

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	12
2.2.3 Waits	14
3 ANALISIS	17
3.1 Analisis Hasil Survei Perekaman Kehadiran Daring dan Luring	17
3.1.1 Hasil Survei Mahasiswa	17
3.1.2 Hasil Survei Dosen	19
3.2 Analisis Alur Perekaman Kehadiran Online	21
3.3 Cara Menerjemahkan Perekaman Kehadiran Online ke dalam Selenium	24
3.4 Analisis Program Sejenis	25
DAFTAR REFERENSI	29

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
3.1	Histogram Waktu Perekaman Kehadiran Daring Mahasiswa	18
3.2	Histogram Waktu Perekaman Kehadiran Daring Mahasiswa	19
3.3	Histogram Perbandingan Waktu Perekaman Kehadiran Dosen	20
3.4	Tampilan halaman awal Portal Akademik Mahasiswa	21
3.5	Tampilan halaman Portal Akademik Mahasiswa untuk memasukan <i>email</i>	22
3.6	Tampilan halaman Portal Akademik Mahasiswa untuk memasukan <i>password</i>	22
3.7	Tampilan peringatan pada halaman Portal Akademik Mahasiswa	22
3.8	Tampilan halaman Portal Akademik Mahasiswa setelah Berhasil <i>Login</i>	23
3.9	Tampilan halaman Portal Akademik Mahasiswa untuk Melakukan Absen	23
3.10	Tampilan Pemberitahuan Absensi Berhasil	24
3.11	Tampilan Menu Awal Selenium IDE	25
3.12	Tampilan Memasukan Nama Proyek	26
3.13	Tampilan Memasukan Situs Web	26
3.14	Tampilan Otomatisasi pada Selenium IDE	27

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.

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

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

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

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

1.2 Rumusan Masalah

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

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

1.3 Tujuan

Tujuan yang ingin dicapai dari penulisan skripsi ini sebagai berikut :

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

1.4 Batasan Masalah

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

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

1.5 Metodologi

Metodologi yang dilakukan pada skripsi ini adalah sebagai berikut :

1. Melakukan studi mengenai Selenium WebDriver.
2. Mempelajari bahasa pemrograman python.
3. Mempelajari cara menggunakan Selenium.
4. Menganalisis web Student Portal UNPAR.
5. Membangun program perekaman kehadiran daring otomatis.
6. Melakukan pengujian dan eksperimen.
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

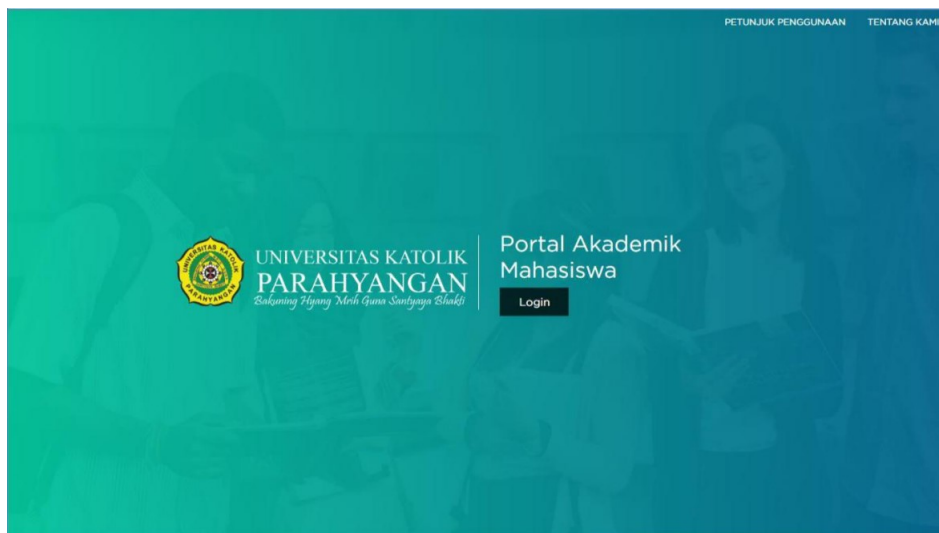
Bab ini berisi kesimpulan dari hasil pembangunan program beserta saran untuk pengembangan selanjutnya.

BAB 2

LANDASAN TEORI

2.1 Portal Akademik Mahasiswa 2018

Portal Akademik Mahasiswa (selanjutnya disingkat dengan PAM) adalah sebuah *web* yang di peruntukan 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 memasukkan 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.

Tagihan Pembayaran Semester Genap 2017/2018	
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,-
PENANGKAPAN	Rp. 0,-
PENGEMBALAN	Rp. 0,-
TOTAL	Rp. 10.648.000,-
Kekurangan Pembayaran Semester Genap 2017/2018	Rp. 0,-

Riwayat Pembayaran			
Tanggal Pembayaran	Jumlah Pembayaran	No. Transaksi	Berkas

Gambar 2.6: Tampilan halaman pembayaran bagian Tagihan Pembayaran

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

The screenshot shows a web interface for payments. At the top, there's a summary section with fields for 'PENGEMBALAN' (Rp. 0,-), 'TOTAL' (Rp. 1650.000,-), and 'Kekurangan Pembayaran Semester Kedua 2017/2018' (Rp. 0,-). Below this is a table titled 'Riwayat Pembayaran' (Payment History) with columns: Tanggal Pembayaran, Jumlah Pembayaran, No. Transaksi, and Bank. The table lists three transactions: 02 Januari 2018 (Rp. 2.890.000,-, BRI), 18 Februari 2018 (Rp. 5.300.000,-, BEA UNPAR), and 07 Maret 2018 (Rp. 3.480.000,-, BRI). Below the table is a section titled 'Keterangan' (Notes) explaining payment methods: Bank BRI (Tunai or ATM) and other banks (Virtual Account).

Tanggal Pembayaran	Jumlah Pembayaran	No. Transaksi	Bank
02 Januari 2018	Rp. 2.890.000,-	8546988	BRI
18 Februari 2018	Rp. 5.300.000,-	8EAI	BEA UNPAR
07 Maret 2018	Rp. 3.480.000,-	0547894	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.

The screenshot shows the 'Keterangan' (Notes) section of the payment page. It provides detailed instructions for three payment methods: Bank BRI (Tunai), Bank BRI (Kartu ATM), and other banks (Virtual Account). The instructions are numbered 1 through 6, covering steps from choosing the location to saving the receipt.

