

**SKRIPSI**

**PEREKAMAN KEHADIRAN DARING OTOMATIS**



**Reinalta Sugianto**

**NPM: 2017730035**

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



# DAFTAR ISI

<b>DAFTAR ISI</b>	<b>iii</b>
<b>DAFTAR GAMBAR</b>	<b>v</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	2
1.3 Tujuan . . . . .	2
1.4 Batasan Masalah . . . . .	2
1.5 Metodologi . . . . .	3
1.6 Sistematika Pembahasan . . . . .	3
<b>2 LANDASAN TEORI</b>	<b>5</b>
2.1 Portal Akademik Mahasiswa 2018 . . . . .	5
2.2 AKUHADIR 2.1 . . . . .	10
2.3 <i>Cascading Style Sheets (CSS) Selector</i> . . . . .	12
2.4 Selenium . . . . .	13
2.5 WebDriver . . . . .	14
2.5.1 Browser . . . . .	14
2.5.2 Menemukan elemen . . . . .	15
2.5.3 Interaksi Elemen . . . . .	17
2.5.4 Waits . . . . .	18
2.6 <i>Library</i> Python . . . . .	19
2.6.1 <i>Configuration File Parser</i> (ConfigParser) . . . . .	20
2.6.2 <i>Library</i> Tkinter . . . . .	20
2.6.3 <i>Library</i> OS . . . . .	21
<b>3 ANALISIS</b>	<b>23</b>
3.1 Analisis Hasil Survei Perekaman Kehadiran Daring dan Luring . . . . .	23
3.1.1 Hasil Survei Mahasiswa . . . . .	23
3.1.2 Hasil Survei Dosen . . . . .	26
3.2 Analisis Alur Perekaman Kehadiran Online . . . . .	27
3.3 Cara Menerjemahkan Perekaman Kehadiran Online ke dalam Selenium . . . . .	30
3.4 Analisis Program Sejenis . . . . .	31
<b>4 PERANCANGAN</b>	<b>35</b>
4.1 Diagram Aktivitas . . . . .	35
4.2 Masukan Perangkat Lunak . . . . .	37
<b>5 IMPLEMENTASI DAN PENGUJIAN</b>	<b>39</b>
5.1 Implementasi . . . . .	39
5.1.1 Lingkungan Implementasi . . . . .	39
5.1.2 Hasil Implementasi . . . . .	39

5.2	Pengujian . . . . .	41
5.2.1	Pengujian Fungsional . . . . .	41
5.2.2	Pengujian Eksperimental . . . . .	41
<b>6</b>	<b>KESIMPULAN DAN SARAN</b>	<b>43</b>
6.1	Kesimpulan . . . . .	43
6.2	Saran . . . . .	43
	<b>DAFTAR REFERENSI</b>	<b>45</b>
<b>A</b>	<b><i>File</i> MASUKAN UNTUK PERANGKAT LUNAK</b>	<b>47</b>
A.1	<i>File</i> Konfigurasi . . . . .	47
<b>B</b>	<b>KODE PROGRAM PERANGKAT LUNAK PEREKAMAN KEHADIRAN DARING OTOMATIS</b>	<b>49</b>

## 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
2.11	Tampilan awal halaman AKUHADIR	10
2.12	Tampilan menu WFH	10
2.13	Tampilan konfirmasi check in AKUHADIR	11
2.14	Tampilan halaman check out AKUHADIR	11
2.15	Alur Komunikasi WebDriver dengan Browser	13
2.16	Contoh File Konfigurasi Sederhana	20
2.17	Contoh Informasi dari <i>Message Box</i>	20
2.18	Contoh Warning dari <i>Message Box</i>	21
3.1	Histogram Waktu Perekaman Kehadiran Daring Mahasiswa	24
3.2	Histogram Waktu Perekaman Kehadiran Luring Mahasiswa	25
3.3	Histogram Waktu Perekaman Kehadiran Daring Dosen	26
3.4	Histogram Waktu Perekaman Kehadiran Luring Dosen	27
3.5	Tampilan halaman awal Portal Akademik Mahasiswa	28
3.6	Tampilan halaman Portal Akademik Mahasiswa untuk memasukan <i>email</i>	28
3.7	Tampilan halaman Portal Akademik Mahasiswa untuk memasukan <i>password</i>	28
3.8	Tampilan peringatan pada halaman Portal Akademik Mahasiswa	29
3.9	Tampilan halaman Portal Akademik Mahasiswa setelah Berhasil <i>Login</i>	29
3.10	Tampilan halaman Portal Akademik Mahasiswa untuk Melakukan Absen	30
3.11	Tampilan Pemberitahuan Absensi Berhasil	30
3.12	Tampilan Menu Awal Selenium IDE	31
3.13	Tampilan Memasukan Nama Proyek	32
3.14	Tampilan Memasukan Situs Web	32
3.15	Tampilan Otomatisasi pada Selenium IDE	33
4.1	Diagram Aktivitas untuk <i>Setup</i> Menjalankan Program	36
4.2	Diagram Aktivitas Perangkat Lunak Absen Daring Otomatis	37
4.3	Tampilan Melakukan <i>Inspect Element</i>	38
5.1	Tampilan <i>Command Prompt</i> dengan <i>Directory File</i>	40
5.2	Tampilan Notifikasi Berhasil Absen	40
5.3	Tampilan Notifikasi Gagal Absen	40
5.4	Diagram Lingkaran Kesetujuan Perangkat Lunak Tidak <i>Error</i> atau <i>Crash</i>	42

5.5	Diagram Lingkaran Kesetujuan Perangkat Lunak Menghemat Waktu Interaksi dengan Browser . . . . .	42
-----	---	----

# 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) <sup>1</sup>. 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 memasukkan *email* dan *password* hingga akhirnya melakukan perekaman kehadiran.

Selenium adalah *open-source framework* pengujian otomatisasi untuk aplikasi web[1]. Cara kerja Selenium untuk melakukan otomatisasi pada browser web adalah seperti menirukan interaksi pengguna dengan browser, yang nantinya akan dijalankan otomatis oleh Selenium. WebDriver adalah *Application Programming Interface* (API) yang berfungsi menghubungkan Selenium dengan browser web, sehingga Selenium dapat mengontrol atau melakukan otomatisasi pada browser web. WebDriver ini seolah-olah membuat pengguna secara langsung mengoperasikan *browser*, padahal dijalankan secara otomatis langsung oleh WebDriver tersebut. Selenium ini tersedia untuk bahasa pemrograman Ruby, Java, Python, C#, dan JavaScript. Pembuatan Perekaman kehadiran daring otomatis ini akan menggunakan Selenium dengan bahasa pemrograman Python.

---

<sup>1</sup>Pandemi Covid-19 di Indonesia [https://id.wikipedia.org/wiki/Pandemi\\_Covid-19\\_di\\_Indonesia](https://id.wikipedia.org/wiki/Pandemi_Covid-19_di_Indonesia)

Proses perekaman kehadiran daring di UNPAR harus melakukan beberapa hal untuk dapat melakukan perekaman kehadiran daring untuk mata kuliah yang diambil. Berikut ini hal-hal yang perlu dilakukan untuk melakukan perekaman kehadiran daring di UNPAR:

1. Membuka browser.
2. Membuka situs <https://studentportal.unpar.ac.id>.
3. Mengisi *email* dan *password* mahasiswa.
4. Menuju ke halaman web untuk perekaman kehadiran mahasiswa.
5. Melakukan rekam kehadiran.

Pada skripsi ini, akan dibuat sebuah perangkat lunak yang dapat melakukan perekaman kehadiran otomatis dengan sistem menerima rangsangan satu “klik” yang dapat menjalankan langkah-langkah perekaman kehadiran daring manual secara otomatis pada situs <https://studentportal.unpar.ac.id>, sehingga perangkat lunak yang dibuat ini menjalankan perintah yang biasa dilakukan mahasiswa lakukan secara manual untuk melakukan perekaman kehadiran daring menjadi otomatis dilakukan oleh perangkat lunak tersebut. Perangkat lunak ini bertujuan agar mahasiswa dapat melakukan perekaman kehadiran secara online di situs web Portal Akademik Mahasiswa UNPAR dengan lebih mudah dikarenakan mahasiswa hanya perlu menjalankan perangkat lunak tersebut untuk melakukan perekaman kehadiran 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 mendekati atau menyamai waktu perekaman kehadiran secara luring. Dikarenakan terbimbing tidak memiliki akses ke <https://akuhadir.unpar.ac.id> situs perekaman kehadiran milik dosen, maka terbimbing mensimulasikan dengan Portal Akademik Mahasiswa dan kemudian Pembimbing mengubah aksesnya ke situs perekaman kehadiran milik dosen.

## 1.2 Rumusan Masalah

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

1. Bagaimana cara membangun program Perekaman Kehadiran Daring Otomatis?
2. Bagaimana cara mengurangi waktu interaksi dengan aplikasi atau situs web untuk merekam kehadiran secara otomatis?

## 1.3 Tujuan

Tujuan yang ingin dicapai dari penulisan skripsi ini sebagai berikut :

1. Membangun program menggunakan Selenium WebDriver.
2. Membuat program yang mampu menerima rangsangan satu tombol untuk melakukan perekaman daring otomatis 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.

