

SKRIPSI

PEREKAMAN KEHADIRAN DARING OTOMATIS



Reinalta Sugianto

NPM: 2017730035

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2022**

DAFTAR ISI

DAFTAR ISI	iii
DAFTAR GAMBAR	v
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metodologi	2
1.6 Sistematika Pembahasan	3
2 LANDASAN TEORI	5
2.1 Portal Akademik Mahasiswa 2018	5
2.2 Selenium	9
2.2.1 Navigasi <i>Browser</i>	10
2.2.2 Menemukan elemen	10
2.2.3 Waits	12
DAFTAR REFERENSI	13
A KODE PROGRAM	15
B HASIL EKSPERIMEN	17

DAFTAR GAMBAR

2.1	Tampilan halaman awal Portal Akademik Mahasiswa	5
2.2	Tampilan halaman untuk memasukan <i>email</i> Portal Akademik Mahasiswa	6
2.3	Tampilan halaman untuk memasukan <i>password</i> Portal Akademik Mahasiswa	6
2.4	Tampilan halaman setelah berhasil <i>login</i>	6
2.5	Tampilan halaman profil mahasiswa	7
2.6	Tampilan halaman pembayaran bagian Tagihan Pembayaran	7
2.7	Tampilan halaman pembayaran bagian Riwayat Pembayaran	8
2.8	Tampilan halaman pembayaran bagian Keterangan	8
2.9	Tampilan halaman nilai bagian Nilai per Semester	9
2.10	Tampilan halaman nilai bagian Riwayat Index Prestasi	9
B.1	Hasil 1	17
B.2	Hasil 2	17
B.3	Hasil 3	17
B.4	Hasil 4	17

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Perkuliahan di UNPAR biasanya membutuhkan perekaman kehadiran untuk mengetahui kehadiran mahasiswa dan dosen, bagi mahasiswa UNPAR perekaman kehadiran biasanya dilakukan dengan melakukan tanda tangan pada daftar kehadiran atau dicatat langsung oleh dosen yang memanggil mahasiswanya, sedangkan bagi dosen UNPAR perekaman kehadiran dilakukan dengan menggunakan *fingerprint*. Perekaman kehadiran diperkirakan membutuhkan waktu sekitar kurang dari 5 detik.

Pada tahun 2020 terjadi pandemi Covid-19 di seluruh negara. Pandemi Covid-19 masuk ke Indonesia pada awal bulan Maret tahun 2020. Covid-19 adalah penyakit yang disebabkan oleh virus *severe acute respiratory syndrome coronavirus 2* (SARS-CoV-2)¹. Penularan virus Covid-19 terjadi saat seseorang menyentuh barang yang sudah terkontaminasi oleh droplet orang yang terkena virus Covid-19 atau terkena droplet orang lain saat berinteraksi langsung dengan orang yang terkena virus Covid-19. Akibat pandemi Covid-19 yang dapat menular ini, maka hampir seluruh kegiatan di Indonesia dilakukan secara daring untuk mengurangi interaksi orang secara langsung yang dapat meningkatkan angka penularan virus tersebut.

Pembelajaran secara daring diberlakukan oleh UNPAR di akhir bulan Maret untuk seluruh kegiatan perkuliahan demi mencegah penularan virus Covid-19. Akibat diberlakukannya pembelajaran secara daring, maka perekaman kehadiran di UNPAR dilakukan dengan menggunakan aplikasi atau situs web milik UNPAR. Cara perekaman kehadiran secara daring di UNPAR ini membutuhkan waktu lebih agar dapat tercatat perekaman kehadirannya, karena butuh waktu untuk membuka situs web serta perlu memasukan *email* dan *password* hingga akhirnya melakukan perekaman kehadiran.

Selenium adalah *open-source framework* pengujian otomatisasi untuk aplikasi web[1]. WebDriver menggunakan API otomatisasi *browser* yang disediakan oleh vendor *browser* untuk mengontrol *browser* dan melakukan pengujian. API WebDriver ini seolah-olah membuat pengguna secara langsung mengoperasikan *browser*, padahal dijalankan secara otomatis langsung oleh API WebDriver tersebut. Selenium WebDriver adalah sebuah *tools* yang berguna untuk melakukan otomatisasi terhadap web pada *browser*. Selenium WebDriver ini tersedia untuk bahasa pemrograman Ruby, Java, Python, C#, dan JavaScript. Pembuatan Perekaman kehadiran daring otomatis ini akan menggunakan Selenium WebDriver dengan bahasa pemrograman Python.