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

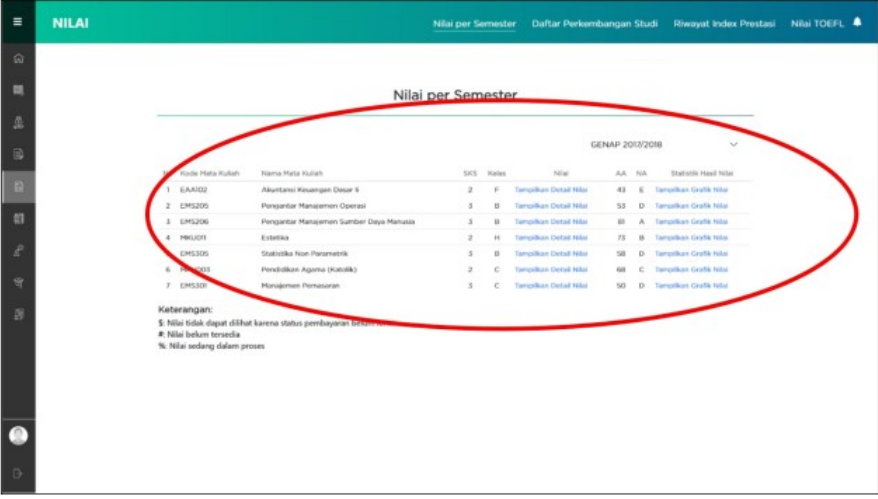
Gambar 2.8: Tampilan halaman pembayaran bagian Keterangan

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

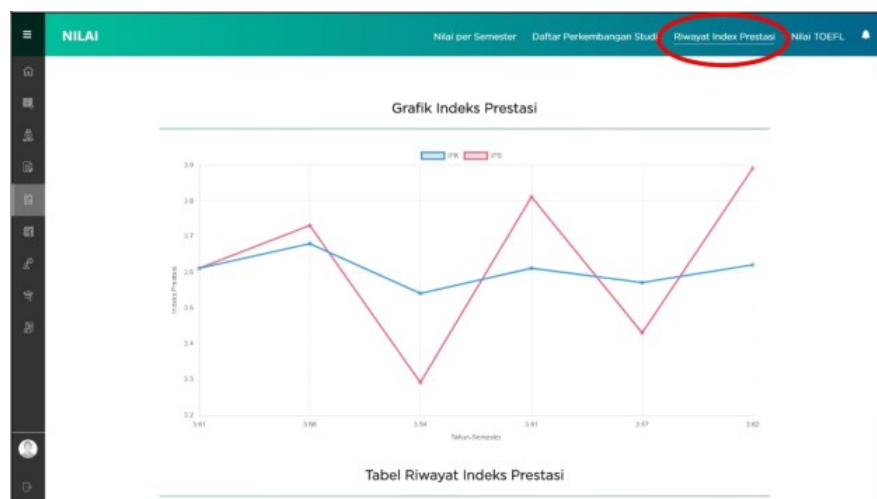
GENAP 2017/2018

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	EMS205	Pengantar Manajemen Operasi	3	B	Tampilkan Detail Nilai	53	D	Tampilkan Grafik Nilai
3	EMS206	Pengantar Manajemen Sumber Daya Manusia	3	B	Tampilkan Detail Nilai	61	A	Tampilkan Grafik Nilai
4	EMS207	Etika	2	H	Tampilkan Detail Nilai	75	B	Tampilkan Grafik Nilai
5	EMS205	Statistika Non-Parametrik	3	B	Tampilkan Detail Nilai	58	D	Tampilkan Grafik Nilai
6	EMS208	Pengambilan Keputusan (Kualitatif)	2	C	Tampilkan Detail Nilai	68	C	Tampilkan Grafik Nilai
7	EMS207	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* serta Menemukan elemen.

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 from selenium import webdriver
2 driver = webdriver.Chrome(executable_path='D:\Selenium\chromedriver.exe')
3 url = "https://selenium.dev"
4 link = driver.get(url)
```

Pada Kode 2.1 merupakan contoh untuk memunculkan situs web yang ingin dijalankan dengan selenium. Baris 1 melakukan *import* webdriver terlebih dahulu, lalu baris 2 *string* dengan nama driver memanggil webdriver yang ingin digunakan, yaitu Google Chrome dan diisi letak file chromedriver.exe disimpan. Baris 3 *string* dengan nama url diisi dengan situs web yang dituju dalam contoh adalah <https://selenium.dev>. Baris 4 adalah *string* dengan nama link menggunakan *method* *get* yang memanggil *string* dengan nama driver yang sudah memanggil webdriver, lalu ditambahkan *method* *get* yang memanggil *string* dengan nama url yang sudah berisi situs web yang dituju.

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

Kode 2.2: Contoh kode *Get current URL*

```
1 from selenium import webdriver
2 driver = webdriver.Chrome(executable_path='D:\Selenium\chromedriver.exe')
3 url = "https://selenium.dev"
4 link = driver.get(url)
5 current = driver.current_url
6 print(current)
```

Pada Kode 2.2 merupakan contoh untuk membaca situs web yang dijalankan dari *browser*. Pada baris 1 sampai 4 merupakan contoh untuk memunculkan situs web yang ingin dijalankan dengan selenium dan sudah dijelaskan pada Kode 2.1. Baris 5 adalah *string* dengan nama *current* yang memanggil *method* *current_url* yang berfungsi untuk mendapatkan situs web dari *browser*. Baris 6 adalah untuk menampilkan situs webnya dengan *method* *print* dan diisi dengan *string* dengan nama *current* sehingga nantinya akan muncul situs webnya.

3. *Back*: menekan tombol kembali pada *browser*.

Kode 2.3: Contoh kode *Back*

```
1 from selenium import webdriver
2 driver = webdriver.Chrome(executable_path='D:\Selenium\chromedriver.exe')
3 url = "https://selenium.dev"
4 link = driver.get(url)
5 kembali = driver.back()
```

Pada Kode 2.3 merupakan contoh untuk menekan tombol kembali pada *browser*. Pada baris 1 sampai 4 merupakan contoh untuk memunculkan situs web yang ingin dijalankan dengan selenium dan sudah dijelaskan pada Kode 2.1. Baris 5 adalah *string* dengan nama kembali diisi dengan *method* *back()* yang berfungsi untuk menekan tombol kembali pada *browser*.

4. *Forward*: menekan tombol maju *browser*.

Kode 2.4: Contoh kode Forward

```
1 from selenium import webdriver
2 driver = webdriver.Chrome(executable_path='D:\\Selenium\\chromedriver.exe')
3 url = "https://selenium.dev"
4 link = driver.get(url)
5 maju = driver.forward()
```

Pada Kode 2.4 merupakan contoh untuk menekan tombol maju pada *browser*. Pada baris 1 sampai 4 merupakan contoh untuk memunculkan situs web yang ingin dijalankan dengan selenium dan sudah dijelaskan pada Kode 2.1. Baris 5 adalah *string* dengan nama maju diisi dengan *method forward()* yang berfungsi untuk menekan tombol maju pada *browser*.

5. *Refresh*: melakukan *refresh* halaman.

Kode 2.5: Contoh kode Refresh