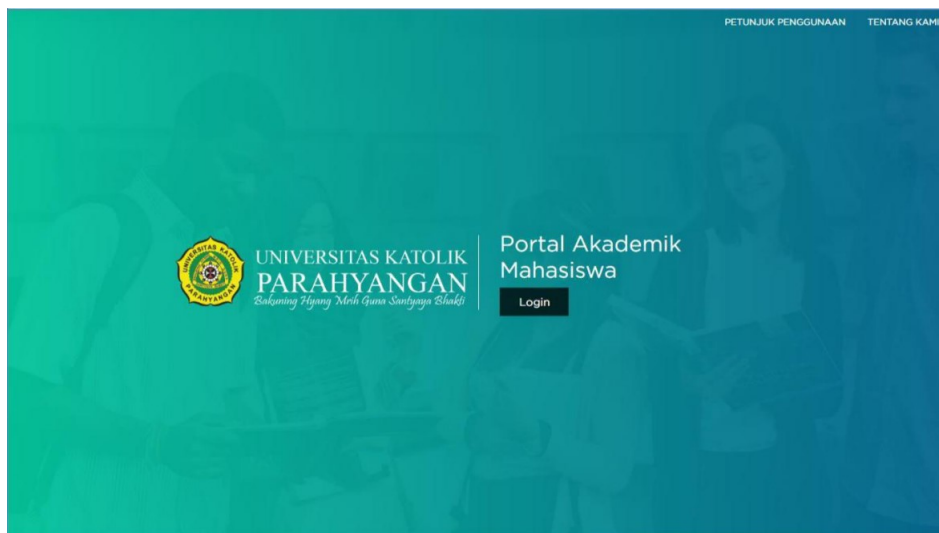


## BAB 2

### 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.

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,-

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.

The screenshot shows a web interface for payments. At the top, there's a summary section with fields for 'PENGEMBALAN' (Return) and 'TOTAL' (Total), both showing 'Rp. 0,-'. Below this is a section titled 'Riwayat Pembayaran' (Payment History) which contains a table with four columns: 'Tanggal Pembayaran' (Payment Date), 'Jumlah Pembayaran' (Payment Amount), 'No. Transaksi' (Transaction No.), and 'Bank'. The table lists three transactions: one on 02 Januari 2018 for Rp. 2.890.000,- via BRI, one on 18 Februari 2018 for Rp. 5.300.000,- via BEA UNPAR, and one on 07 Maret 2018 for Rp. 3.488.000,- via BRI. Below the table is a 'Keterangan' (Notes) section with instructions on how to make payments, including options for cash, BRI bank, and ATM.

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. 3.488.000,-	0547894	BRI

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: A. Bank BRI (including cash payment at a teller), B. Kartu ATM Bank BRI & Mesin BRI (including using a virtual account), and C. Kartu ATM Bank Lain & Mesin BRI (including using a virtual account). It also includes instructions for D. Kartu ATM & Mesin Bank Lainnya (including using a virtual account). The instructions are numbered 1 through 6, detailing the steps from selecting the payment method to confirming the transaction.

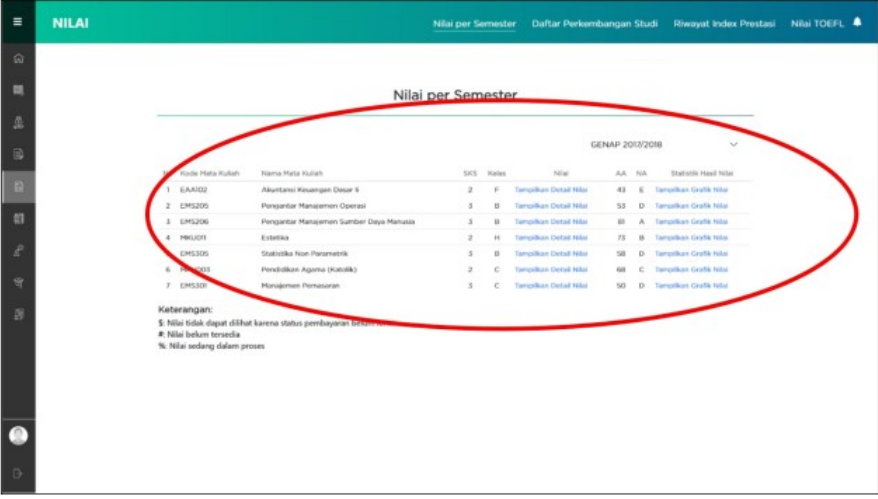
Gambar 2.8: Tampilan halaman pembayaran bagian Keterangan

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**

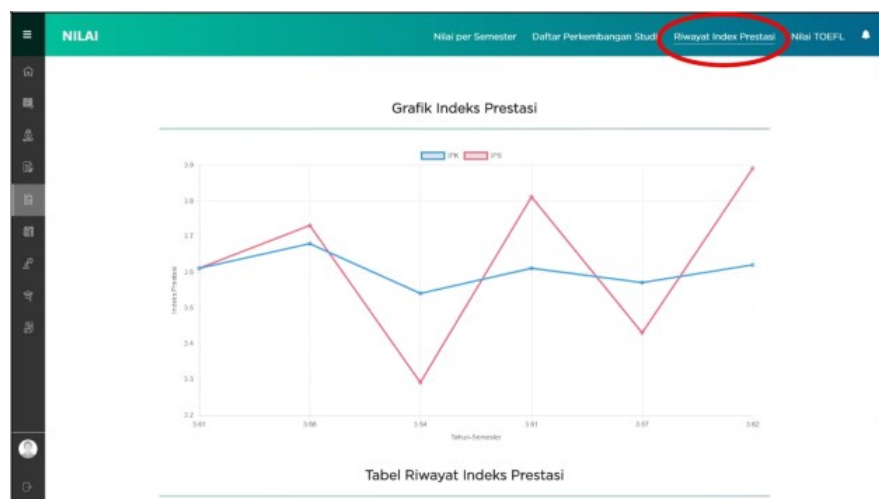
GENAP 2017/2018

No	Kode Mata Kuliah	Nama Mata Kuliah	SKS	Kelas	Nilai	AA	NA	Statistik Hasil Nilai
1	EAAD02	Akuntansi Keuangan Dasar 4	2	F	Tampilkan Detail Nilai	43	E	Tampilkan Grafik Nilai
2	EMIS205	Pengantar Manajemen Operasi	3	B	Tampilkan Detail Nilai	53	D	Tampilkan Grafik Nilai
3	EMIS206	Pengantar Manajemen Sumber Daya Manusia	3	B	Tampilkan Detail Nilai	61	A	Tampilkan Grafik Nilai
4	EMIS201	Etika	2	H	Tampilkan Detail Nilai	75	B	Tampilkan Grafik Nilai
5	EMIS305	Statistika Non-Parametrik	3	B	Tampilkan Detail Nilai	58	D	Tampilkan Grafik Nilai
6	EMIS202	Pengalihan Algoritma (Kalkulus)	2	C	Tampilkan Detail Nilai	68	C	Tampilkan Grafik Nilai
7	EMIS301	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

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

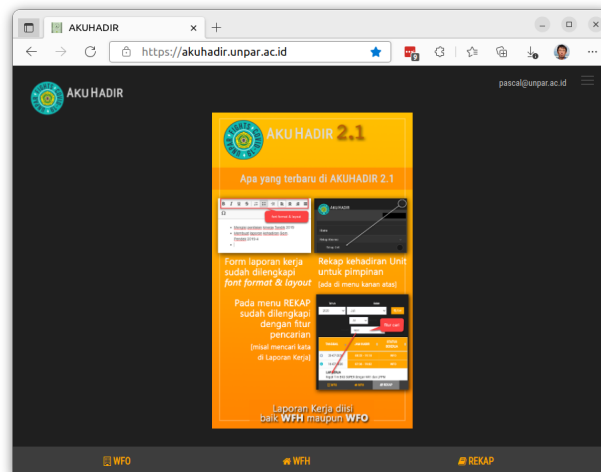


Gambar 2.10: Tampilan halaman nilai bagian Riwayat Index Prestasi

## 2.2 AKUHADIR 2.1

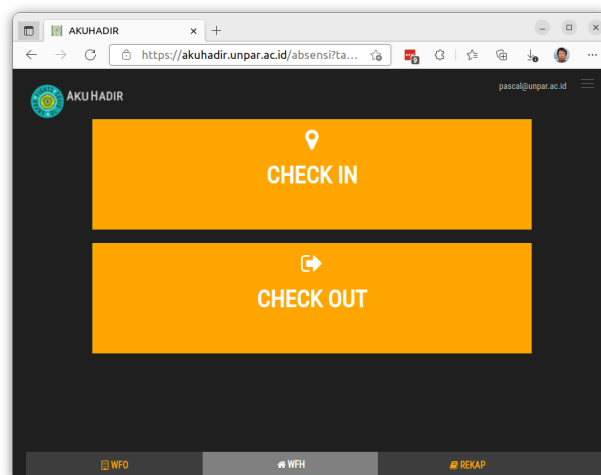
*Subbab ini ditulis oleh dosen pembimbing.*

AKUHADIR adalah sebuah portal web yang dibuat bagi pegawai UNPAR dalam melaporkan kehadiran kerja nya secara daring. Pegawai UNPAR mencatatkan kehadirannya setiap hari pada portal tersebut (dapat diakses pada <https://akuhadir.unpar.ac.id>), sesuai surat edaran Rektor III/R/2020-07/1153 [3].