Pada skripsi ini, akan dibuat sebuah perangkat lunak yang dapat melakukan perekaman kehadiran otomatis dengan sistem menerima rangsangan satu “klik” sehingga dapat melakukan

¹Pandemi Covid-19 di Indonesia https://id.wikipedia.org/wiki/Pandemi_Covid-19_di_Indonesia

1 hal-hal berikut :

- 2 1. Membuka peramban.
- 3 2. Membuka situs web perekaman kehadiran.
- 4 3. Mengisi dan *login* dengan *username* serta *password* yang ddiambil dari file konfigurasi.
- 5 4. Melakukan rekam kehadiran.

6 perangkat lunak ini bertujuan agar mahasiswa dan dosen dapat melakukan perekaman kehadiran
7 secara online dengan lebih mudah serta mengurangi waktu yang dibutuhkan untuk berinteraksi
8 dengan aplikasi atau situs web dan bukan untuk mempercepat waktu agar kehadiran terekam,
9 sehingga membuat waktu perekaman kehadiran secara daring dapat menyamai waktu perekaman
10 kehadiran secara luring.

11 1.2 Rumusan Masalah

12 Rumusan masalah yang akan dibahas di skripsi ini adalah sebagai berikut :

- 13 • Bagaimana cara membangun program Perekaman Kehadiran Daring Otomatis?
- 14 • Bagaimana cara mengurangi waktu interaksi dengan aplikasi atau situs web untuk merekam
- 15 kehadiran?

16 1.3 Tujuan

17 Tujuan yang ingin dicapai dari penulisan skripsi ini sebagai berikut :

- 18 • Membangun program menggunakan Selenium WebDriver.
- 19 • Membuat program yang mampu menerima rangsangan satu tombol untuk melakukan beberapa
- 20 hal menggunakan Selenium.

21 1.4 Batasan Masalah

22 Beberapa batasan yang dibuat terkait dengan pengerjaan skripsi ini adalah sebagai berikut :

- 23 1. Program ini bukan untuk mempercepat kehadiran terekam, hanya untuk mengurangi waktu
- 24 untuk berinteraksi dengan aplikasi.

25 1.5 Metodologi

26 Metodologi yang dilakukan pada skripsi ini adalah sebagai berikut :

- 27 1. Melakukan studi mengenai Selenium WebDriver.
- 28 2. Mempelajari bahasa pemrograman python.
- 29 3. Mempelajari cara menggunakan Selenium.
- 30 4. Menganalisis web Student Portal UNPAR.
- 31 5. Membangun program perekaman kehadiran daring otomatis.
- 32 6. Melakukan pengujian dan eksperimen.
- 33 7. Menulis dokumen skripsi.

1.6 Sistematika Pembahasan

Sistematika penulisan setiap bab skripsi ini adalah sebagai berikut :

1. Bab 1 Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan yang digunakan untuk menyusun skripsi ini.

2. Bab 2 Dasar Teori

Bab ini berisi teori-teori yang digunakan dalam pembuatan skripsi ini. Teori yang digunakan yaitu Selenium dan Portal Akademik Mahasiswa.

3. Bab 3 Analisis Masalah

Bab ini berisi analisis yang digunakan pada skripsi ini, analisa kebutuhan program Perekaman Kehadiran Online dan analisis Portal Akademik Mahasiswa.

4. Bab 4 Perancangan

Bab ini berisi perancangan program perekaman kehadiran daring otomatis yang akan dibuat.

5. Bab 5 Implementasi dan Pengujian

Bab ini berisi implementasi dan pengujian program, meliputi lingkungan implementasi, hasil implementasi, pengujian fungsional, dan pengujian eksperimental.

6. Bab 6 Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pembangunan program beserta saran untuk pengembangan selanjutnya.