```
1 from selenium import webdriver
2 driver = webdriver.Chrome(executable_path='D:\\Selenium\\chromedriver.exe')
3 url = "https://selenium.dev"
4 link = driver.get(url)
5 refresh = driver.refresh()
```

Pada Kode 2.5 merupakan contoh untuk menekan tombol *refresh* halaman web pada *browser*. Pada baris 1 sampai 4 merupakan contoh untuk memunculkan situs web yang ingin dijalankan dengan selenium dan sudah dijelaskan pada Kode 2.1. Baris 5 adalah *string* dengan nama *refresh* diisi dengan *method refresh()* yang berfungsi untuk menekan tombol *refresh* halaman web pada *browser*.

6. *Get title*: untuk dapat membaca judul halaman situs web pada *browser*

Kode 2.6: Contoh kode Get title

```
1 from selenium import webdriver
2 driver = webdriver.Chrome(executable_path='D:\\Selenium\\chromedriver.exe')
3 url = "https://selenium.dev"
4 link = driver.get(url)
5 title = driver.title
6 print(judul)
```

Pada Kode 2.6 merupakan contoh untuk membaca judul halaman situs web yang dijalankan dari *browser*. Pada baris 1 sampai 4 merupakan contoh untuk memunculkan situs web yang ingin dijalankan dengan selenium dan sudah dijelaskan pada Kode 2.1. Baris 5 adalah *string* dengan nama *title* yang memanggil *method title* yang berfungsi untuk mendapatkan judul halaman situs web dari *browser*. Baris 6 adalah untuk menampilkan judul situs webnya dengan *method print* dan diisi dengan *string* dengan nama *title* sehingga nantinya akan muncul judul situs webnya.

7. *Quit the browser*: untuk dapat keluar dari *browser* setelah selesai menggunakan.

Kode 2.7: Contoh kode Get title

```
1 from selenium import webdriver
2 driver = webdriver.Chrome(executable_path='D:\\Selenium\\chromedriver.exe')
3 url = "https://selenium.dev"
4 link = driver.get(url)
5 quit = driver.quit()
```

Pada Kode 2.7 merupakan contoh untuk dapat keluar dari *browser* setelah selesai menggunakan. Pada baris 1 sampai 4 merupakan contoh untuk memunculkan situs web yang ingin dijalankan dengan selenium dan sudah dijelaskan pada Kode 2.1. Baris 5 adalah *string* dengan nama *quit* yang memanggil *method quit()* yang berfungsi untuk dapat keluar dari *browser* setelah selesai digunakan.

2.2.2 Menemukan elemen

Salah satu teknik mendasar untuk dipelajari saat menggunakan WebDriver adalah cara menemukan elemen di halaman web. WebDriver menyediakan berbagai cara untuk menemukan elemen, terdapat delapan strategi menemukan lokasi elemen yang berbeda di WebDriver:

1. Id : Menemukan elemen yang atribut ID-nya cocok dengan nilai pencarian.

Kode 2.8: Contoh kode untuk menemukan elemen dengan atribut ID

```
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 driver = webdriver.Chrome(executable_path='D:\Selenium\chromedriver.exe')
4 url = "https://selenium.dev"
5 driver.get(url)
6 driver.find_element(By.ID, "cheese")
7 driver.find_element_by_id("cheese")
```

Pada Kode 2.8 baris 6 atau 7 merupakan contoh kode yang dapat digunakan untuk menemukan elemen berdasarkan atribut ID dengan nama id “cheese” dari situs web <https://selenium.dev>.

2. *Class name*: Menemukan elemen yang nama kelasnya berisi nilai pencarian.

Kode 2.9: Contoh kode untuk menemukan elemen dengan *class name*

```
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 driver = webdriver.Chrome(executable_path='D:\Selenium\chromedriver.exe')
4 url = "https://selenium.dev"
5 driver.get(url)
6 kelas = driver.find_elements(By.CLASS_NAME, "text-center")
```

Pada Kode 2.9 baris 6 merupakan contoh kode untuk mencari elemen dengan *class name* “text-center” dan disimpan dalam *string* kelas.

3. *CSS selector*: Menemukan elemen yang cocok dengan pemilihan *Cascading Style Sheets* (CSS). Pemilihan pada CSS adalah pola yang digunakan untuk memilih elemen dengan *style* yang diinginkan.

Kode 2.10: Contoh kode untuk menemukan elemen dengan *CSS selector*

```
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 driver = webdriver.Chrome(executable_path='D:\Selenium\chromedriver.exe')
4 url = "https://selenium.dev"
5 driver.get(url)
6 select = driver.find_element(By.CSS_SELECTOR, "#selenium_logo")
```

Pada Kode 2.10 baris 6 merupakan contoh kode yang disimpan dalam *string select* untuk mencari elemen berdasarkan pemilihan CSS dengan mengambil elemen dengan id “selenium_logo”.

4. *Name*: Menemukan elemen yang atribut *name* yang cocok dengan nilai pencarian.

Kode 2.11: Contoh kode untuk menemukan elemen dengan atribut nama

```
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 driver = webdriver.Chrome(executable_path='D:\\Selenium\\chromedriver.exe')
4 url = "https://www.facebook.com/"
5 driver.get(url)
6 nama = driver.find_element(By.NAME, "email")
```

- Pada Kode 2.11 baris 6 mencari elemen dari atribut namanya dari situs web <https://www.facebook.com/> dengan atribut namanya adalah "email" dan disimpan dalam *string* nama.
5. *Link text*: Menemukan elemen *link* yang teksnya terlihat cocok dengan nilai pencarian.

Kode 2.12: Contoh kode untuk menemukan elemen dengan *link text*

```
13 1 from selenium import webdriver
14 2 from selenium.webdriver.common.by import By
15 3 driver = webdriver.Chrome(executable_path='D:\\Selenium\\chromedriver.exe')
16 4 url = "https://selenium.dev"
17 5 driver.get(url)
18 6 nama = driver.find_element(By.LINK_TEXT, "Documentation")
19
```

- Pada Kode 2.12 baris 6 mencari elemen *link* yang dengan nama teksnya adalah "Documentation" dari situs web <https://selenium.dev>
6. *Partial link text*: Menemukan elemen *link* yang teksnya terlihat berisi nilai pencarian. Jika beberapa elemen cocok, hanya yang pertama yang akan dipilih.

Kode 2.13: Contoh kode untuk menemukan elemen dengan *partial link text*

```
25 1 from selenium import webdriver
26 2 from selenium.webdriver.common.by import By
27 3 driver = webdriver.Chrome(executable_path='D:\\Selenium\\chromedriver.exe')
28 4 url = "https://selenium.dev"
29 5 driver.get(url)
30 6 nama = driver.find_element(By.PARTIAL_LINK_TEXT, "About_Selenium")
31
```

- Pada Kode 2.13 baris 6 mencari elemen *link* yang dengan nama teksnya adalah "About Selenium" dari situs web <https://selenium.dev>, namun ketika ada beberapa elemen yang cocok dengan nama teks yang dicari maka akan diambil yang pertamanya saja.
7. *Tag name*: Menemukan elemen yang nama tagnya cocok dengan nilai pencarian.

Kode 2.14: Contoh kode untuk menemukan elemen dengan *tag name*