Gambar 2.11: Tampilan awal halaman AKUHADIR

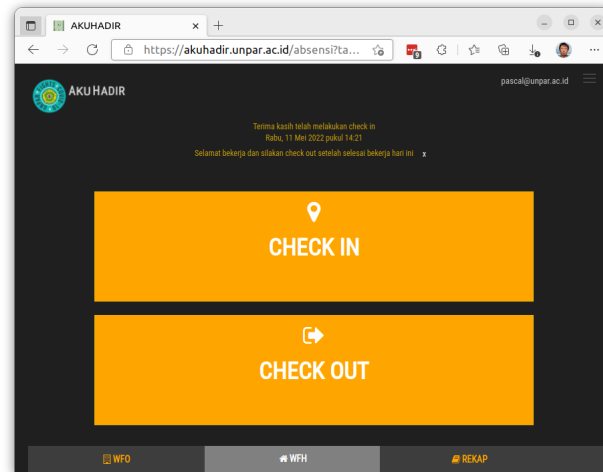
Gambar 2.11 menunjukkan halaman awal portal AKUHADIR, setelah pegawai mengautentikasi dirinya melalui SSO (*Single Sign On*) UNPAR. Kehadiran virtual dapat didaftarkan dengan memilih menu WFH di bagian bawah layar, di mana pengguna akan dibawa ke halaman lain yang menunjukkan dua tombol: satu untuk *check in* dan satu untuk *check out* (Gambar 2.12).



Gambar 2.12: Tampilan menu WFH

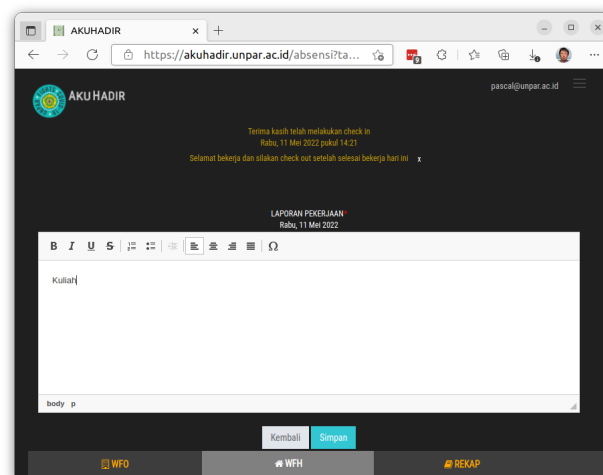
Di pagi hari sebelum memulai bekerja, pegawai mengklik menu *Check in*. Setelah tombol tersebut diklik, akan muncul pesan konfirmasi bahwa *check in* sudah berhasil dilakukan (Gambar 2.13).





Gambar 2.13: Tampilan konfirmasi check in AKUHADIR

1 Di sore hari setelah bekerja, pegawai kembali mengakses AKUHADIR, namun memilih menu  
 2 *check out*. Prosesnya mirip dengan *check in*, namun pada proses *check out* pegawai diminta untuk  
 3 menuliskan laporan pekerjaan yang sudah dilakukan pada hari tersebut dalam bentuk tulisan  
 4 (Gambar 2.14).



Gambar 2.14: Tampilan halaman check out AKUHADIR

5 Selain kedua fitur tersebut, ada beberapa fitur lain dari AKUHADIR yang tidak dijelaskan lebih  
 6 mendalam di sini karena tidak terkait erat dengan penelitian yang dilakukan:

- 7 • **WFO** untuk melihat status pelaporan bekerja dari kantor (pelaporan bekerja dari kantor  
 8 dilakukan oleh petugas keamanan UNPAR yang memindai kode batang pegawai).
- 9 • **Rekap** untuk melihat rekapitulasi pelaporan bekerja, baik WFH maupun WFO.
- 10 • **Profil** untuk menampilkan foto serta kode batang pegawai. Bisa digunakan untuk menun-  
 11 jukkan kode batang kepada petugas keamanan dalam rangka pelaporan WFO.
- 12 • **About** saat ini hanya menampilkan informasi versi dan *copyright*.

## 2.3 Cascading Style Sheets (CSS) Selector

*Cascading Style Sheets* (CSS) adalah bahasa untuk menerapkan tampilan pada sebuah halaman web, hal tersebut termasuk tata letak, warna, dan *font*[4]. CSS sudah menjadi bagian penting untuk sebuah Web. CSS *Selector* digunakan untuk menemukan atau memilih elemen HTML yang diinginkan berdasarkan *style* CSS<sup>1</sup>. Terdapat lima kategori pada CSS *Selector*:

1. *Simple selectors* merupakan pemilihan elemen dari CSS berdasarkan *name*, *id*, dan *class*. Pada Tabel 2.1 merupakan contoh cara mengambil elemen menggunakan cara *simple selector*.

Tabel 2.1: Tabel Contoh *Simple Selector*.

<b>Selector</b>	Contoh	Penjelasan
#id	#firstname	Memilih elemen dengan id = "firstname".
.class	.intro	Memilih elemen dengan <i>class</i> = "intro".
element	p	Memilih semua elemen <p>.
*	*	Memilih semua elemen.
element.class	p.intro	Memilih hanya elemen <p> pada <i>class</i> = "intro".

2. *Combinator selectors* merupakan pemilihan elemen dari CSS berdasarkan gabungan atau kombinasi antar elemen pada CSS. Pada Tabel 2.2 merupakan contoh cara mengambil elemen menggunakan cara *combinator selector*.

Tabel 2.2: Tabel Contoh *Combinator Selector*.

<b>Selector</b>	Contoh	Penjelasan
element element	div p	Memilih semua elemen <p> yang berada di dalam elemen <div>.
element > element	div > p	Memilih semua elemen <p> yang menjadi anak bagian elemen <div>.
element + element	div + p	Memilih elemen <p> pertama setelah elemen <div>.
element1 ~ element2	p ~ ul	Memilih setiap elemen <ul> yang diawali dulu oleh elemen <p>.

3. *Pseudo-class selectors* merupakan pemilihan elemen yang berada dalam kondisi khusus elemen tersebut. Penjelasan lebih detail dapat dilihat pada Tabel 2.3 merupakan contoh cara mengambil elemen menggunakan cara *pseudo-class selector*.

Tabel 2.3: Tabel Contoh *Pseudo-class Selector*.

<b>Selector</b>	Contoh	Penjelasan
:active	a:active	Memilih elemen yang memiliki link aktif.
:checked	input:checked	Memilih setiap elemen <input> yang sudah ditandai.
:disabled	input:disabled	Memilih setiap elemen <input> yang dinonaktifkan.

<sup>1</sup> CSS Selector [https://www.w3schools.com/css/css\\_selectors.asp](https://www.w3schools.com/css/css_selectors.asp)

4. *Pseudo-elements selectors* merupakan pemilih elemen berdasarkan pada letak spesifik elemen tersebut berada. Pada Tabel 2.4 merupakan contoh cara mengambil elemen menggunakan cara *pseudo-elements selector*.

Tabel 2.4: Tabel Contoh *Pseudo-elements Selector*.

<b>Selector</b>	Contoh	Penjelasan
::first-letter	p::first-letter	Memilih huruf pertama dari setiap elemen <p>.
::first-line	p::first-line	Memilih baris pertama dari setiap elemen <p>.

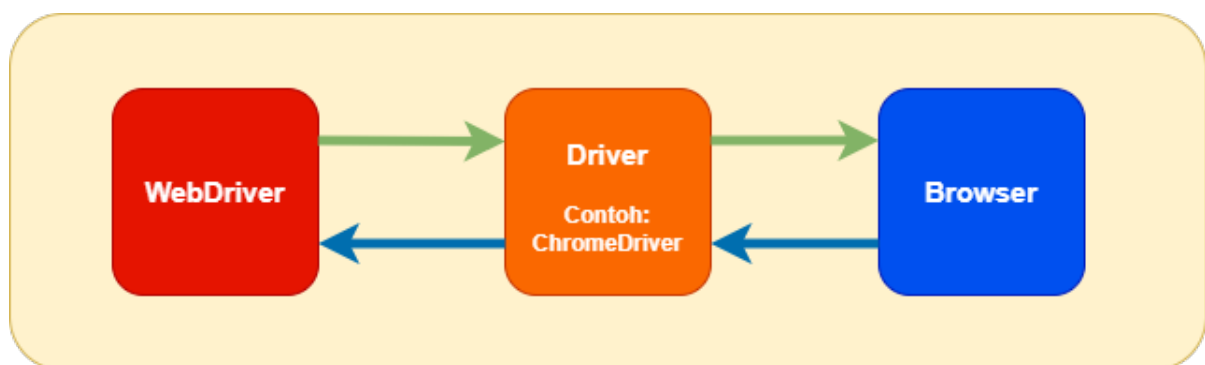
5. *Attribute selectors* merupakan pemilihan elemen berdasarkan atribut tertentu atau nilai dari atributnya. Pada Tabel 2.5 merupakan contoh cara mengambil elemen menggunakan cara *attribute selector*.

Tabel 2.5: Tabel Contoh *Attribute Selector*.

<b>Selector</b>	Contoh	Penjelasan
[attribute]	[target]	Memilih semua elemen yang mengandung atribut "target".
[attribute=value]	[target=blank]	Memilih semua elemen yang mengandung atribut "target" yang nilainya "blank".
[attribute =value]	[title =flower]	Memilih semua elemen yang "title" atributnya mengandung kata "flower".