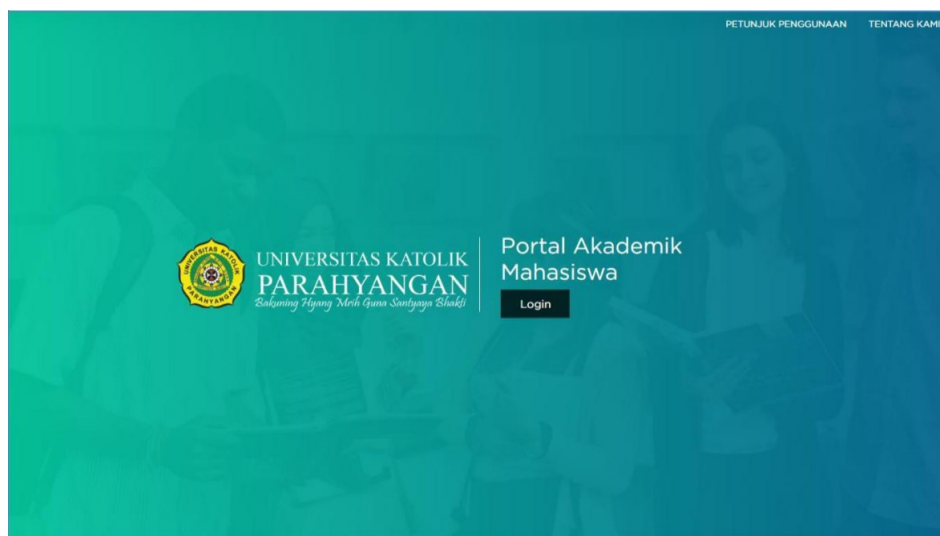
BAB 2

LANDASAN TEORI

Pada bab ini akan menjelaskan dasar teori mengenai Portal Akademik Mahasiswa dan Selenium.

2.1 Portal Akademik Mahasiswa 2018

Portal Akademik Mahasiswa (selanjutnya disingkat dengan PAM) adalah sebuah *web* yang diperuntukan bagi mahasiswa dalam rangka mendapatkan informasi kegiatan akademik mulai dari registrasi, melihat jadwal kuliah dan ujian, info nilai sampai pendaftaran sidang[2]. Portal Akademik Mahasiswa dapat diakses melalui <https://studentportal.unpar.ac.id/>.



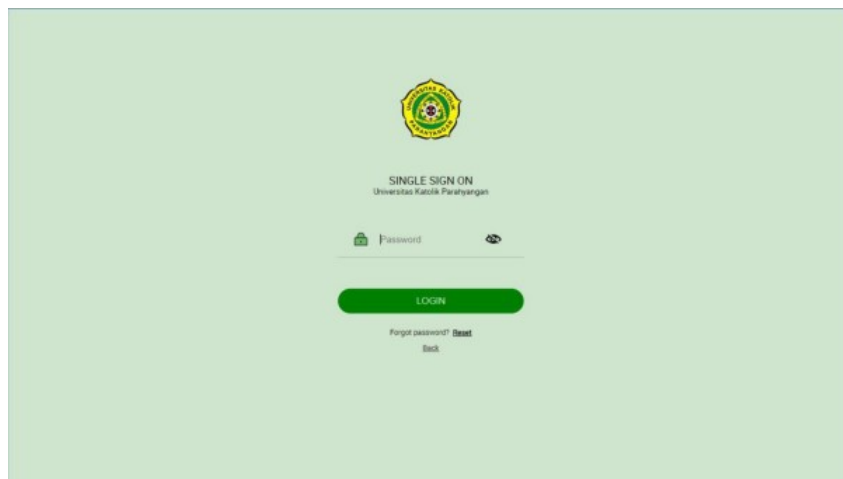
Gambar 2.1: Tampilan halaman awal Portal Akademik Mahasiswa

Pada Gambar 2.1 adalah tampilan awal ketika masuk ke halaman <https://studentportal.unpar.ac.id/>. Mahasiswa perlu melakukan *login* dengan *email* dan *password* mahasiswa UNPAR untuk dapat menggunakan fitur-fitur yang tersedia seperti:

1. Fitur mengisi form rencana semester (FRS) atau melakukan perubahan rencana studi (PRS) secara online
Panduan untuk melakukan FRS/PRS online.
 - (a) Masuk ke halaman <https://studentportal.unpar.ac.id/> lalu klik tombol “*Login*” yang dapat dilihat pada Gambar 2.1
 - (b) Lakukan “*Login*” dengan memasukan email dan password mahasiswa UNPAR pada halaman sso.



Gambar 2.2: Tampilan halaman untuk memasukan *email* Portal Akademik Mahasiswa



Gambar 2.3: Tampilan halaman untuk memasukan *password* Portal Akademik Mahasiswa

- 1 (c) Ketika *login* telah berhasil, maka browser akan menampilkan halaman utama, lalu klik
2 pada heksagon berlabel 'FRS/PRS' untuk melakukan FRS/PRS online.



Gambar 2.4: Tampilan halaman setelah berhasil *login*

- (d) Mahasiswa dapat melakukan FRS sesuai waktu yang sudah ditentukan atau mahasiswa dapat melakukan PRS setelah FRS selesai dan sesuai waktu yang sudah ditentukan untuk PRS.
2. Fitur Profil Mahasiswa Panduan untuk melihat profil mahasiswa.
 - (a) Mahasiswa melakukan *login* terlebih dahulu.
 - (b) Menekan menu “PROFIL” pada halaman setelah berhasil login seperti pada Gambar 2.4.
 - (c) Mahasiswa dapat melihat informasi data diri di halaman profil mahasiswa.



