasikan tautan yang termasuk dalam kategori halaman web
27 dengan *host* yang sama seperti URL awal. Kelas ini mewarisi *Link* dan memiliki relasi
28 *many-to-many* dengan *BrokenLink*.

- 29 • **brokenLinks** : Atribut ini bertipe data *Map<BrokenLink, String>* dan digunakan untuk
30 menyimpan daftar tautan rusak yang ditemukan di halaman beserta informasi anchor
31 text-nya.

32 **3. Kelas BrokenLink**

33 Kelas *BrokenLink* digunakan untuk menyimpan data tautan yang gagal diakses atau meng-

hasilkan status *error*. Kelas ini mewarisi `Link` dan memiliki relasi *many-to-many* dengan `WebpageLink`.

- `webpageLinks` : Atribut ini bertipe data `Map<WebpageLink, String>` dan digunakan untuk mencatat daftar halaman tempat tautan rusak tersebut ditemukan beserta lokasi anchor text-nya.

4. Kelas Crawler

Kelas `Crawler` merupakan inti dari proses *web crawling* dan bertanggung jawab mengelola antrean URL, melakukan pengambilan halaman, serta memeriksa status tautan. Kelas ini berhubungan langsung dengan `Frontier`, `WebpageLink`, dan `BrokenLink`, serta memiliki dependensi terhadap `Jsoup` dan `HttpClient`.

- `rootHost` : Atribut ini bertipe data `String` dan digunakan untuk menyimpan host dari URL awal (seed URL).
- `frontier` : Atribut ini bertipe data `Frontier` dan berfungsi sebagai antrean URL yang akan diproses.
- `repository` : Atribut ini bertipe data `Set<String>` dan digunakan untuk menyimpan daftar URL yang sudah pernah dikunjungi agar tidak diperiksa dua kali.
- `USER_AGENT` : Atribut statis ini bertipe data `String` dan digunakan untuk mendefinisikan identitas permintaan HTTP yang dikirim oleh aplikasi.
- `TIMEOUT` : Atribut statis ini bertipe data `int` dan menentukan batas waktu maksimum koneksi HTTP.
- `HTTP_CLIENT` : Atribut statis ini bertipe data `HttpClient` dan digunakan sebagai klien bawaan Java untuk mengirim permintaan HTTP.
- `Crawler` : Metode ini merupakan konstruktor yang menerima parameter `seedUrl` bertipe data `String` dan digunakan untuk menginisialisasi proses *crawling*.
- `startCrawling` : Metode ini menerima parameter `streamBrokenLink` bertipe `Consumer<BrokenLink>` dan tidak mengembalikan nilai. Metode ini berfungsi untuk memulai proses *crawling* dan mengirim hasilnya secara *streaming*.
- `extractLinks` : Metode ini menerima parameter `doc` bertipe `Document` dan mengembalikan daftar objek `BrokenLink`. Metode ini digunakan untuk mengekstrak seluruh tautan dari halaman HTML yang diambil.
- `normalizeUrl` : Metode ini menerima parameter `url` bertipe `String` dan mengembalikan hasil normalisasi dalam bentuk `String`. Metode ini digunakan untuk memastikan URL dalam format yang konsisten.
- `fetchUrl` : Metode ini menerima parameter `url` bertipe `String` dan mengembalikan nilai `int` yang merupakan kode status HTTP hasil pemeriksaan. Metode ini berfungsi untuk melakukan pemeriksaan tautan eksternal atau tautan umum menggunakan Java `HttpClient`.
- `isPotentialWebpage` : Metode ini menerima parameter `url` bertipe `String` dan mengembalikan nilai `boolean`. Metode ini digunakan untuk menentukan apakah suatu URL berpotensi merupakan halaman web.

5. Kelas Frontier

Kelas `Frontier` berfungsi sebagai struktur data antrean (*queue*) yang menyimpan URL yang

akan diproses berikutnya. Kelas ini berelasi erat dengan **Crawler** yang mengelola antrean selama proses *crawling* berlangsung.