```
37 1 from selenium import webdriver
38 2 from selenium.webdriver.common.by import By
39 3 driver = webdriver.Chrome(executable_path='D:\\Selenium\\chromedriver.exe')
40 4 url = "https://selenium.dev"
41 5 driver.get(url)
42 6 tag = driver.find_element(By.TAG_NAME, "h1")
43
```

- Pada Kode 2.14 baris 6 mencari elemen yang nama tagnya adalah "h1" dari situs web <https://selenium.dev> yang disimpan dengan *string tag*.
8. *XPath*: Menemukan elemen yang cocok dengan ekspresi *XML Path Language* (XPath).

Kode 2.15: Contoh kode untuk menemukan elemen dengan ekspresi XPath

```
48 1 from selenium import webdriver
49 2 from selenium.webdriver.common.by import By
50 3 driver = webdriver.Chrome(executable_path='D:\\Selenium\\chromedriver.exe')
51 4 url = "https://selenium.dev"
52 5 driver.get(url)
53 6 contoh1 = driver.find_element(By.XPATH, "//*[@id='td-cover-block-0']/div/div/div/div/h1")
54 7 contoh2 = driver.find_element(By.XPATH, "/html/body/div/main/section[1]/div/div/div/div/h1")
55
```

Pada Kode 2.15 baris 6 mencari elemen dengan XPath mulai dari nama id dari element yang dicari adalah 'td-cover-block-0', lalu diarahkan hingga tempat elemen yang dicari itu berada, dan disimpan di *string* dengan nama "contoh1". Pada baris 7 mencari elemen dengan XPath yang mulai dari struktur webnya dari atas hingga menuju tempat elemen itu berada dan disimpan di *string* dengan nama "contoh2".

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 atau tidak dilakukan secara *real time*, maka WebDriver tidak melacak status *Document Object Model* (DOM) yang aktif dan *real time*.

1. Implicit wait: memberi tahu WebDriver untuk melakukan polling DOM selama jangka waktu tertentu ketika mencoba menemukan elemen. Pengaturan awalnya adalah 0, artinya dinonaktifkan. Setelah disetel, maka *wait implicit* disetel untuk masa pakai sesi yang sudah disetel.

Kode 2.16: Contoh kode Implicit wait

```

1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 from selenium.webdriver.support.ui import WebDriverWait
4 driver = webdriver.Chrome(executable_path='D:\Selenium\chromedriver.exe')
5 driver.implicitly_wait(10)
6 url = "https://selenium.dev"
7 driver.get(url)
8 cari = driver.find_element(By.ID, "navbarDropdown")

```

Pada Kode 2.16 merupakan contoh kode *implicit wait* dimana pada baris 1 sampai 3 melakukan *import* library yang diperlukan. Baris 4 untuk menjalankan webdriver Google Chrome. Baris 5 merupakan kode *implicit wait* yang dimana kode tersebut memberikan waktu selama 10 detik untuk menemukan elemen yang ingin dicari. Baris 6 *string* dengan nama "url" diisi dengan situs web yang akan dituju. Baris 7 menggunakan *method get* yang memanggil *string* dengan nama "url" yang sudah berisi situs web yang dituju. Baris 8 adalah untuk menemukan elemen yang dicari dengan id "navbarDropdown". Jika selama waktu yang diberikan tidak dapat menemukan elemen yang dicari maka program akan mengeluarkan *output* bahwa elemen yang dicari tidak ditemukan.

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 2.17: Contoh kode Explicit wait