Gambar 2.5: Tampilan halaman profil mahasiswa

3. Fitur Pembayaran Panduan untuk melihat informasi pembayaran.
 - (a) Mahasiswa melakukan *login* terlebih dahulu.
 - (b) Menekan menu “PEMBAYARAN” pada halaman setelah berhasil login seperti pada Gambar 2.4.
 - (c) Pada halaman pembayaran, mahasiswa dapat melihat informasi pembayaran yang terdiri dari Tagihan Pembayaran, Riwayat Pembayaran, dan Keterangan. Pada Gambar 2.6 adalah tabel “Tagihan Pembayaran” yang menampilkan jenis tagihan dan jumlah tagihan dari setiap jenis tagihan yang ada.

Jenis Tagihan	Jumlah Tagihan
HUTANG SEBELUMNYA	Rp. 0,-
Tahap 01	Rp. 8.190.000,-
Denda Tahap 01	Rp. 0,-
Tahap 02	Rp. 2.458.000,-
Denda Tahap 02	Rp. 0,-
PENANGKARAN	Rp. 0,-
PENGEMBALAN	Rp. 0,-
TOTAL	Rp. 10.648.000,-
Kekurangan Pembayaran Semester Genap 2017/2018	Rp. 0,-

Tanggal Pembayaran	Jumlah Pembayaran	No. Transaksi	Bank
--------------------	-------------------	---------------	------

Gambar 2.6: Tampilan halaman pembayaran bagian Tagihan Pembayaran

Pada Gambar 2.7 adalah tabel “Riwayat Pembayaran” yang menampilkan histori pembayaran yang telah dilakukan.

Tanggal Pembayaran	Jumlah Pembayaran	No. Transaksi	Bank
02 Januari 2018	Rp. 2.890.000,-	85040988	BRI
18 Februari 2018	Rp. 5.300.000,-	8EAI	BEA UNPAR
07 Maret 2018	Rp. 3.488.000,-	05478984	BRI

Keterangan

Pembayaran dapat dilakukan dengan cara :

A. Bank BRI

I. Tunai Teller di Bank BRI Seluruh Indonesia :

1. Ke Kantor Kas/Cabang/Pusat BANK BRI terdekat atau yang ada di UNPAR

Gambar 2.7: Tampilan halaman pembayaran bagian Riwayat Pembayaran

Pada Gambar 2.8 adalah tabel “Keterangan” yang menampilkan tata cara pembayaran yang dapat dilakukan untuk melakukan pembayaran.

Keterangan

Pembayaran dapat dilakukan dengan cara :

A. Bank BRI

I. Tunai Teller di Bank BRI Seluruh Indonesia :

1. Ke Kantor Kas/Cabang/Pusat BANK BRI terdekat atau yang ada di UNPAR

2. Memberitahu kepada Teller bahwa akan membayar uang kuliah (KEY WORD: SPP-ON LINE) Universitas Katolik Parahyangan

3. Memberikan informasi NPM (Nomor Pokok Mahasiswa)

4. Melakukan konfirmasi nama dan jumlah tagihan yang akan dibayar

5. Apabila NPM, Nama dan jumlah tagihan sudah sesuai, silakan melakukan pembayaran

6. Mahasiswa wajib menyimpan slip pembayaran sebagai bukti transaksi yang dapat digunakan jika ada terjadi kesalahan di kemudian hari.