- **urls** : Atribut ini bertipe data *Queue<String>* dan digunakan untuk menampung daftar URL yang menunggu diproses.
- **enqueue** : Metode ini menerima parameter **url** bertipe *String* dan berfungsi untuk menambahkan URL baru ke dalam antrean.
- **dequeue** : Metode ini tidak menerima parameter dan mengembalikan nilai *String* berupa URL paling depan dari antrean.
- **isEmpty** : Metode ini tidak menerima parameter dan mengembalikan nilai *boolean* untuk menunjukkan apakah antrean kosong atau tidak.

6. Kelas Jsoup dan HttpClient

Kedua kelas ini merupakan representasi dari pustaka eksternal yang digunakan untuk melakukan pengambilan data web dan digunakan sebagai dependensi oleh kelas **Crawler**.

- **Jsoup** : Kelas ini digunakan untuk mengakses dan mengurai dokumen HTML dari halaman same-host dengan memanfaatkan parser HTML toleran kesalahan.
- **HttpClient** : Kelas ini digunakan untuk melakukan permintaan **HEAD** atau **GET** terhadap tautan umum atau eksternal untuk memeriksa status HTTP-nya.

7. Kelas Controller

Kelas **Controller** berfungsi sebagai penghubung antara logika aplikasi dengan antarmuka pengguna yang dibangun menggunakan JavaFX. Kelas ini berinteraksi dengan **Crawler** untuk menginisiasi dan menerima hasil proses pemeriksaan tautan.

- **seedUrlField** : Atribut ini bertipe *TextField* dan digunakan untuk menampung input URL dari pengguna.
- **progressLabel** : Atribut ini bertipe *Label* dan menampilkan status kemajuan proses pemeriksaan.
- **totalLinksLabel** : Atribut ini bertipe *Label* dan menampilkan jumlah total tautan yang diperiksa.
- **webpagesLabel** : Atribut ini bertipe *Label* dan menampilkan jumlah halaman yang berhasil di-crawl.
- **brokenLinksLabel** : Atribut ini bertipe *Label* dan menampilkan jumlah tautan rusak yang ditemukan.
- **resultsTable** : Atribut ini bertipe *TableView<BrokenLink>* dan menampilkan hasil daftar tautan rusak.
- **statusColumn** : Atribut ini bertipe *TableColumn<BrokenLink, String>* dan digunakan untuk menampilkan kolom status pada tabel hasil.
- **urlColumn** : Atribut ini bertipe *TableColumn<BrokenLink, String>* dan digunakan untuk menampilkan kolom URL tautan pada tabel hasil.
- **initialize** : Metode ini tidak menerima parameter dan berfungsi untuk melakukan inisialisasi awal saat antarmuka dimuat.
- **onStartClick** : Metode ini tidak menerima parameter dan digunakan untuk menangani event ketika pengguna menekan tombol *Start*, kemudian memulai proses pemeriksaan tautan.

- 1 • **onStopClick** : Metode ini tidak menerima parameter dan digunakan untuk menghentikan
- 2 proses pemeriksaan yang sedang berlangsung.
- 3 • **onExportClick** : Metode ini tidak menerima parameter dan digunakan untuk mengekspor
- 4 hasil pemeriksaan ke dalam berkas Excel.