## 2.4 Selenium

Selenium merupakan *open-source framework* untuk pengujian otomatisasi untuk browser web[1]. Cara kerja Selenium untuk melakukan otomatisasi pada browser web adalah seperti menirukan interaksi pengguna dengan browser, yang nantinya akan dijalankan otomatis oleh Selenium. Otomatisasi dengan Selenium ini dapat dilakukan pada berbagai browser yang umum banyak digunakan (Google Chrome, Safari, Opera, Firefox). Selenium ini tersedia untuk bahasa pemrograman Ruby, Java, Python, C#, dan JavaScript. Bagian inti dari Selenium adalah WebDriver, WebDriver merupakan sebuah *interface* untuk menulis suatu instruksi yang dapat dijalankan secara otomatis dan bergantian pada *browser*. Setiap browser pasti didukung oleh implementasi WebDriver tertentu, yang disebut driver. Driver adalah komponen yang bertanggung jawab untuk menghubungkan komunikasi antara Selenium dengan browser.



Gambar 2.15: Alur Komunikasi WebDriver dengan Browser

Pada Gambar 2.15. Pada Gambar 2.15 ini komunikasi WebDriver yang merupakan bagian dari Selenium memberikan perintah kepada browser melalui *driver* browser tersebut dan menerima hasil informasinya kembali melalui alur yang sama.

## 2.5 WebDriver

WebDriver adalah *Application Programming Interface* (API) yang berfungsi menghubungkan Selenium dengan browser web, sehingga Selenium dapat mengontrol atau melakukan otomatisasi pada browser web[1]. API WebDriver ini seolah-olah membuat pengguna secara langsung mengoperasikan *browser*, padahal dijalankan secara otomatis langsung oleh API WebDriver tersebut. *Driver* browser adalah komponen yang bertanggung jawab untuk menghubungkan antara Selenium dengan browser agar dapat dikendalikan. Driver browser yang tersedia untuk selenium adalah Google Chrome, Firefox, Edge, Internet Explorer, dan Safari. Agar browser dapat dibuka untuk melakukan otomatisasi, maka perlu melakukan *install* driver dari browser yang ingin digunakan. (Tanya)

### 2.5.1 Browser

Selenium Webdriver dapat mengambil informasi mengenai browser yang sedang dibuka. Informasi yang dapat diambil adalah judul dari situs web yang sedang dibuka pada browser (Kode 2.1 baris 1) dan mendapatkan alamat situs web yang sedang dibuka (Kode 2.1 baris 2).

Kode 2.1: Contoh Potongan Kode *Get Title* dan *Get Current URL*

```
driver.title
driver.current_url
```

Hal pertama yang perlu dilakukan setelah berhasil membuka browser adalah membuka situs web yang akan diotomatisasikan. Pada Kode 2.2 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.

Kode 2.2: Contoh kode Navigate to

```
from selenium import webdriver
driver = webdriver.Chrome()
url = "https://selenium.dev"
link = driver.get(url)
```

WebDriver juga dapat melakukan otomatisasi untuk keluar dari browser setelah selesai digunakan. Pada Kode 2.3 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 2.2. Baris 5 adalah *string* dengan nama *quit* yang memanggil *method quit()* yang berfungsi untuk dapat keluar dari *browser* setelah selesai digunakan.

Kode 2.3: Contoh kode Get title

```

1  from selenium import webdriver
2  driver = webdriver.Chrome()
3  url = "https://selenium.dev"
4  link = driver.get(url)
5  quit = driver.quit()

```

## 2.5.2 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 cara menemukan elemen:

- **Id**

Menemukan elemen yang atribut ID-nya cocok dengan nilai pencarian. Pada Kode 2.4 merupakan contoh untuk menemukan elemen dengan atribut ID. Baris 1 melakukan *import* webdriver terlebih dahulu. Baris 2 melakukan *import* By yang merupakan *library* yang sudah ada milik selenium untuk digunakan dalam menemukan elemen. lalu baris 3 *string* dengan nama driver memanggil webdriver yang ingin digunakan, yaitu Google Chrome untuk membuka browser-nya. Baris 4 *string* dengan nama url diisi dengan situs web yang dituju dalam contoh adalah <https://selenium.dev>. Baris 5 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. Baris 6 atau 7 merupakan contoh kode yang dapat digunakan untuk menemukan elemen berdasarkan atribut ID dengan nama id “selenium” dari situs web <https://selenium.dev>. Kode pada baris 6 dan 7 hanya berbeda cara penulisannya saja.

Kode 2.4: 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()
4  url = "https://selenium.dev"
5  driver.get(url)
6  driver.find_element(By.ID, "selenium")
7  driver.find_element_by_id("selenium")

```

- **Class name**

Menemukan elemen yang nama kelasnya berisi nilai pencarian. Pada Kode 2.5 merupakan contoh untuk menemukan elemen dengan nama kelas. Pada baris 1 sampai 5 merupakan contoh melakukan *import library* selenium yang dibutuhkan dan untuk memunculkan situs web yang ingin dijalankan dengan selenium dan sudah dijelaskan pada Kode 2.4. Baris 6 merupakan contoh kode untuk mencari elemen dengan *class name* “text-center” dan disimpan dalam *string* kelas.

Kode 2.5: 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()
4  url = "https://selenium.dev"
5  driver.get(url)
6  kelas = driver.find_elements(By.CLASS_NAME, "text-center")

```

- **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. Pada Kode 2.6 merupakan contoh untuk menemukan elemen yang cocok dengan pemilihan CSS. Baris 6 merupakan contoh kode yang disimpan dalam *string select* untuk mencari elemen berdasarkan pemilihan CSS dengan mengambil elemen dengan id “selenium\_logo”.

Kode 2.6: 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()
4 url = "https://selenium.dev"
5 driver.get(url)
6 select = driver.find_element(By.CSS_SELECTOR, "#selenium_logo")
```

- *Name*

Menemukan elemen yang atribut *name* yang cocok dengan nilai pencarian. Pada Kode 2.7 baris 6 mencari elemen dari atribut namanya dari situs web <https://www.facebook.com/> dengan atribut namanya adalah “email” dan disimpan dalam *string* nama.

Kode 2.7: 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()
4 url = "https://www.facebook.com/"
5 driver.get(url)
6 nama = driver.find_element(By.NAME, "email")
```

- *Link text*

Menemukan elemen *link* yang teksnya terlihat cocok dengan nilai pencarian. Pada Kode 2.8 baris 6 mencari elemen *link* yang dengan nama teksnya adalah “Documentation” dari situs web <https://selenium.dev>.

Kode 2.8: Contoh kode untuk menemukan elemen dengan *link text*

```
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 driver = webdriver.Chrome()
4 url = "https://selenium.dev"
5 driver.get(url)
6 nama = driver.find_element(By.LINK_TEXT, "Documentation")
```

- *Partial link text*

Menemukan elemen *link* yang teksnya terlihat berisi nilai pencarian. Jika beberapa elemen cocok, hanya yang pertama yang akan dipilih. Pada Kode 2.9 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.

Kode 2.9: Contoh kode untuk menemukan elemen dengan *partial link text*

```
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 driver = webdriver.Chrome()
4 url = "https://selenium.dev"
5 driver.get(url)
6 nama = driver.find_element(By.PARTIAL_LINK_TEXT, "About_Selenium")
```

- *Tag name*

Menemukan elemen yang nama tagnya cocok dengan nilai pencarian. Pada Kode 2.10 baris 6 mencari elemen yang nama tagnya adalah “h1” dari situs web <https://selenium.dev> yang disimpan dengan *string* tag.

Kode 2.10: Contoh kode untuk menemukan elemen dengan *tag name*

```

1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 driver = webdriver.Chrome()
4 url = "https://selenium.dev"
5 driver.get(url)
6 tag = driver.find_element(By.TAG_NAME, "h1")

```

- XPath

Menemukan elemen yang cocok dengan ekspresi *XML Path Language* (XPath). XPath adalah bahasa ekspresi yang dirancang untuk mendukung kueri atau transformasi dari dokumen XML[5]. Pada Kode 2.11 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”.

Kode 2.11: Contoh kode untuk menemukan elemen dengan ekspresi XPath

```

1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 driver = webdriver.Chrome()
4 url = "https://selenium.dev"
5 driver.get(url)
6 contoh1 = driver.find_element(By.XPATH, "//*[@id='td-cover-block-0']/div/div/div/div/h1")
7 contoh2 = driver.find_element(By.XPATH, "/html/body/div/main/section[1]/div/div/div/div/h1")

```

### 2.5.3 Interaksi Elemen

Interaksi elemen merupakan metode selenium yang dirancang untuk dapat melakukan otomatisasi seperti meniru langsung pengguna dalam melakukan sesuatu pada browser, seperti mengklik tombol, memasukan email dan *password*, atau menghapus teks sesuatu. Terdapat 5 perintah dasar yang dapat dijalankan pada sebuah elemen:

- Click

Perintah *Click* merupakan perintah dasar milik selenium untuk menekan atau mengklik secara otomatis sesuai dengan elemen yang diambil. Pada Kode 2.12 adalah contoh potongan kode program yang menggunakan perintah *Click*, hanya cukup menambahkan kode “.click()” saja pada bagian akhir saat mengambil suatu elemen.