II. Kartu ATM Bank BRI & Mesin BRI :

1. Pilih Menu Transaksi Lainnya > Pembayaran > Lainnya > Briva

2. Masukkan Nomor Virtual Account dengan format 70285*2+NPM

Contoh: NPM 2008130130, Masukkan 7028522008130130

III. Kartu ATM Bank Lain & Mesin BRI :

1. Pilih Menu Transaksi Lainnya > Pembayaran > Lainnya > Briva

2. Masukkan Nomor Virtual Account dengan format 70285*2+NPM

Contoh: NPM 2008130130, Masukkan 7028522008130130

IV. Kartu ATM & Mesin Bank Lainnya :

1. Pilih Menu Transfer > Transfer antar Bank

2. Masukkan Kode Bank BRI (002) dan Nomor Virtual Account dengan format 70285*2+NPM

3. Masukkan nilai uang sesuai dengan jumlah tagihan (tidak boleh kurang atau lebih)

Contoh: NPM 2008130130, Masukkan 0027028522008130130

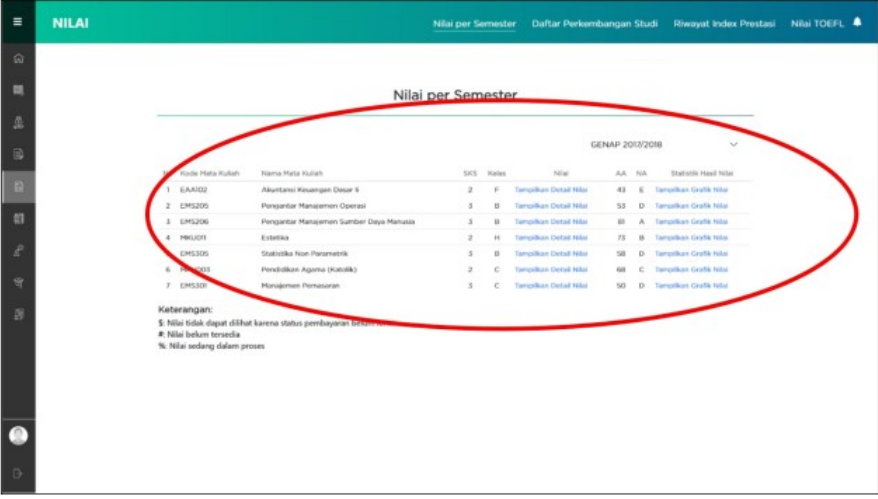
Gambar 2.8: Tampilan halaman pembayaran bagian Keterangan

4. Fitur Nilai Panduan untuk melihat informasi nilai mahasiswa.

(a) Mahasiswa melakukan *login* terlebih dahulu.

(b) Menekan menu “NILAI” pada halaman setelah berhasil login seperti pada Gambar 2.4.

(c) Pada halaman nilai, mahasiswa dapat melihat informasi nilai dari setiap mata kuliah yang diambil.



Nilai per Semester

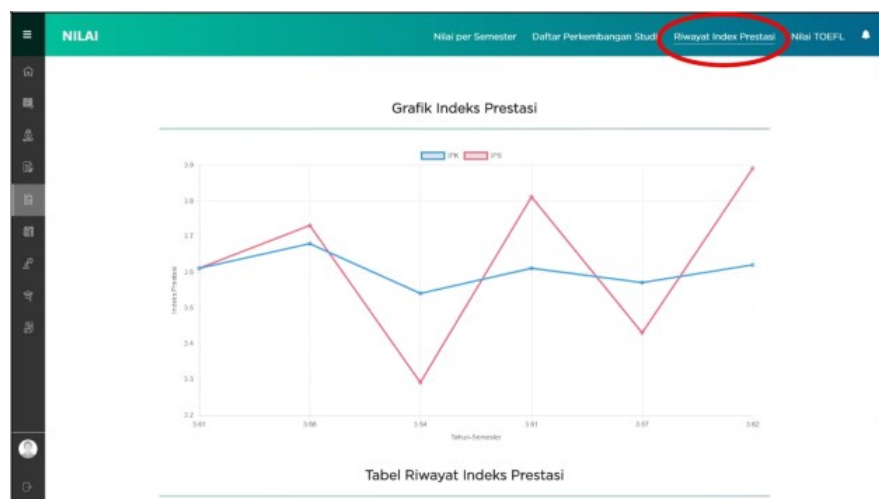
GENAP 2017/2018

No	Kode Mata Kuliah	Nama Mata Kuliah	SKS	Kelas	Nilai	AA	NA	Statistik Hasil Nilai
1	EA002	Akuntansi Keuangan Dasar 1	2	F	Tampilkan Detail Nilai	43	E	Tampilkan Grafik Nilai
2	EH0205	Pengantar Manajemen Operasi	3	B	Tampilkan Detail Nilai	53	D	Tampilkan Grafik Nilai
3	EH0206	Pengantar Manajemen Sumber Daya Manusia	3	B	Tampilkan Detail Nilai	61	A	Tampilkan Grafik Nilai
4	EH0207	Etika	2	H	Tampilkan Detail Nilai	75	B	Tampilkan Grafik Nilai
5	EH0205	Statistika Non-Parametrik	3	B	Tampilkan Detail Nilai	58	D	Tampilkan Grafik Nilai
6	EH0208	Pengolahan Algoritma (Kalkulus)	2	C	Tampilkan Detail Nilai	68	C	Tampilkan Grafik Nilai
7	EH0207	Manajemen Pemasaran	3	C	Tampilkan Detail Nilai	50	D	Tampilkan Grafik Nilai

Keterangan:
 \$ Nilai tidak dapat dilihat karena status pembayaran belum lunas
 # Nilai belum tersedia
 % Nilai sedang dalam proses

Gambar 2.9: Tampilan halaman nilai bagian Nilai per Semester

- (d) Mahasiswa dapat mengakses menu “Riwayat Index Prestasi” untuk melihat ‘IPK’ dan ‘IPS’ mahasiswa.