5 8. Kelas Application

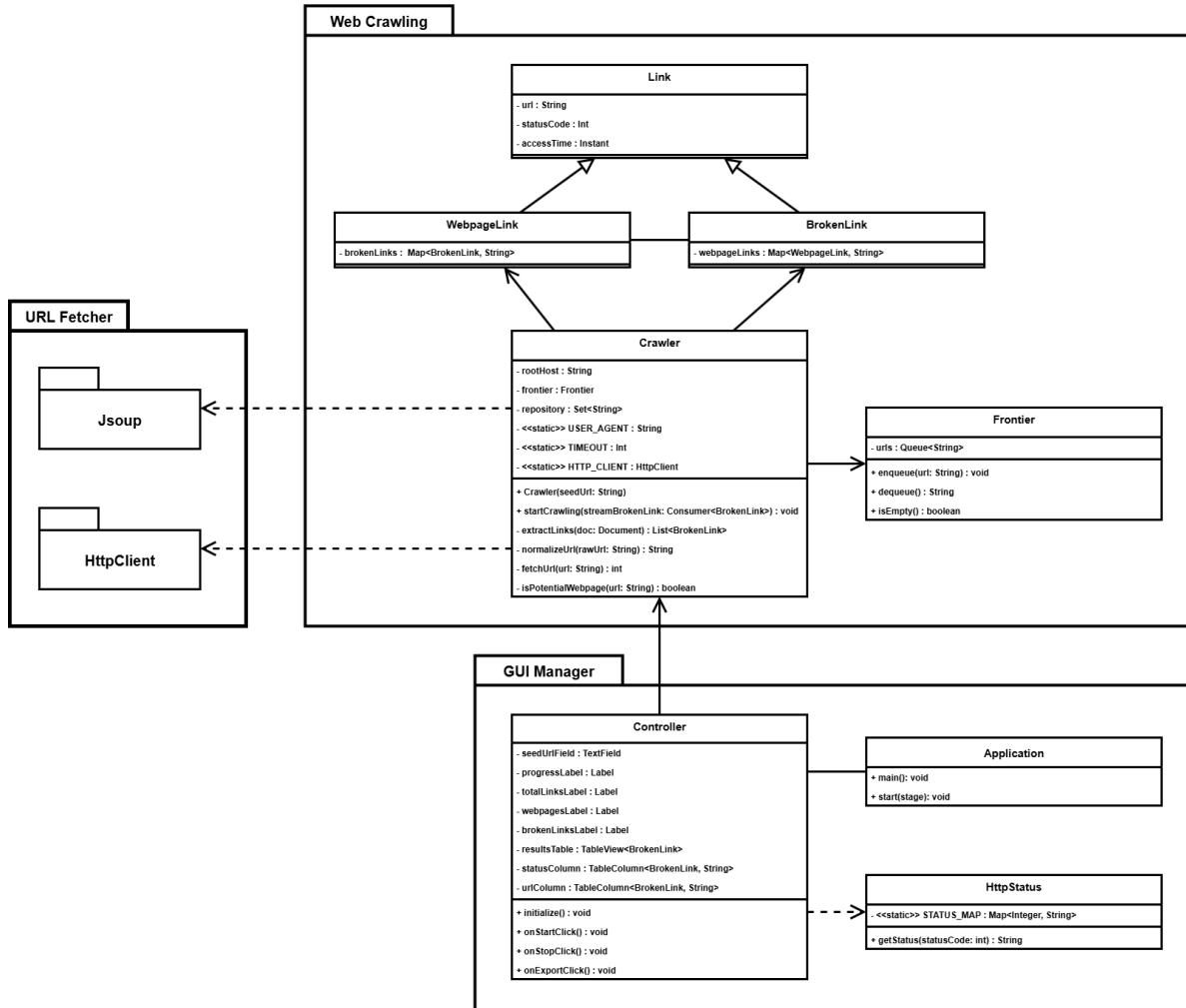
6 Kelas **Application** merupakan titik masuk (*entry point*) dari aplikasi JavaFX. Kelas ini
7 berhubungan dengan **Controller** untuk memuat dan menginisialisasi GUI.

- 8 • **main** : Metode ini tidak menerima parameter dan digunakan untuk menjalankan aplikasi.
- 9 • **start** : Metode ini menerima parameter **stage** bertipe *Stage* dan berfungsi untuk
10 memuat serta menampilkan antarmuka pengguna utama.

11 9. Kelas HttpStatus

12 Kelas **HttpStatus** berfungsi sebagai kelas utilitas yang memetakan kode status HTTP ke
13 *reason phrase*. Kelas ini digunakan oleh **Controller** untuk menampilkan hasil status HTTP
14 pada GUI.

- 15 • **STATUS_MAP** : Atribut ini bertipe *Map<Integer, String>* dan digunakan untuk menyimpan
16 pasangan antara kode status HTTP dengan keterangan resminya.
- 17 • **getStatusCode** : Metode ini menerima parameter **statusCode** bertipe *int* dan mengembalikan
18 nilai *String* berupa keterangan lengkap dari kode status tersebut.