Kode 2.12: Contoh Potongan Kode Perintah *Click* pada Suatu Elemen

```

1 btnIn = driver.find_element(By.CSS_SELECTOR, "#login-button").click()

```

- Send Keys

Perintah *Send Keys* merupakan perintah untuk mengetik sesuatu atau memasukan sesuatu dalam bentuk teks maupun angka pada suatu elemen secara otomatis. Biasanya elemen yang digunakan untuk menjalankan perintah *Send Keys* ini adalah elemen input dari formulir pada suatu situs web dengan tipe teks. Pada Kode 2.13 adalah contoh potongan kode program yang menggunakan perintah *Send Keys*, kode tersebut mengartikan bahwa program melakukan pencarian secara otomatis pada situs “http://www.google.com”, dimana elemen dengan *NAME* “q” diisi dengan nilai “webdriver” dan menekan tombol “ENTER” secara otomatis seperti pengguna menekan tombol “ENTER” manual pada *keyboard* komputer atau

laptop. Hasilnya adalah program akan membuka situs “http://www.google.com” dan sudah melakukan pencarian tentang “webdriver”.

Kode 2.13: Contoh Potongan Kode Perintah *Send Keys* pada Suatu Elemen

```
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 from selenium.webdriver.common.keys import Keys
4 driver = webdriver.Chrome()
5 driver.get("http://www.google.com")
6 driver.find_element(By.NAME, "q").send_keys("webdriver" + Keys.ENTER)
```

- *Clear*

Perintah *Clear* merupakan perintah untuk menghapus secara otomatis terhadap isi konten pada suatu elemen. Elemen yang bisa diberi perintah *Clear* adalah elemen input dari formulir pada suatu situs web dengan tipe teks. Pada Kode 2.14 adalah contoh potongan kode program yang menggunakan perintah *Clear*, kode tersebut mengartikan bahwa program setelah menggunakan perintah *Send Keys* untuk memasukkan “webdriver” pada elemen dengan *NAME* “q” untuk dicari pada situs “http://www.google.com”. Lalu dihapus dengan menggunakan perintah *Clear*.

Kode 2.14: Contoh Potongan Kode Perintah *Clear* pada Suatu Elemen

```
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 driver = webdriver.Chrome()
4 driver.get("http://www.google.com")
5 SearchInput = driver.find_element(By.NAME, "q").send_keys("webdriver")
6 SearchInput.clear()
```

- *Submit*

Perintah *Submit* pada Selenium versi 4 saat ini sudah tidak lagi diimplementasikan lagi, sehingga disarankan tidak menggunakan perintah ini lagi dan disarankan menggunakan perintah *Click* dengan memilih elemen yang tepat untuk tombol *submit* pada situs web.

- *Select List*

Perintah *Select List* ini berfungsi untuk mengambil nilai dari elemen yang berada dalam bentuk *list*.

## 2.5.4 Waits

*Waits* merupakan API pemblokiran yang dimiliki WebDriver. *Waits* ini memiliki fungsi untuk menunggu suatu perintah saat melakukan proses otomatisasi terhadap suatu situs web beres dijalankan, lalu menjalankan perintah selanjutnya.

- Implicit wait: memberi tahu WebDriver untuk menunggu selama jangka waktu tertentu ketika mencoba menemukan elemen. Pengaturan awal lama menunggu adalah 0 detik, artinya dinonaktifkan. Setelah disetel, maka *wait implicit* disetel untuk menunggu selama waktu yang sudah ditentukan.

Kode 2.15: 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()
5 driver.implicitly_wait(10)
6 url = "https://selenium.dev"
7 driver.get(url)
8 cari = driver.find_element(By.ID, "navbarDropdown")
```



Pada Kode 2.15 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.

- 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 2.16: 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()
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()

```

Pada Kode 2.16 merupakan contoh kode *explicit wait* dimana pada baris 1 sampai 4 melakukan *import* library yang diperlukan. Baris 5 untuk menjalankan webdriver Google Chrome. 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 berikutnya adalah selenium akan menunggu selama 10 detik untuk menemukan elemen yang sesuai dengan id “navbarDropdown”. Jika berhasil menemukan elemen yang dicari maka akan langsung masuk kondisi kode *finally* pada baris 11 dan langsung keluar dari webdriver Google Chrome, Jika tidak ada elemen yang ditemukan selama waktu yang diberikan maka program memberikan *output TimeoutException* dan akan masuk ke kode baris 11 serta langsung keluar dari webdriver Google Chrome.

## 2.6 Library Python

Bahasa pemrograman Python memiliki banyak library yang dapat dipakai. *Library* Python adalah kumpulan modul yang berisi kumpulan kode yang dapat digunakan secara berulang kali di berbagai program. *Library* Python ini membuat programmer menjadi mudah dan sederhana, karena tidak perlu menuliskan kode yang sama secara berulang. Pada subbab ini dijelaskan *library* python yang digunakan untuk pembuatan perangkat lunak perekaman kehadiran daring otomatis.

### 2.6.1 Configuration File Parser (ConfigParser)

ConfigParser merupakan sebuah modul yang sudah tersedia pada bahasa pemrograman Python dan mengimplementasikan bahasa konfigurasi dasar[6]. Penggunaan ConfigParser pada bahasa pemrograman python perlu melakukan *import library* terlebih dahulu, sehingga dapat digunakan. Struktur dari file konfigurasi terdiri dari *section* dari file tersebut dan masing-masing *key* dan *value*. ConfigParser ini dapat membaca dan menulis file tersebut.

```
[Database]
Nama      = Budi
Email     = Budi@gmail.com
Status    = Admin
```

Gambar 2.16: Contoh File Konfigurasi Sederhana

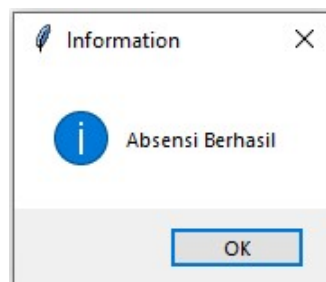
Pada Gambar 2.16 merupakan contoh file konfigurasi sederhana. Tulisan “[Database]” pada gambar 2.16 merupakan *section*, untuk tulisan yang berwarna biru merupakan bagian *key*, dan tulisan yang berwarna merah merupakan bagian *value*. Berikut ini penjelasan *key* dan *value* dari Gambar 2.16:

- Pada file konfigurasi dengan *section* “[Database]” memiliki 3 *key* dan *value*.
- *Key* pertama adalah “Nama” dengan *value* “Budi”.
- *Key* kedua adalah “Email” dengan *value* “Budi@gmail.com”.
- *Key* ketiga adalah “Status” dengan *value* “Admin”.

### 2.6.2 Library Tkinter

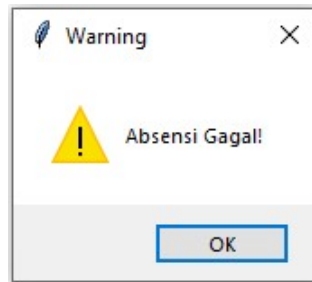
Library Tkinter (Tk interface) adalah *library* standar dari Python untuk membuat aplikasi Python dengan antarmuka yang mudah diprogram. Tk merupakan sebuah fasilitas untuk grafis dan *scripting* yang dikembangkan oleh John Ousterhout [7]. Modul yang digunakan dari *library* Tkinter untuk pembuatan perangkat lunak ini adalah `tkinter.messagebox`. Modul `tkinter.messagebox` ini berguna untuk memberi suatu informasi mengenai status dari proses dalam menjalankan sebuah perangkat lunak. Berikut ini adalah beberapa fungsi untuk menampilkan status untuk *message box* yang digunakan dalam pembuatan perangkat lunak:

- `messagebox.showinfo()`: Menampilkan informasi yang berguna untuk pengguna. Pada Gambar 2.17 yang merupakan sebuah contoh tampilan *message box* berupa informasi yang berisi informasi bahwa “Absensi Berhasil”.



Gambar 2.17: Contoh Informasi dari Message Box

- `messagebox.showwarning()`: Menampilkan sebuah peringatan kepada pengguna. Pada Gambar 2.18 yang merupakan sebuah contoh tampilan *message box* berupa *warning* yang berisi peringatan bahwa “Absensi Gagal!”.



Gambar 2.18: Contoh Warning dari *Message Box*

### 2.6.3 Library OS

*Library OS* adalah modul sistem operasi pada python yang menyediakan cara untuk dapat berinteraksi langsung dengan sistem operasi. Sistem operasi yang dapat melakukan interaksi dengan python adalah windows, linux, dan mac. Berikut ini beberapa fungsi yang dimiliki pada *Library OS*:

- `os.environ`: Melakukan pemetaan terhadap objek yang mewakili *environment variable* milik pengguna.
- `os.getcwd()` : Mengembalikan direktori kerja yang digunakan.