```

1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 from selenium.webdriver.support.ui import WebDriverWait
4 from selenium.webdriver.support import expected_conditions as EC
5 driver = webdriver.Chrome(executable_path='D:\Selenium\chromedriver.exe')
6 url = "https://selenium.dev"
7 driver.get(url)
8 try:
9     element = WebDriverWait(driver, 10).until(
10         EC.presence_of_element_located((By.ID, "navbarDropdown"))
11     )
12 finally:
13     driver.quit()

```

1 Pada Kode 2.17 merupakan contoh kode *explicit wait* dimana pada baris 1 sampai 4 melakukan
2 *import* library yang diperlukan. Baris 5 untuk menjalankan webdriver Google Chrome. Baris
3 6 *string* dengan nama “url” diisi dengan situs web yang akan dituju. Baris 7 menggunakan
4 *method get* yang memanggil *string* dengan nama “url” yang sudah berisi situs web yang
5 dituju. Baris berikutnya adalah selenium akan menunggu selama 10 detik untuk menemukan
6 elemen yang sesuai dengan id “navbarDropdown”. Jika berhasil menemukan elemen yang
7 dicari maka akan langsung masuk kondisi kode *finally* pada baris 11 dan langsung keluar
8 dari webdriver Google Chrome, Jika tidak ada elemen yang ditemukan selama waktu yang
9 diberikan maka program memberikan *output TimeoutException* dan akan masuk ke kode baris
10 11 serta langsung keluar dari webdriver Google Chrome.

BAB 3

ANALISIS

Bab ini berisi analisis yang digunakan pada skripsi ini, analisis hasil survei perekaman kehadiran daring dan luring, analisis alur perekaman kehadiran online, cara menerjemahkan perekaman kehadiran online ke dalam selenium, dan analisis program sejenis.

3.1 Analisis Hasil Survei Perekaman Kehadiran Daring dan Luring

Survei perekaman kehadiran daring dan luring dilakukan untuk mengetahui berapa lama waktu yang dibutuhkan untuk melakukan perekaman kehadiran secara daring maupun luring. Survei ini diberikan kepada mahasiswa dan dosen Teknik Informatika Universitas Katolik Parahyangan. Hasil survei menunjukkan bahwa waktu yang dibutuhkan untuk perekaman kehadiran secara luring lebih cepat bagi para mahasiswa maupun dosen dibandingkan waktu yang dibutuhkan untuk perekaman kehadiran secara daring.

3.1.1 Hasil Survei Mahasiswa

Berdasarkan hasil survei yang telah diterima dari 21 orang responden yang merupakan mahasiswa Teknik Informatika Universitas Katolik Parahyangan yang terdiri dari mahasiswa angkatan 2017 sampai 2019, dengan pertanyaan yang diajukan kepada responden dan rangkuman jawaban hasil survei sebagai berikut:

1. Berapa detik perkiraan waktu interaksi yang Anda butuhkan untuk melakukan perekaman kehadiran daring di <https://studentportal.unpar.ac.id/>, mulai dari membuka *browser*, lalu masuk ke <https://studentportal.unpar.ac.id/>, lalu mengklik tombol presensi?



Gambar 3.1: Histogram Waktu Perekaman Kehadiran Daring Mahasiswa

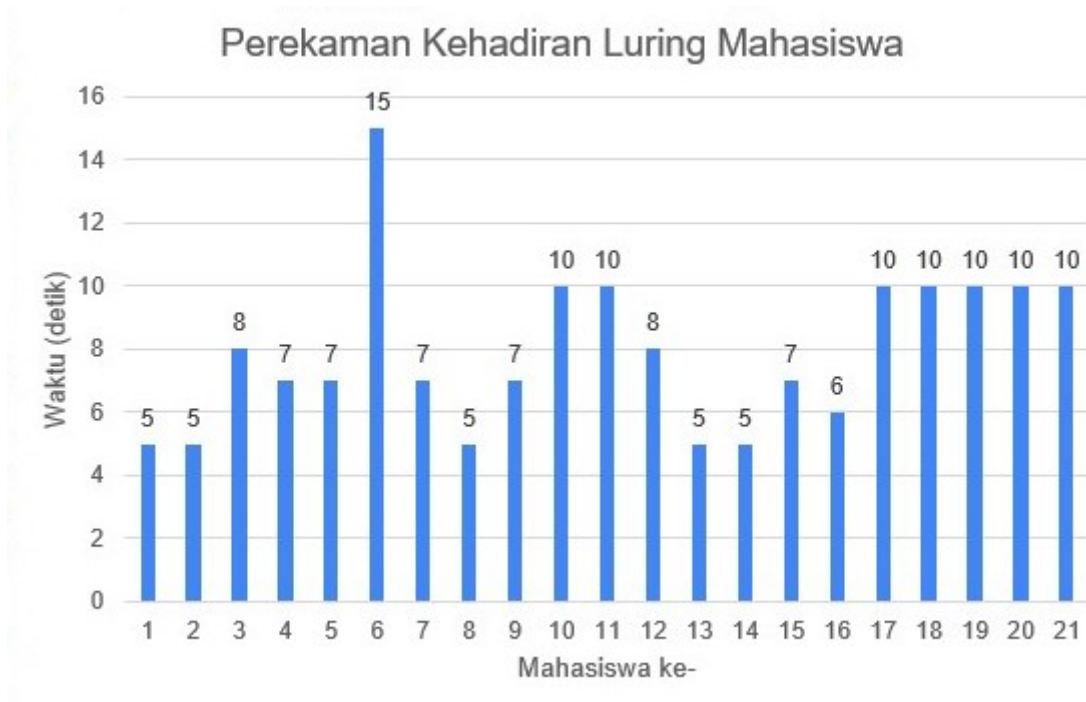
Pada Gambar 3.1 merupakan visualisasi dari hasil survei mengenai lama waktu yang dibutuhkan dari 21 mahasiswa untuk melakukan perekaman kehadiran secara daring. Jawaban dari 21 orang responden adalah mulai dari waktu paling cepat 10 detik hingga waktu paling lama 600 detik.

Jumlah Responden	Waktu Perekaman Kehadiran Daring
1 orang	10 detik
1 orang	13 detik
5 orang	15 detik