Gambar 2.10: Tampilan halaman nilai bagian Riwayat Index Prestasi

2.2 Selenium

Selenium adalah *open-source framework* pengujian otomatisasi untuk aplikasi web[1]. Selenium ini adalah WebDriver yang merupakan sebuah *interface* untuk menulis suatu instruksi yang dapat dijalankan secara otomatis dan bergantian pada *browser*. Selenium WebDriver adalah sebuah *tools* yang berguna untuk melakukan otomatisasi terhadap web pada *browser*. Selenium WebDriver menggunakan API otomatisasi *browser* yang disediakan oleh vendor *browser* untuk mengontrol *browser* dan melakukan pengujian. API WebDriver ini seolah-olah membuat pengguna secara langsung mengoperasikan *browser*, padahal dijalankan secara otomatis langsung oleh API WebDriver tersebut. Selenium WebDriver ini tersedia untuk bahasa pemrograman Ruby, Java, Python, C#, dan JavaScript. Selenium WebDriver memiliki berbagai fungsi, yaitu Navigasi *Browser*, Menemukan elemen, *Waits*.

2.2.1 Navigasi *Browser*

Navigasi *browser* ini berfungsi untuk menjalankan otomatisasi pada browser. Terdapat beberapa metode navigasi browser:

1. Navigate to: hal pertama untuk menggunakan WebDriver adalah melakukan navigasi ke situs web.

Kode 2.1: Contoh kode Navigate to

```
1 driver.get("https://selenium.dev")
```

2. Get current URL: untuk membaca URL saat ini dari alamat *browser*.

Kode 2.2: Contoh kode Get current URL

```
1 driver.current_url
```

3. Back: menekan tombol kembali *browser*.

Kode 2.3: Contoh kode Back

```
1 driver.back()
```

4. Forward: menekan tombol maju *browser*.

Kode 2.4: Contoh kode Forward

```
1 driver.forward()
```

5. Refresh: melakukan *refresh* halaman.

Kode 2.5: Contoh kode Refresh

```
1 driver.refresh()
```

6. Get title: untuk dapat membaca judul halaman saat ini pada *browser*

Kode 2.6: Contoh kode Get title

```
1 driver.title
```

2.2.2 Menemukan elemen

Salah satu teknik mendasar untuk dipelajari saat menggunakan WebDriver adalah cara menemukan elemen di halaman web. WebDriver menawarkan sejumlah tipe pemilih bawaan, di antaranya menemukan elemen dengan atribut ID.

Kode 2.7: Contoh kode untuk menemukan elemen dengan atribut ID

```
1 driver.find_element(By.ID, "cheese")
2 driver.find_element_by_id("cheese")
```

Kode 2.7 merupakan 2 contoh kode yang dapat digunakan untuk menemukan elemen berdasarkan atribut ID. Terdapat delapan strategi lokasi elemen bawaan yang berbeda di WebDriver:

1. class name: Menemukan elemen yang nama kelasnya berisi nilai pencarian (nama kelas gabungan tidak diizinkan).
2. css selector: Menemukan elemen yang cocok dengan pemilih CSS.
3. id : Menemukan elemen yang atribut ID-nya cocok dengan nilai pencarian.

4. name: Menemukan elemen yang atribut NAME-nya cocok dengan nilai pencarian.
5. link text: Menemukan elemen jangkar yang teksnya terlihat cocok dengan nilai pencarian.
6. partial link text: Menemukan elemen jangkar yang teksnya terlihat berisi nilai pencarian.
- Jika beberapa elemen cocok, hanya yang pertama yang akan dipilih.
7. tag name: Menemukan elemen yang nama tagnya cocok dengan nilai pencarian.
8. xpath: Menemukan elemen yang cocok dengan ekspresi XPath.

Selenium 4 menghadirkan *Relative Locator* yang sebelumnya disebut *Friendly Locators*. Fungsi ini ditambahkan untuk membantu menemukan elemen yang berdekatan dengan elemen lain. Pencari Relatif yang Tersedia adalah:

1. above: Mengembalikan WebElement, yang muncul di atas elemen yang ditentukan.