## 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.

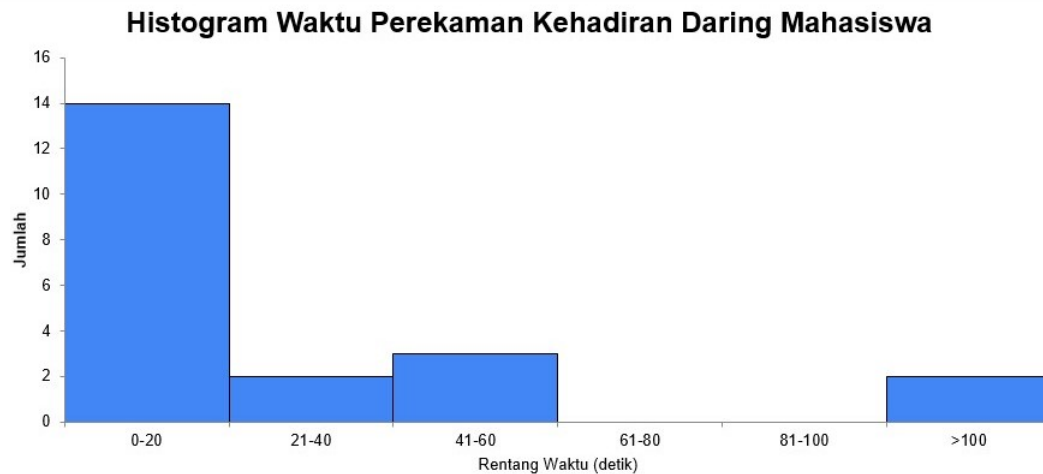
### 3.1 Analisis Hasil Survei Perekaman Kehadiran Daring dan Luring

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. Histogram ini dikelompokkan berdasarkan rentang waktu per 20 detik. Histogram tersebut menunjukkan bahwa mayoritas mahasiswa sebanyak 14 orang memiliki rentang waktu mulai dari 0 sampai 20 detik melakukan perekaman kehadiran secara daring, sebanyak 2 orang memiliki rentang waktu 21 sampai 40 detik, 3 orang memiliki rentang waktu 41 sampai 60 detik, dan 2 orang memiliki rentang waktu di atas 100 detik. Hasil survei perekaman kehadiran secara daring untuk setiap mahasiswa secara jelas dapat dilihat pada tabel 3.1. Jawaban dari 21 orang responden adalah mulai dari waktu paling cepat 10 detik hingga waktu paling lama 600 detik.

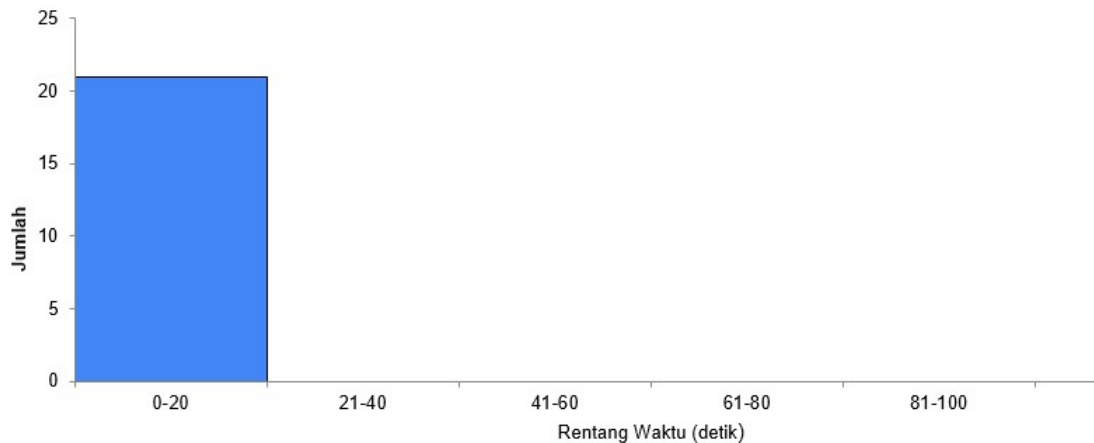
Tabel 3.1: Tabel Perekaman Daring Mahasiswa

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?

**Histogram Waktu Perekaman Kehadiran Luring Mahasiswa**



Gambar 3.2: Histogram Waktu Perekaman Kehadiran Luring Mahasiswa

Pada Gambar 3.2 merupakan visualisasi dari hasil survei mengenai lama waktu yang dibutuhkan dari 21 mahasiswa untuk melakukan perekaman kehadiran secara luring. Histogram ini dikelompokkan berdasarkan rentang waktu per 20 detik. Histogram tersebut menunjukkan bahwa seluruh mahasiswa sebanyak 21 orang memiliki rentang waktu mulai dari 0 sampai 20 detik melakukan perekaman kehadiran secara daring. Hasil survei perekaman kehadiran secara luring untuk setiap mahasiswa secara jelas dapat dilihat pada tabel 3.2. Jawaban dari 21 orang responden adalah mulai dari waktu paling cepat 5 detik hingga waktu paling lama 15 detik.

Tabel 3.2: Tabel Perekaman Luring Mahasiswa

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 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:

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



Gambar 3.3: Histogram Waktu Perekaman Kehadiran Daring Dosen

Pada Gambar 3.3 merupakan visualisasi dari hasil survei mengenai lama waktu yang dibutuhkan dari 6 dosen untuk melakukan perekaman kehadiran secara daring. Histogram ini dikelompokkan berdasarkan rentang waktu per 20 detik. Histogram menunjukkan bahwa sebanyak 4 dosen memiliki rentang waktu 0 sampai 20 detik, 1 dosen memiliki rentang waktu 21 sampai 40 detik, dan 1 dosen memiliki rentang waktu 101 sampai 120 detik. Hasil survei perekaman kehadiran daring untuk setiap dosen secara jelas dapat dilihat pada tabel 3.3.

Tabel 3.3: Tabel Perekaman Daring Dosen

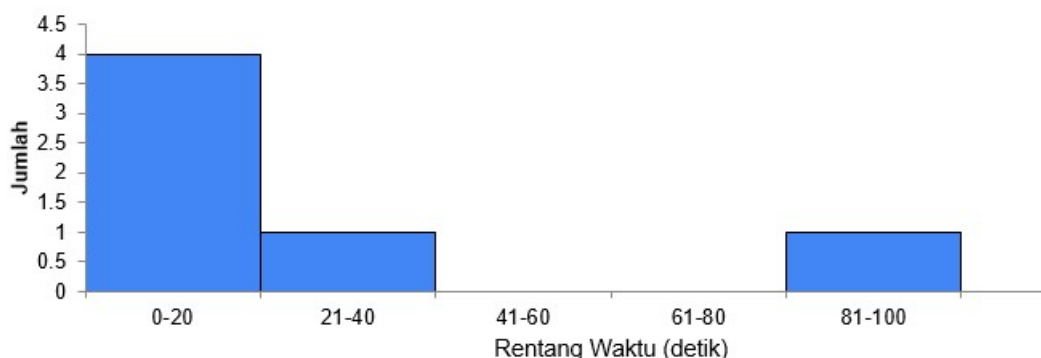
Jumlah Responden	Waktu Perekaman Kehadiran Daring
1 orang	1 detik
1 orang	10 detik
2 orang	15 detik
1 orang	30 detik
1 orang	120 detik

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?



### Histogram Waktu Perekaman Kehadiran Luring Dosen



Gambar 3.4: Histogram Waktu Perekaman Kehadiran Luring Dosen

Pada Gambar 3.4 merupakan visualisasi dari hasil survei mengenai lama waktu yang dibutuhkan dari 6 dosen untuk melakukan perekaman kehadiran secara luring. Histogram ini dikelompokkan berdasarkan rentang waktu per 20 detik. Histogram menunjukkan bahwa sebanyak 4 dosen memiliki rentang waktu 0 sampai 20 detik, 1 dosen memiliki rentang waktu 21 sampai 40 detik, dan 1 dosen memiliki rentang waktu 81 sampai 100 detik. Hasil survei perekaman kehadiran luring untuk setiap dosen secara jelas dapat dilihat pada tabel 3.4.

Tabel 3.4: Tabel Perekaman Luring Dosen

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 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:

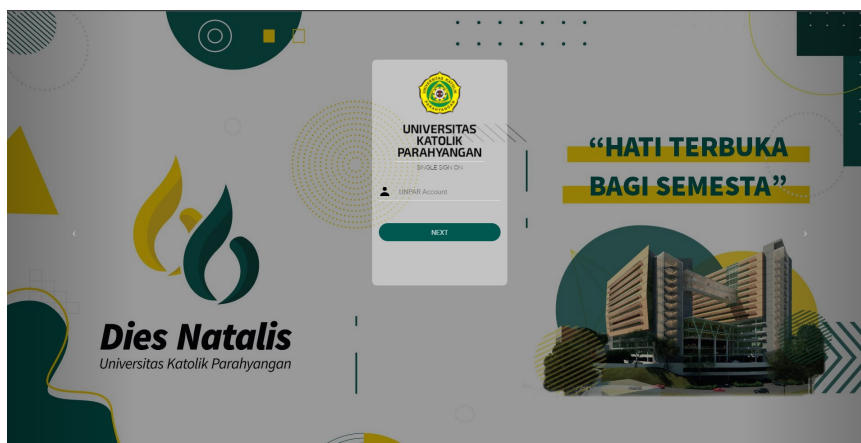
1. Melakukan akses Portal Akademik Mahasiswa yang dapat diakses melalui

<https://studentportal.unpar.ac.id/>.