Gambar 4.1: Diagram kelas aplikasi

4.2 Perancangan Alur

Perancangan alur menggambarkan interaksi antarobjek pada sistem *Broken Link Checker* selama proses pemeriksaan tautan berlangsung. Interaksi ini divisualisasikan dalam bentuk *sequence diagram* yang menunjukkan urutan pesan antarobjek sejak pengguna memulai pemeriksaan hingga proses selesai tanpa interupsi. Diagram lengkap ditunjukkan pada Gambar 4.2.

Berikut adalah penjelasan dari diagram pada Gambar 4.2:

1. (1) Pengguna memasukkan *seed URL* pada GUI.
2. (2) Pengguna menekan tombol *Check*.
3. (3) GUI memanggil metode `onCheckClick()` pada **Controller**.
4. (4) **Controller** membuat objek **Crawler** baru.
5. (5) **Crawler** menambahkan URL awal ke **Frontier** melalui `enqueueUrl()`.
6. (6) **Controller** memanggil `startCrawling()` pada **Crawler**.
7. (7–8) **Crawler** masuk ke dalam *loop* dengan mengambil URL dari **Frontier** menggunakan `dequeue()`, yang mengembalikan URL paling depan.
8. (9) URL di-*fetch* oleh **Crawler**.
9. (10–12) Jika terjadi *error*, maka dibuat objek **BrokenLink**. Objek ini dikirim ke **Controller**.

melalui `streamBrokenLink()`, kemudian (12–13) ditampilkan pada tabel *BrokenLink* di GUI.

10. (14–20) Jika status code hasil `fetch` oke, maka dilakukan:

11. (a) (14) Parsing HTML.

12. (b) (15–16) Ekstraksi dan normalisasi URL dari dokumen.

13. (c) (17) Dibuat objek `WebpageLink`.

14. (d) (18–20) Objek ini dikirim ke `Controller` melalui `streamWebpageLink()` dan ditampilkan
15. di tabel *WebpageLink* pada GUI.

16. 11. (21–26) Untuk setiap URL hasil ekstraksi (*loop*):

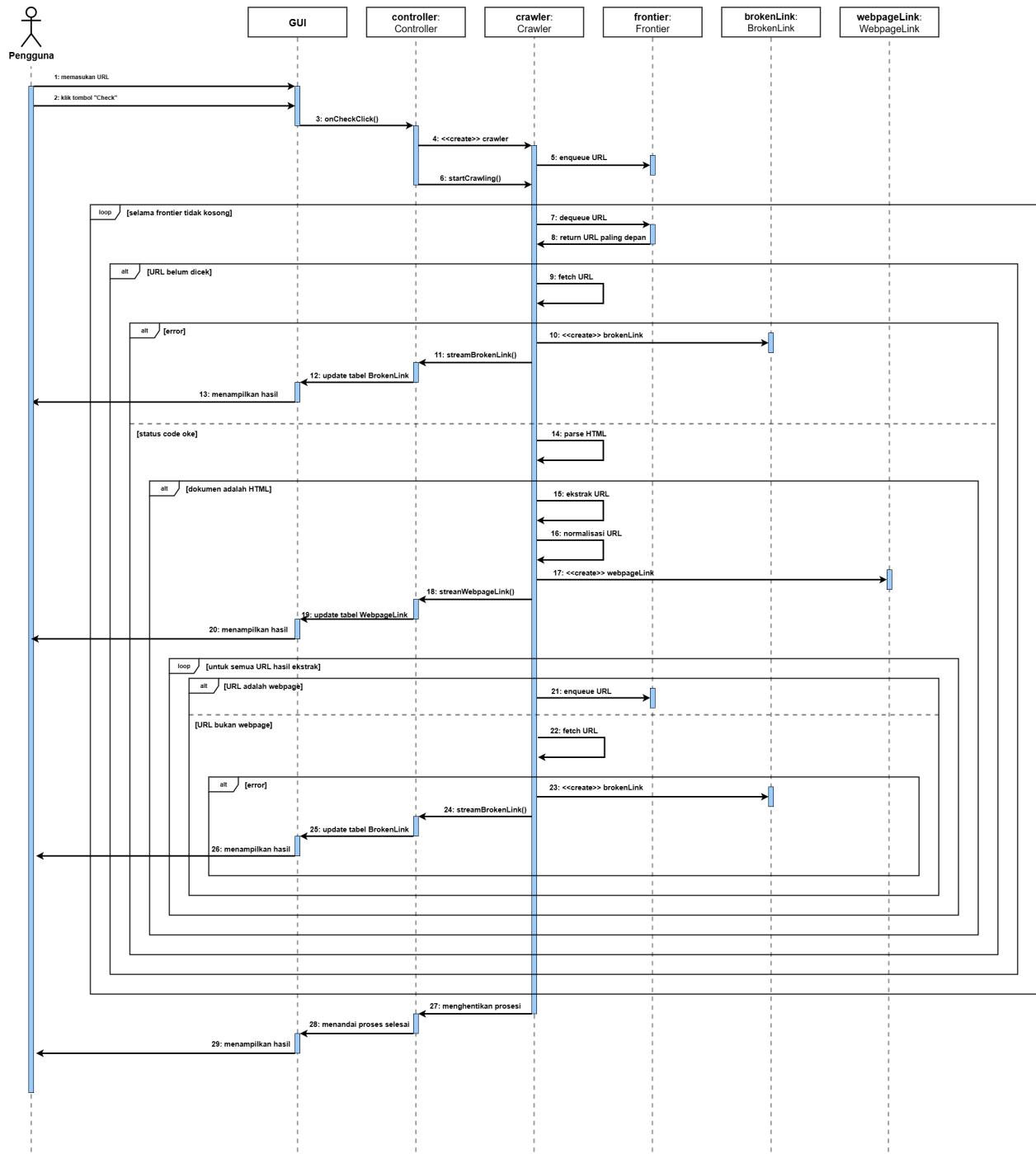
17. (a) Jika URL adalah halaman web, maka dimasukkan kembali ke `Frontier` melalui `enqueueUrl()`.