2 orang	17 detik
2 orang	18 detik
3 orang	20 detik
1 orang	25 detik
1 orang	30 detik
2 orang	45 detik
1 orang	50 detik
1 orang	300 detik
1 orang	600 detik

Jika dihitung rata-rata waktu yang dibutuhkan untuk melakukan perekaman kehadiran daring bagi para mahasiswa adalah 63 detik.

2. **Berapa detik perkiraan waktu interaksi yang Anda butuhkan untuk melakukan perekaman kehadiran luring menggunakan metode tanda tangan seperti pembelajaran di kelas, mulai dari mengambil kertas absen, lalu tanda tangan, lalu**

memberikannya ke rekan di sebelah anda?



Gambar 3.2: Histogram Waktu Perekaman Kehadiran Daring Mahasiswa

Pada Gambar 3.2 merupakan visualisasi dari hasil survei mengenai lama waktu yang dibutuhkan dari 21 mahasiswa untuk melakukan perekaman kehadiran secara luring. Jawaban dari 21 orang responden adalah mulai dari waktu paling cepat 5 detik hingga waktu paling lama 15 detik.

Jumlah Responden	Waktu Perekaman Kehadiran Luring
5 orang	5 detik
1 orang	6 detik
5 orang	7 detik
2 orang	8 detik
7 orang	10 detik
1 orang	15 detik

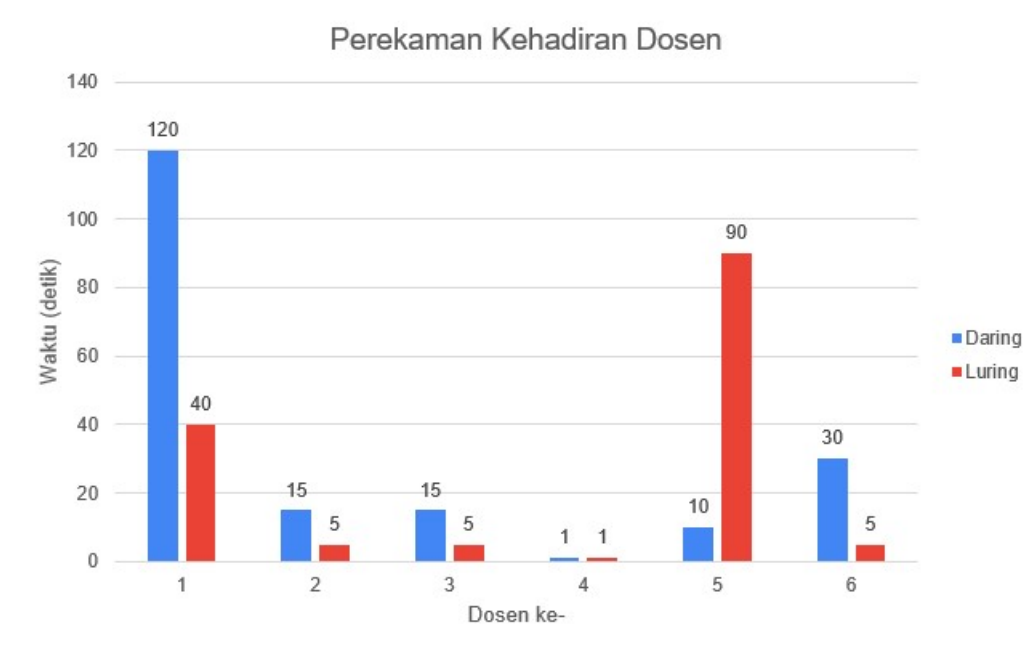
Jika dihitung rata-rata waktu yang dibutuhkan untuk melakukan perekaman kehadiran luring bagi para mahasiswa adalah 7,95 detik.

Kesimpulan dari hasil survei mahasiswa menunjukkan bahwa rata-rata waktu yang dibutuhkan secara luring adalah 7,95 detik lebih cepat dibandingkan dengan rata-rata waktu yang dibutuhkan secara daring adalah waktu 63 detik.

3.1.2 Hasil Survei Dosen

Berdasarkan hasil survei yang telah diterima dari 6 orang responden yang merupakan dosen Teknik Informatika Universitas Katolik Parahyangan, dengan pertanyaan yang diajukan kepada responden

dan rangkuman jawaban hasil survei sebagai berikut:



Gambar 3.3: Histogram Perbandingan Waktu Perekaman Kehadiran Dosen

Pada Gambar 3.3 merupakan visualisasi dari perbandingan waktu yang dibutuhkan dari 6 dosen Teknik Informatika Universitas Katolik Parahyangan. Grafik warna biru menjelaskan perekaman kehadiran secara daring dan warna merah menjelaskan perekaman kehadiran secara luring. Mayoritas menyatakan bahwa perekaman kehadiran luring lebih cepat dibandingkan daring, hanya dosen ke-4 yang menyatakan waktunya sama dan dosen ke-5 yang menyatakan waktu perekaman kehadiran daring lebih cepat dibandingkan luring.

1. **Berapa detik perkiraan waktu interaksi yang Anda butuhkan untuk melakukan perekaman kehadiran daring di <https://akuhadir.unpar.ac.id> ?**

Jawaban dari 6 orang responden adalah mulai dari waktu paling cepat 1 detik hingga waktu paling lama 120 detik. Hal tersebut dapat dilihat pada Gambar 3.3.

Jumlah Responden	Waktu Perekaman Kehadiran Daring
1 orang	1 detik
1 orang	10 detik
2 orang	15 detik
1 orang	30 detik
1 orang	120 detik

Jika dihitung rata-rata waktu yang dibutuhkan untuk melakukan perekaman kehadiran daring bagi para dosen adalah 31,83 detik.

2. **Berapa detik perkiraan waktu interaksi yang Anda butuhkan untuk melakukan perekaman kehadiran luring menggunakan metode fingerprint?**

Jawaban dari 6 orang responden adalah mulai dari waktu paling cepat 1 detik hingga waktu paling lama 90 detik. Hal tersebut dapat dilihat pada Gambar 3.3.

Jumlah Responden	Waktu Perekaman Kehadiran Luring
1 orang	1 detik
3 orang	5 detik
1 orang	40 detik
1 orang	90 detik