Kode 2.8: Contoh kode Pencari Relatif Above

```
1 from selenium.webdriver.common.by import By
2 from selenium.webdriver.support.relative_locator import locate_with
3
4 passwordField = driver.find_element(By.ID, "password")
5 emailAddressField = driver.find_element(locate_with(By.TAG_NAME, "input").above(passwordField))
```

2. below: Mengembalikan WebElement, yang muncul di bawah elemen yang ditentukan.

Kode 2.9: Contoh kode Pencari Relatif Below

```
1 from selenium.webdriver.common.by import By
2 from selenium.webdriver.support.relative_locator import locate_with
3
4 emailAddressField = driver.find_element(By.ID, "email")
5 passwordField = driver.find_element(locate_with(By.TAG_NAME, "input").below(emailAddressField))
```

3. toLeftOf: Mengembalikan WebElement, yang muncul di sebelah kiri elemen yang ditentu.

Kode 2.10: Contoh kode Pencari Relatif toLeftOf

```
1 from selenium.webdriver.common.by import By
2 from selenium.webdriver.support.relative_locator import locate_with
3
4 submitButton = driver.find_element(By.ID, "submit")
5 cancelButton = driver.find_element(locate_with(By.TAG_NAME, "button").
6 to_left_of(submitButton))
```

4. toRightOf: Mengembalikan WebElement, yang muncul di sebelah kanan elemen yang ditentukan.

Kode 2.11: Contoh kode Pencari Relatif toRightOf

```
1 from selenium.webdriver.common.by import By
2 from selenium.webdriver.support.relative_locator import locate_with
3
4 cancelButton = driver.find_element(By.ID, "cancel")
5 submitButton = driver.find_element(locate_with(By.TAG_NAME, "button").
6 to_right_of(cancelButton))
```

5. near: Mengembalikan WebElement, yang paling jauh 50px dari elemen yang ditentukan.

Kode 2.12: Contoh kode Pencari Relatif Near

```
1 from selenium.webdriver.common.by import By
2 from selenium.webdriver.support.relative_locator import locate_with
3
4 emailAddressLabel = driver.find_element(By.ID, "lbl-email")
5 emailAddressField = driver.find_element(locate_with(By.TAG_NAME, "input").
6 near(emailAddressLabel))
```

2.2.3 Waits

WebDriver secara umum dapat dikatakan memiliki API pemblokiran. Karena ini adalah *library* di luar proses yang menginstruksikan browser apa yang harus dilakukan, dan karena platform web secara intrinsik memiliki sifat asinkron, WebDriver tidak melacak status document object model (DOM) yang aktif dan real-time. Selenium WebDriver memiliki tiga tipe waits:

1. Implicit wait: memberi tahu WebDriver untuk melakukan polling DOM selama jangka waktu tertentu ketika mencoba menemukan elemen atau elemen jika tidak segera tersedia. Pengaturan default adalah 0, artinya dinonaktifkan. Setelah disetel, penantian implisit disetel untuk masa pakai sesi.

Kode 2.13: Contoh kode Implicit wait

```
1 driver = Firefox()
2 driver.implicitly_wait(10)
3 driver.get("http://somedomain/url_that_delays_loading")
4 my_dynamic_element = driver.find_element(By.ID, "myDynamicElement")
```

2. Explicit wait: mengizinkan kode untuk menghentikan eksekusi program, atau membekukan *thread*, hingga suatu kondisi dapat teratasi. Kondisi ini dipanggil dengan frekuensi tertentu sampai batas waktu tunggu terlewati.

Kode 2.14: Contoh kode Explicit wait

```
1 from selenium.webdriver.support.ui import WebDriverWait
2
3 driver.navigate("file:///race_condition.html")
4 el = WebDriverWait(driver).until(lambda d: d.find_element_by_tag_name("p"))
5 assert el.text == "Hello_from_JavaScript!"
```

3. Fluent wait: menentukan jumlah waktu maksimum untuk menunggu suatu kondisi, serta frekuensi untuk memeriksa kondisi tersebut.

Kode 2.15: Contoh kode FluentWait

```
1 driver = Firefox()
2 driver.get("http://somedomain/url_that_delays_loading")
3 wait = WebDriverWait(driver, 10, poll_frequency=1, ignored_exceptions=[ElementNotVisibleException,
4                                     ElementNotSelectableException])
5 element = wait.until(EC.element_to_be_clickable((By.XPATH, "//div")))
```

DAFTAR REFERENSI

- [1] 9f08b37 (2021) *Selenium*. Software Freedom Conservancy. Online.
- [2] 2018, T. P. P. A. M. P. (2018) Portal akademik mahasiswa. https://studentportal.unpar.ac.id/assets/BUKU_PANDUAN_PENGGUNAAN_FRS_GABUNGAN.pdf. Online; diakses 15-November-2021.