18. (b) Jika URL bukan halaman web, dilakukan `fetch`.

19. (c) (23–26) Jika `fetch` error, dibuat objek `BrokenLink`, dikirim ke `Controller` (`streamBrokenLink()`),
20. kemudian diperbarui pada tabel dan ditampilkan di GUI.

21. 12. (27–29) Proses berulang sampai `Frontier` kosong. Pada akhir siklus, `Crawler` menghentik-

22. an proses, `Controller` menandai proses selesai, dan GUI menampilkan status akhir serta
23. ringkasan hasil.



Gambar 4.2: Sequence diagram proses pemeriksaan tautan

4.3 Perancangan Antarmuka

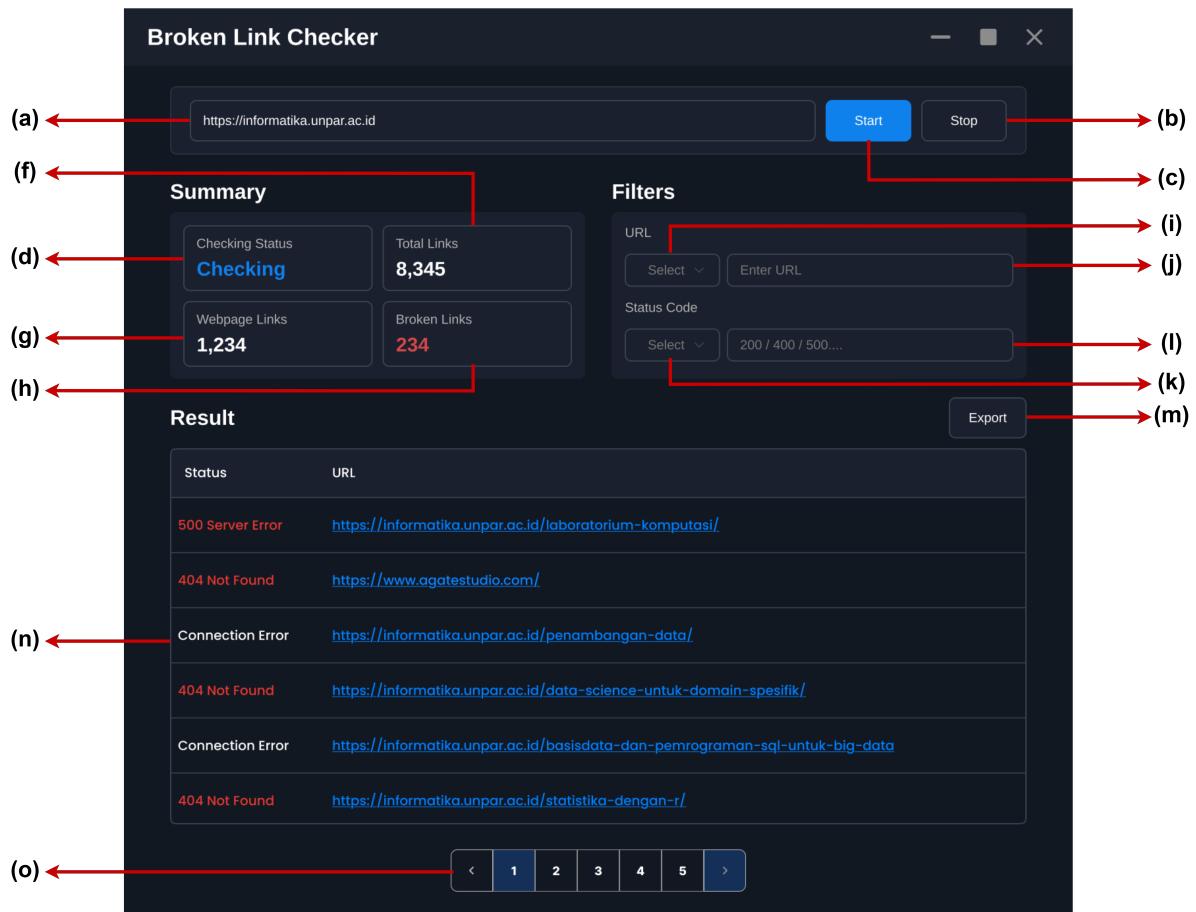
- Perancangan antarmuka pengguna dibuat dalam bentuk desain fidelitas tinggi dengan tujuan untuk memberikan gambaran visual yang mendekati tampilan akhir aplikasi berbasis JavaFX. Perancangan ini mempertimbangkan aspek keterbacaan, konsistensi elemen antarmuka, serta kemudahan interaksi bagi pengguna dalam menjalankan proses pemeriksaan tautan rusak.

1. Jendela Utama

- Jendela utama merupakan tampilan awal yang muncul ketika aplikasi dijalankan. Halaman ini

memfasilitasi pengguna untuk memasukkan URL situs web yang akan diperiksa, memulai atau menghentikan proses pemeriksaan, serta menampilkan hasil secara langsung. Elemen-elemen penting pada jendela ini ditunjukkan pada Gambar 4.3 dan dijelaskan sebagai berikut:

- (a) **Kolom Masukan URL:** digunakan untuk memasukkan alamat situs web yang akan diperiksa.
- (b) **Tombol Start:** digunakan untuk memulai proses pemeriksaan tautan.
- (c) **Tombol Stop:** menghentikan proses pemeriksaan yang sedang berlangsung.
- (d) **Checking Status:** menampilkan status terkini dari proses pemeriksaan, seperti *Checking*, *Stopped*, atau *Completed*.
- (e) **Total Links:** menunjukkan jumlah total tautan yang ditemukan selama proses *crawling*.