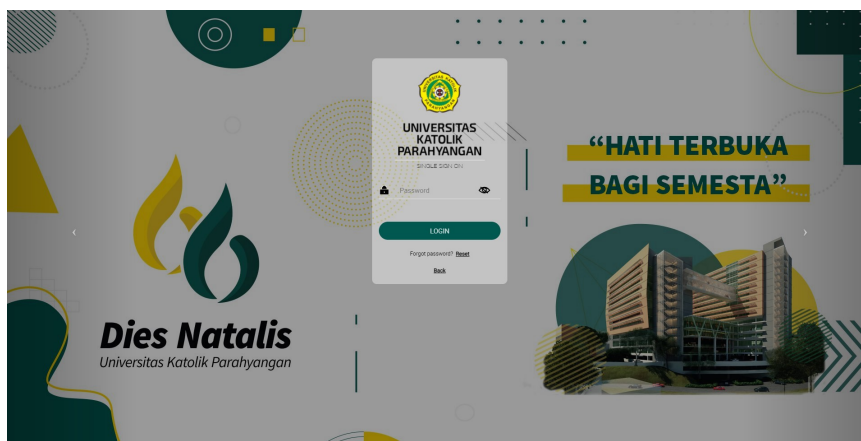
Gambar 3.5: Tampilan halaman awal Portal Akademik Mahasiswa

- 1 2. Menekan tombol “Login” yang sudah tersedia agar dapat masuk ke dalam Portal Akademik Mahasiswa, dapat dilihat pada Gambar 3.5.
- 3 3. Memasukan *email* mahasiswa.



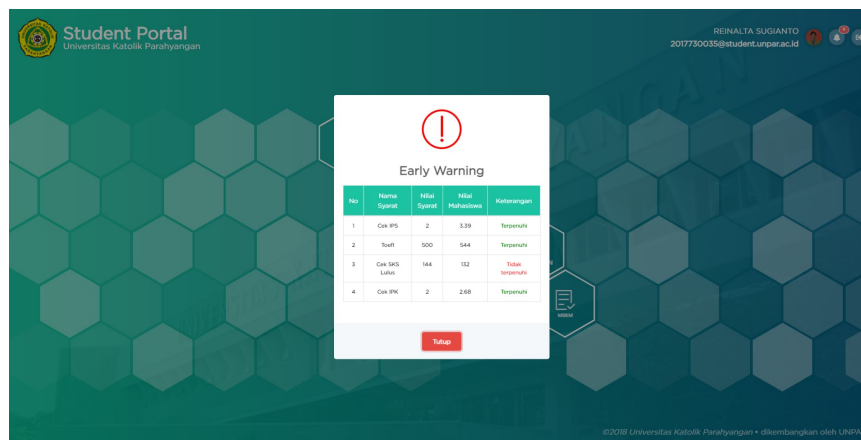
Gambar 3.6: Tampilan halaman Portal Akademik Mahasiswa untuk memasukan *email*

- 4 4. Menekan tombol “NEXT” setelah memasukan *email*, dapat dilihat pada Gambar 3.6.
- 5 5. Memasukan *password* milik mahasiswa.

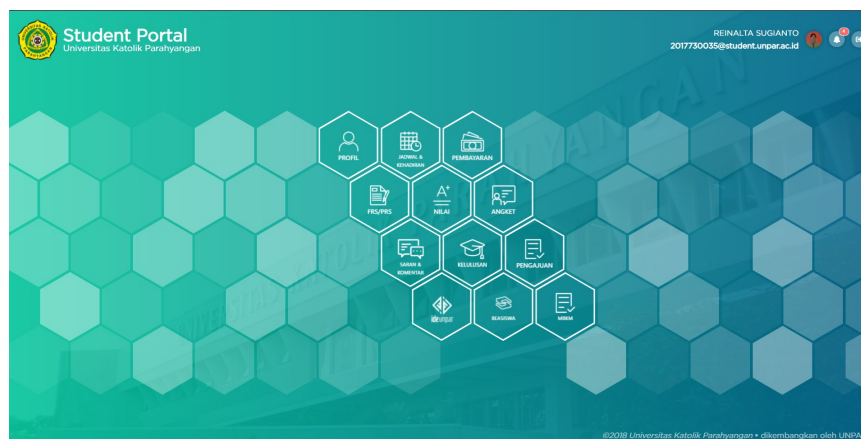


Gambar 3.7: Tampilan halaman Portal Akademik Mahasiswa untuk memasukan *password*

6. Menekan tombol “LOGIN” setelah memasukkan *password*, dapat dilihat pada Gambar 3.7.
7. Menekan tombol “Tutup” jika muncul peringatan atau langsung menekan tombol berbentuk heksagon “JADWAL & KEHADIRAN” jika tidak muncul peringatan.

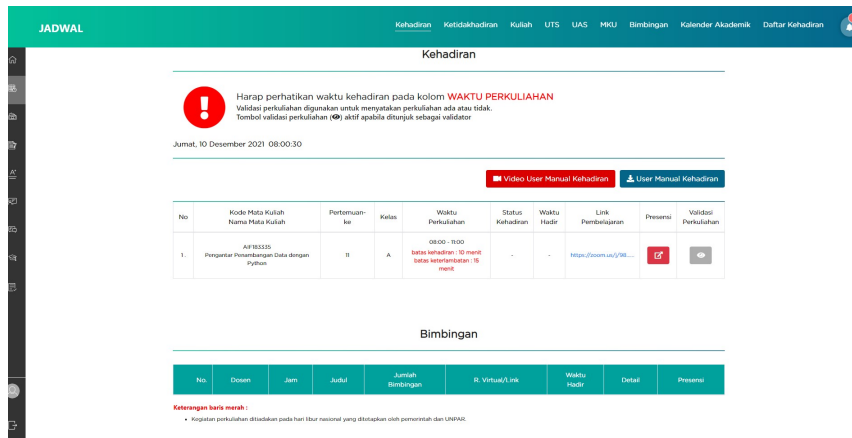


Gambar 3.8: Tampilan peringatan pada halaman Portal Akademik Mahasiswa



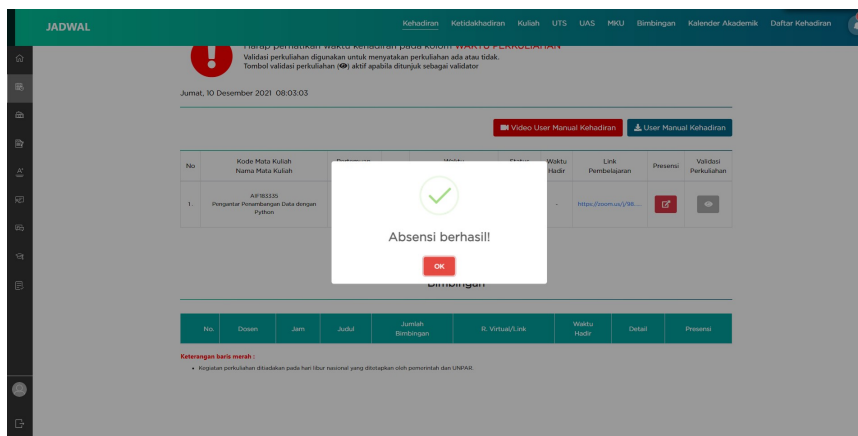
Gambar 3.9: Tampilan halaman Portal Akademik Mahasiswa setelah Berhasil *Login*

- Pada Gambar 3.8 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 3.9. Jika tidak terjadi peringatan seperti pada Gambar 3.8, maka dapat langsung menekan tombol berbentuk heksagon “JADWAL & KEHADIRAN” seperti pada Gambar 3.9.
8. Menekan tombol berwarna merah pada kolom bagian persensi dari tabel jadwal kehadiran mata kuliah.



Gambar 3.10: Tampilan halaman Portal Akademik Mahasiswa untuk Melakukan Absen

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



Gambar 3.11: 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 2.4. 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

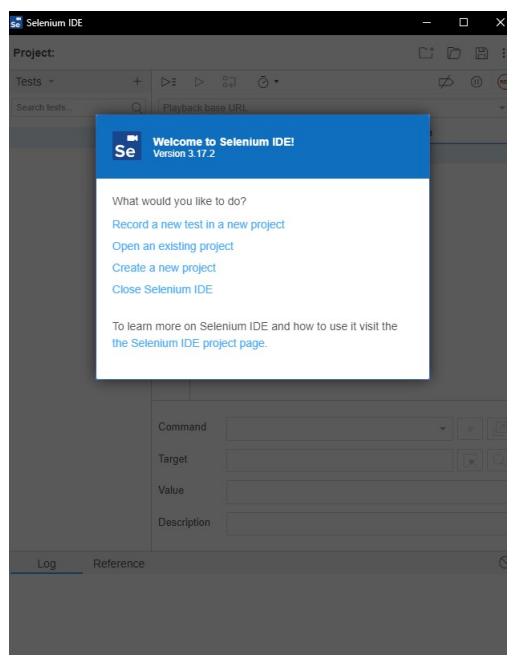
Menulis *query selector* yang mudah dibaca dan dimengerti sehingga lebih mudah untuk dipahami, contohnya “`#login-button`” yang artinya memilih elemen tombol untuk *login*. Tidak disarankan menulis *query selector* yang panjang atau sulit dibaca, contohnya mengambil elemen dengan cara XPath seperti yang sudah ditulis pada Bab 2.4 dengan kode program 2.11.