LAMPIRAN A

KODE PROGRAM

Kode A.1: MyCode.c

```
1
2 // This does not make algorithmic sense,
3 // but it shows off significant programming characters.
4
5 #include<stdio.h>
6
7 void myFunction( int input, float* output ) {
8     switch ( array[i] ) {
9         case 1: // This is silly code
10             if ( a >= 0 || b <= 3 && c != x )
11                 *output += 0.005 + 20050;
12             char = 'g';
13             b = 2^n + ~right_size - leftSize * MAX_SIZE;
14             c = (--aaa + &daa) / (bbb++ - ccc % 2 );
15             strcpy(a,"hello_$@?");
16         }
17         count = ~mask | 0x00FF00AA;
18     }
19
20 // Fonts for Displaying Program Code in LATEX
21 // Adrian P. Robson, nepsweb.co.uk
22 // 8 October 2012
23 // http://nepsweb.co.uk/docs/progfonts.pdf
```

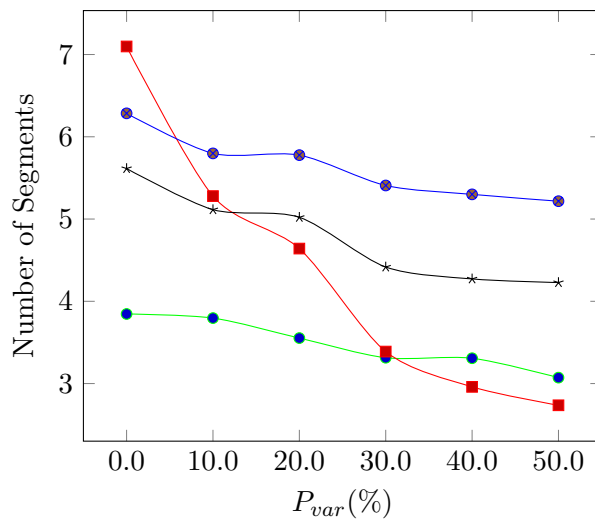
Kode A.2: MyCode.java

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id; //id of the set
8     protected MyEdge FurthestEdge; //the furthest edge
9     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
10    protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each trajectory
11    protected ArrayList<Integer> closeID; //store the ID of all vertices
12    protected ArrayList<Double> closeDist; //store the distance of all vertices
13    protected int totaltrj; //total trajectories in the set
14
15    /*
16     * Constructor
17     * @param id : id of the set
18     * @param totaltrj : total number of trajectories in the set
19     * @param FurthestEdge : the furthest edge
20     */
21    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
22        this.id = id;
23        this.totaltrj = totaltrj;
24        this.FurthestEdge = FurthestEdge;
25        set = new HashSet<MyVertex>();
26        ordered = new ArrayList<ArrayList<Integer>>();
27        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
28        closeID = new ArrayList<Integer>(totaltrj);
29        closeDist = new ArrayList<Double>(totaltrj);
30        for (int i = 0;i <totaltrj;i++) {
31            closeID.add(-1);
32            closeDist.add(Double.MAX_VALUE);
33        }
34    }
35
36 }
```


LAMPIRAN B

HASIL EKSPERIMEN

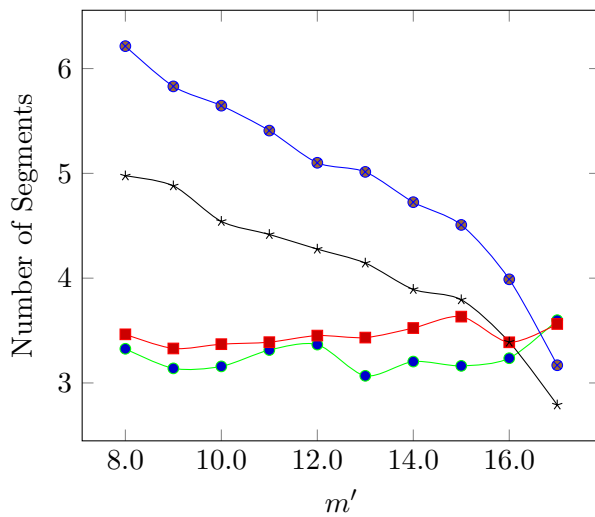
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4