- (f) **Webpage Links:** menunjukkan jumlah tautan yang merupakan halaman situs web.
- (g) **Broken Links:** menunjukkan jumlah tautan rusak yang ditemukan selama proses pemeriksaan.
- (h) **Opsi Filter URL:** digunakan untuk menentukan jenis pencarian pada kolom URL. Opsi yang tersedia adalah:
 - **Equals:** menampilkan hasil dengan URL yang sama persis.
 - **Contains:** menampilkan hasil yang mengandung teks tertentu pada URL.
 - **Starts With:** menampilkan hasil dengan URL yang diawali teks tertentu.
 - **Ends With:** menampilkan hasil dengan URL yang diakhiri teks tertentu.
- (i) **Kolom Masukan Filter URL:** berfungsi untuk memasukkan kata kunci pencarian URL sesuai dengan opsi filter yang dipilih sebelumnya.
- (j) **Opsi Filter Status Code:** digunakan untuk menyaring hasil berdasarkan kategori kode status HTTP. Opsi yang tersedia adalah:
 - **Equals:** menampilkan tautan dengan kode status tertentu.
 - **Greater Than:** menampilkan tautan dengan kode status lebih besar dari nilai yang dimasukkan.
 - **Less Than:** menampilkan tautan dengan kode status lebih kecil dari nilai yang dimasukkan.
- (k) **Kolom Masukan Filter Status Code:** digunakan untuk memasukkan nilai numerik kode status HTTP sesuai dengan jenis filter yang dipilih pada poin sebelumnya.
- (l) **Tombol Export:** digunakan untuk mengekspor hasil pemeriksaan ke dalam berkas Excel.
- (m) **Tabel Hasil:** menampilkan daftar hasil pemeriksaan dalam bentuk tabel dengan dua kolom utama:
 - i. **Kolom Status:** menampilkan kode status HTTP dan *reason phrase*.
 - ii. **Kolom URL:** menampilkan alamat tautan rusak.
- (n) **Pagination:** digunakan untuk menavigasi hasil pemeriksaan jika jumlah tautan yang ditemukan cukup banyak.