Jika dihitung rata-rata waktu yang dibutuhkan untuk melakukan perekaman kehadiran luring bagi para dosen adalah 24,33 detik.

Kesimpulan dari hasil survei dosen menunjukkan bahwa rata-rata waktu yang dibutuhkan secara luring adalah 24,33 detik lebih cepat dibandingkan dengan rata-rata waktu yang dibutuhkan secara daring adalah waktu 31,83 detik.

3.2 Analisis Alur Perekaman Kehadiran Online

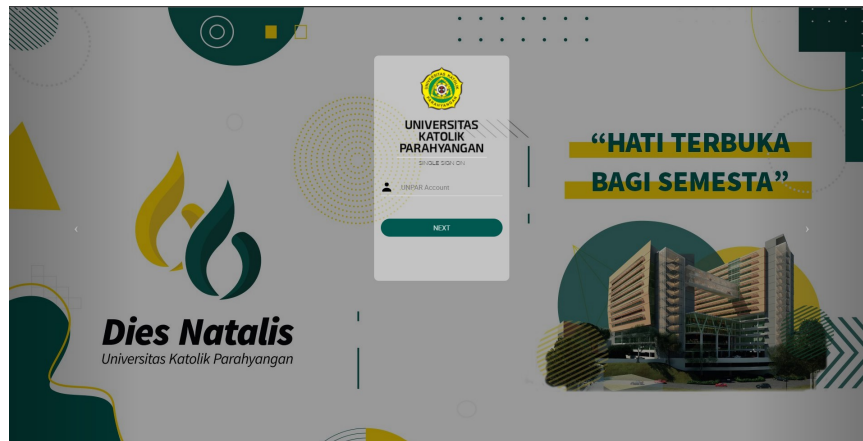
Portal Akademik Mahasiswa Universitas Katolik Parahyangan yang terbaru sejak 2020 sudah dapat melakukan perekaman kehadiran secara online untuk setiap mata kuliah yang diambil. Berikut ini adalah alur untuk melakukan perekaman kehadiran online melalui Portal Akademik Mahasiswa Universitas Katolik Parahyangan:

1. Melakukan akses Portal Akademik Mahasiswa yang dapat diakses melalui <https://studentportal.unpar.ac.id/>.



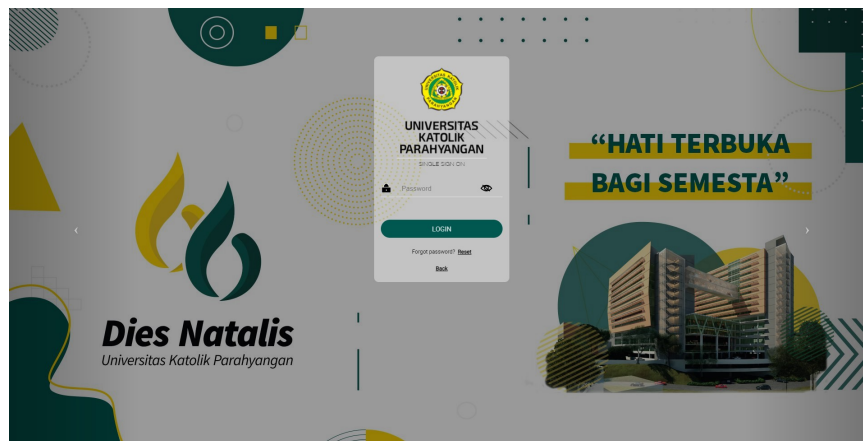
Gambar 3.4: Tampilan halaman awal Portal Akademik Mahasiswa

2. Menekan tombol “Login” yang sudah tersedia agar dapat masuk ke dalam Portal Akademik Mahasiswa, dapat dilihat pada Gambar 3.4.
3. Memasukan *email* mahasiswa.



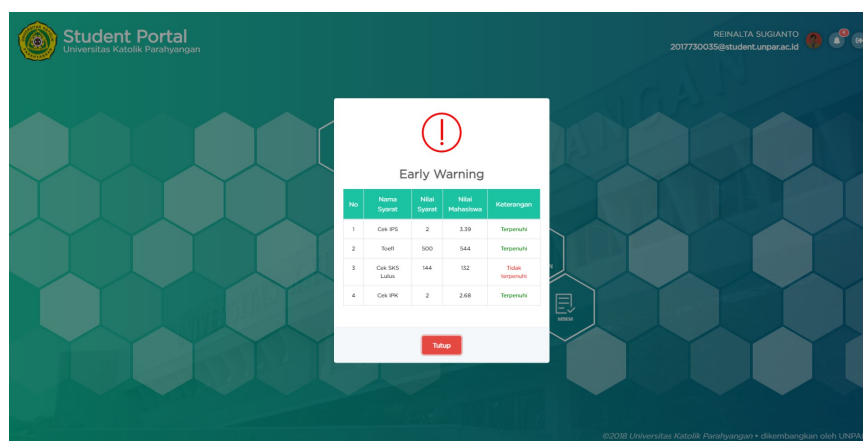
Gambar 3.5: Tampilan halaman Portal Akademik Mahasiswa untuk memasukan *email*

- 1 4. Menekan tombol “NEXT” setelah memasukan *email*, dapat dilihat pada Gambar 3.5.
- 2 5. Memasukan *password* milik mahasiswa.



Gambar 3.6: Tampilan halaman Portal Akademik Mahasiswa untuk memasukan *password*

- 3 6. Menekan tombol “LOGIN” setelah memasukan *password*, dapat dilihat pada Gambar 3.6.
- 4 7. Menekan tombol “Tutup” jika muncul peringatan atau langsung menekan tombol berbentuk
- 5 heksagon “JADWAL & KEHADIRAN” jika tidak muncul peringatan.



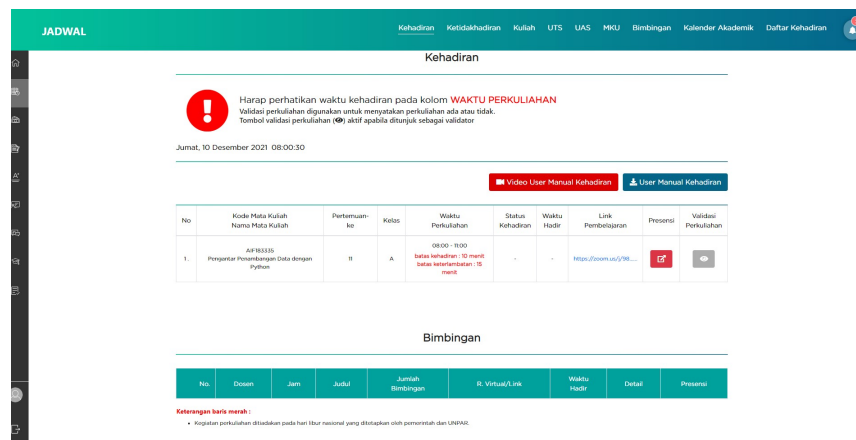
Gambar 3.7: Tampilan peringatan pada halaman Portal Akademik Mahasiswa



Gambar 3.8: Tampilan halaman Portal Akademik Mahasiswa setelah Berhasil *Login*

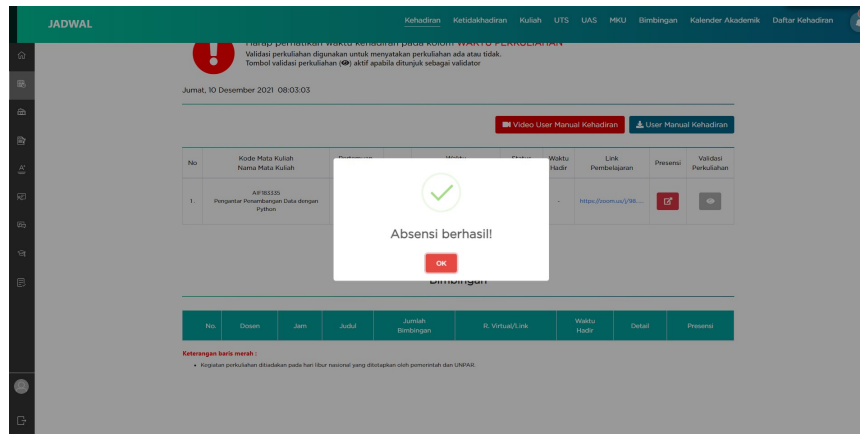
Pada Gambar 3.7 merupakan sebuah peringatan yang terkadang muncul menjelang berakhirnya suatu semester untuk melihat status kebutuhan mahasiswa untuk lulus, sehingga perlu menekan tombol “Tutup” terlebih dahulu untuk menekan tombol berbentuk heksagon “JADWAL & KEHADIRAN” seperti pada Gambar 3.8. Jika tidak terjadi peringatan seperti pada Gambar 3.7, maka dapat langsung menekan tombol berbentuk heksagon “JADWAL & KEHADIRAN” seperti pada Gambar 3.8.

8. Menekan tombol berwarna merah pada kolom bagian persensi dari tabel jadwal kehadiran mata kuliah.



Gambar 3.9: Tampilan halaman Portal Akademik Mahasiswa untuk Melakukan Absen

9. Menekan tombol “OK” ketika muncul pemberitahuan setelah berhasil melakukan presensi.



Gambar 3.10: Tampilan Pemberitahuan Absensi Berhasil

3.3 Cara Menerjemahkan Perekaman Kehadiran Online ke dalam Selenium

Otomatisasi perekaman kehadiran online ini akan menggunakan selenium, sehingga perlu diterjemahkan dari cara perekaman kehadiran online secara normal ke dalam selenium. Membuka situs web <https://studentportal.unpar.ac.id/> menggunakan selenium adalah dengan menggunakan *method get()*. Setiap tombol yang ingin ditekan akan diambil elemennya agar dapat diotomatisasikan dengan selenium. Pada *browser* Google Chrome, cara mendapatkan setiap elemen yang dibutuhkan adalah dengan melakukan *inspect* elemen pada bagian yang ingin diambil elemennya. Elemen yang ingin diambil dapat dilakukan dengan berbagai macam cara seperti yang sudah dijelaskan pada Bab 2.2. Beberapa faktor yang dapat dijadikan acuan untuk memilih cara untuk mengambil elemen dapat dilihat dari sebagai berikut:

1. Sederhana

Semakin pendek penulisan *query selector* semakin baik dan stabil, misalnya mengambil elemen dengan *CSS selector* yang namanya “#username”.

2. Mudah dimengerti dan dibaca

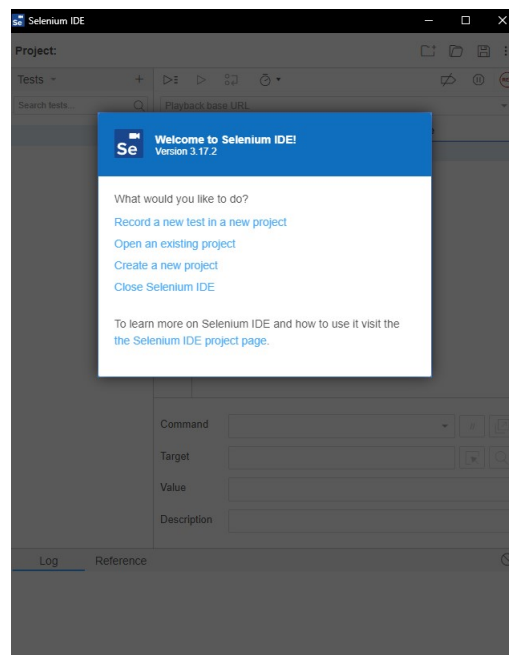
Menulis *query selector* yang mudah dibaca dan dimengerti sehingga lebih mudah untuk dipahami, contohnya “#login-button” yang artinya memilih elemen tombol untuk *login*. Tidak disarankan menulis *query selector* yang panjang atau sulit dibaca, contohnya mengambil elemen dengan cara XPath seperti yang sudah ditulis pada Bab 2.2 dengan kode program 2.15.

Pemilihan cara pengambilan elemen yang diutamakan adalah dengan mengambil elemen berdasarkan *CSS selector*, tetapi tidak menutup kemungkinan menggunakan cara yang lain untuk menemukan suatu elemen. Jika mengambil elemen berdasarkan *CSS selector* tidak perlu khawatir jika struktur HTML diubah, karena *CSS selector* sangat jarang diubah saat melakukan pembaharuan pada suatu situs web. Dalam melakukan otomatisasi perekaman kehadiran online pasti perlu memasukan *email* dan *password*, sehingga untuk memasukan hal tersebut perlu menggunakan *method sendKeys()*. Memasukan *email* dan *password* ini tidak langsung dimasukan ke dalam programnya, tetapi melalui file konfigurasi yang diisi *email* dan *password*, lalu dipanggil ke kode programnya.

3.4 Analisis Program Sejenis

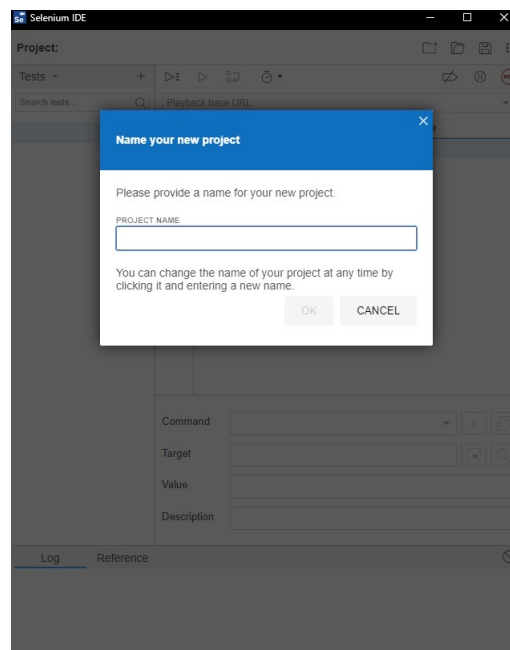
Selenium IDE merupakan program *open source* untuk otomatisasi di web. Selenium IDE dapat di *install* di browser, contohnya di Google Chrome yang setelah di *install* akan menjadi *extensions*. *Extensions* di Google Chrome adalah sebuah aplikasi kecil yang dapat dijalankan pada Google Chrome itu sendiri. Berikut ini langkah-langkah untuk melakukan otomatisasi menggunakan Selenium IDE:

1. Membuka Selenium Ide yang tersimpan di *Extensions* pada Google Chrome.
2. Memilih menu *Record a new test in a new project* (merekam tes baru untuk proyek baru).



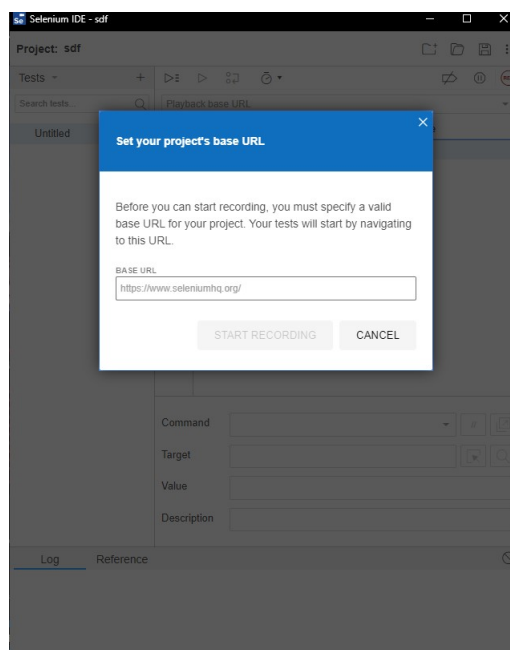
Gambar 3.11: Tampilan Menu Awal Selenium IDE

- 1 3. Memasukan nama proyek, lalu tekan tombol “OK”.



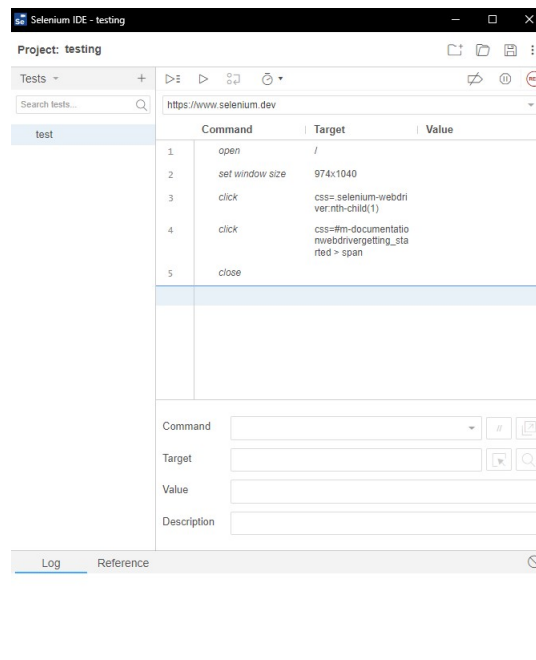
Gambar 3.12: Tampilan Memasukan Nama Proyek

- 2 4. Memasukan situs web, lalu menekan tombol “START RECORDING”



Gambar 3.13: Tampilan Memasukan Situs Web

- 3 Setelah menekan tombol “*START RECORDING*” seperti pada Gambar 3.13, maka akan
- 4 langsung muncul *windows* Google Chrome baru yang langsung menuju situs web yang sudah
- 5 dimasukan tadi.
- 6 5. Melakukan apa yang ingin diotomatisasikan di *windows* Google Chrome baru yang sudah
- 7 menuju situs web hingga selesai dan menutup *windows* Google Chrome.



Gambar 3.14: Tampilan Otomatisasi pada Selenium IDE

- 1 Pada Gambar 3.14 menunjukan hasil yang sudah terekam dari apa yang sudah dilakukan
2 pada situs web yang ingin diotomatisasikan.

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.