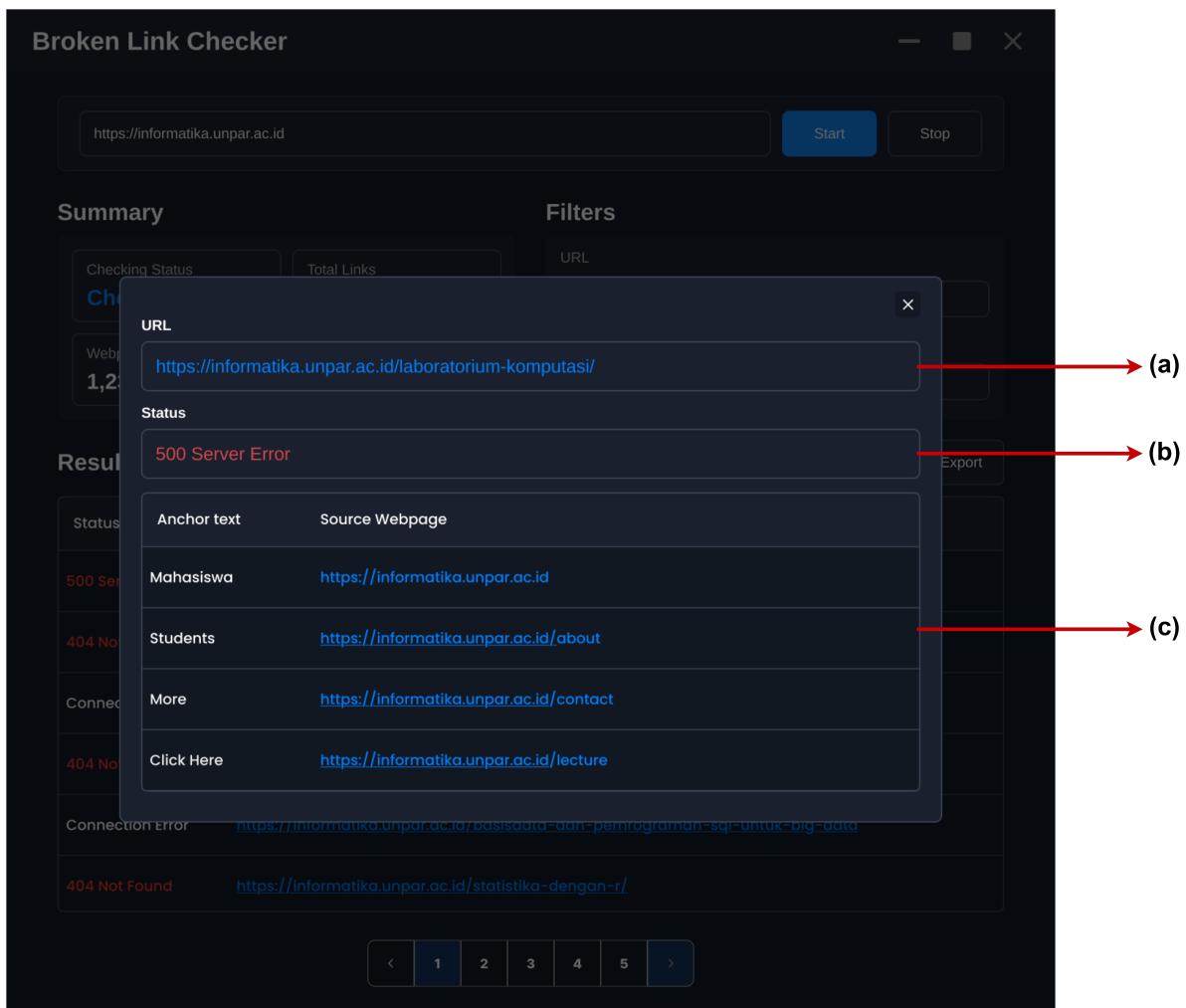


Gambar 4.3: Rancangan Halaman Utama

2. Jendela Detail Broken Link

Jendela ini muncul ketika pengguna memilih salah satu tautan rusak pada tabel hasil. Elemen-elemen penting pada jendela ini ditunjukkan pada Gambar 4.4 dan dijelaskan sebagai berikut:

- (a) **Kolom URL:** menampilkan alamat lengkap dari tautan rusak yang sedang diperiksa.
- (b) **Kolom Status:** menampilkan kode status HTTP dan *reason phrase*.
- (c) **Tabel Referensi Tautan:** berisi daftar halaman yang menjadi sumber dimana tautan rusak ditemukan, dengan dua kolom utama:
 - i. **Anchor Text:** teks tautan yang muncul pada halaman sumber.
 - ii. **Source Webpage:** URL halaman tempat tautan rusak ditemukan.



Gambar 4.4: Rancangan Jendela Detail Broken Link

¹

BAB 5

²

IMPLEMENTASI DAN PENGUJIAN

- ³ Bab ini membahas hasil implementasi aplikasi desktop pemeriksa tautan rusak pada situs web
⁴ berdasarkan perancangan yang telah disusun pada Bab [4](#), yang mencakup penetapan lingkungan
⁵ implementasi, implementasi kelas-kelas utama dalam sistem, dan implementasi antarmuka pengguna.
⁶ Setelah implementasi, dilakukan pengujian fungsional dan eksperimental serta dilakukan analisis
⁷ terhadap hasil pengujian.

⁸ 5.1 Implementasi

⁹ 5.2 Pengujian

¹

BAB 6

²

KESIMPULAN DAN SARAN

³ Bab ini menyajikan kesimpulan dari penelitian dan pengembangan aplikasi pemeriksa tautan
⁴ rusak yang telah dilakukan. Kesimpulan dirumuskan berdasarkan hasil analisis, perancangan,
⁵ implementasi, serta pengujian yang telah dipaparkan pada bab-bab sebelumnya. Selain itu, bab
⁶ ini juga memberikan saran yang dapat menjadi masukan untuk pengembangan lebih lanjut, baik
⁷ dalam peningkatan fitur maupun perluasan cakupan penelitian di masa mendatang.

⁸ 6.1 Kesimpulan

⁹ 6.2 Saran

DAFTAR REFERENSI

- [1] Fielding, R. T., Nottingham, M., dan Reschke, J. (2022) Http semantics. RFC 9110. RFC Editor, <http://www.rfc-editor.org>.
- [2] Berners-Lee, T., Fielding, R. T., dan Masinter, L. (2005) Uniform resource identifier (uri): Generic syntax. RFC 3986. RFC Editor, <http://www.rfc-editor.org>.
- [3] Powell, T. A. (2010) *HTML & CSS: The Complete Reference*, 5th edition. McGraw-Hill, New York.
- [4] Liu, B. (2011) *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, 2nd edition. Springer, Berlin, Heidelberg.
- [5] Sharan, K. dan Späth, P. (2022) *Learn JavaFX 17: Building User Experience and Interfaces with Java*, 2nd edition. Apress, Montgomery, AL, USA.

LAMPIRAN A
KODE PROGRAM

LAMPIRAN B
HASIL EKSPERIMEN