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

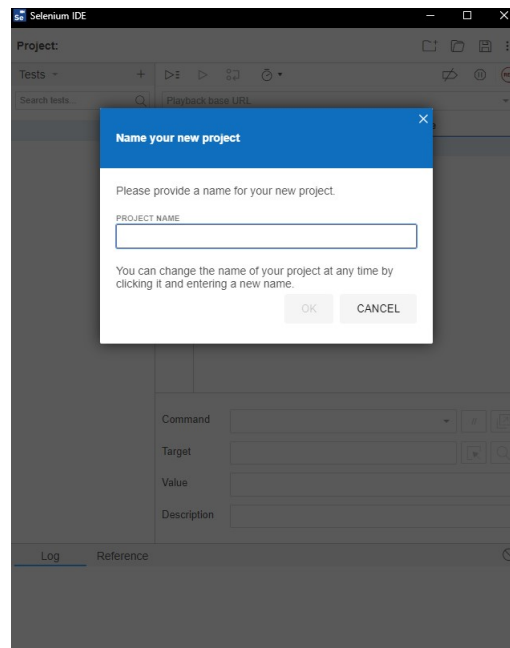
Selenium IDE merupakan program *open source* untuk otomatisasi di web. Selenium IDE dapat di *install* di browser, contohnya di Google Chrome yang setelah di *install* akan menjadi *extensions*. *Extensions* di Google Chrome adalah sebuah aplikasi kecil yang dapat dijalankan pada Google Chrome itu sendiri.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



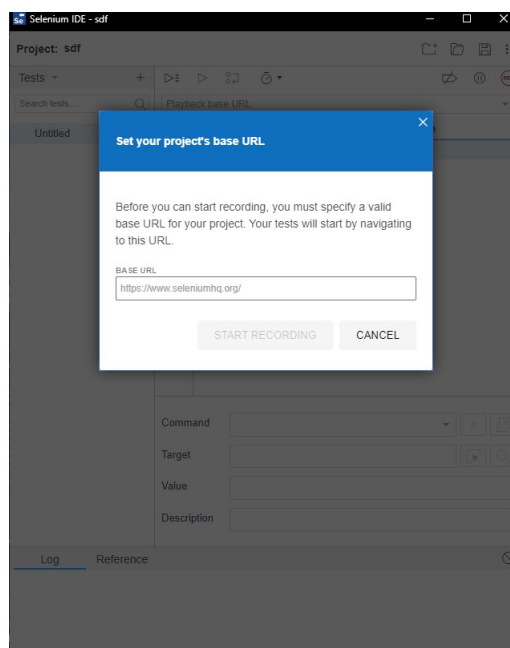
Gambar 3.12: Tampilan Menu Awal Selenium IDE

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



Gambar 3.13: Tampilan Memasukan Nama Proyek

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



Gambar 3.14: Tampilan Memasukan Situs Web

- 3 Setelah menekan tombol “*START RECORDING*” seperti pada Gambar 3.14, 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.







## BAB 4

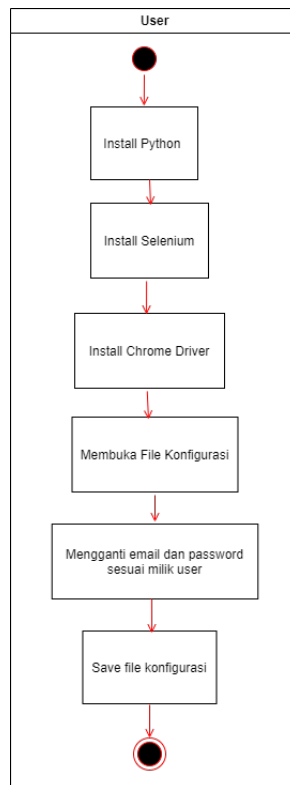
### PERANCANGAN

Pada bab ini akan dijelaskan perancangan perangkat lunak yang dibuat pada penelitian ini. Perancangan terdiri dari diagram aktivitas dan masukan perangkat lunak.

#### 4.1 Diagram Aktivitas

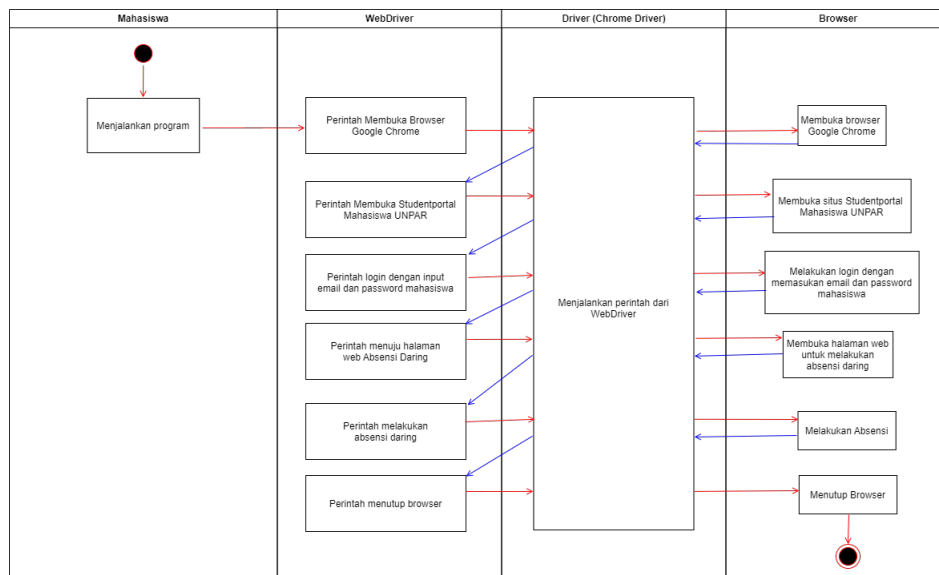
Perangkat lunak perekaman absen daring otomatis adalah perangkat lunak yang digunakan untuk melakukan absensi secara otomatis bagi mahasiswa UNPAR. Perangkat lunak ini menggunakan Selenium WebDriver sebagai *tools* yang berguna untuk melakukan otomatisasi pada browser web. Perangkat lunak ini juga membutuhkan masukan dari sebuah file konfigurasi untuk menjalankannya. Diagram Aktivitas untuk *setup* menjalankan program dan menyiapkan file konfigurasi dapat dilihat Gambar 4.1. Berikut ini adalah penjelasan langkah-langkah pada diagram aktivitas:

1. Pengguna melakukan *install* python, karena program perekaman absen daring otomatis menggunakan bahasa pemrograman python.
2. Pengguna melakukan *install* selenium, karena untuk melakukan perekaman absen daring secara otomatis menggunakan selenium.
3. Pengguna melakukan *install* chrome driver, karena menggunakan browser Google Chrome untuk melakukan perekaman absen daring otomatis.
4. Pengguna membuka file konfigurasi dan mengubah email serta password sesuai milik pengguna agar dapat digunakan sebagai masukan pada perangkat lunak untuk melakukan perekaman absen daring otomatis.
5. Pengguna menyimpan hasil perubahan yang telah dilakukan pada file konfigurasi.



Gambar 4.1: Diagram Aktivitas untuk *Setup* Menjalankan Program

- 1 Diagram Aktivitas untuk perangkat lunak perekaman kehadiran daring otomatis dapat dilihat pada  
2 Gambar 4.2. Berikut ini adalah penjelasan langkah-langkah pada diagram aktivitas:
- 3 1. Pengguna menjalankan langsung programnya.
  - 4 2. WebDriver memberikan perintah untuk membuka browser Google Chrome kepada Driver  
5 (Chrome Driver).
  - 6 3. Chrome driver disini bertugas menghubungkan setiap perintah dari WebDriver untuk dija-  
7 lankan pada browser Google Chrome.
  - 8 4. Setelah browser Google Chrome terbuka, Browser Google Chrome memberi tahu melalui  
9 Chrome driver bahwa perintah sudah dijalankan dan siap menjalankan perintah selanjutnya.
  - 10 5. WebDriver memberi perintah membuka web Portal Mahasiswa UNPAR dan Chrome driver  
11 memberi tahu kepada browser Google Chrome membuka web Portal Mahasiswa UNPAR.
  - 12 6. Browser selanjutnya melakukan login dengan memasukan *email* dan *password* mahasiswa,  
13 sesuai dengan perintah dari WebDriver.
  - 14 7. Perintah menuju halaman web untuk absensi daring diberikan oleh WebDriver untuk dijalankan  
15 secara otomatis oleh Google Chrome.
  - 16 8. WebDriver memberi perintah untuk melakukan absensi daring kepada Google Chrome, sehingga  
17 dapat terjadi absensi daring secara otomatis.
  - 18 9. Terakhir WebDriver memberi perintah untuk menutup browser Google Chrome setelah berhasil  
19 melakukan absensi dan browser Google Chrome ditutup secara otomatis.



Gambar 4.2: Diagram Aktivitas Perangkat Lunak Absen Daring Otomatis

## 4.2 Masukan Perangkat Lunak

Perangkat lunak perekaman kehadiran daring otomatis membutuhkan 1 file sebagai masukan, yaitu *file* .ini (*file* konfigurasi). Pada *file* .ini, nomor baris sebagai *keys* dan *string* berupa kata yang merupakan fungsi dari Selenium WebDriver dan elemen yang diambil untuk melakukan perekaman kehadiran daring otomatis sebagai *values*. Contoh *file* .ini dapat dilihat pada Listing 4.1.

Kode 4.1: Contoh *file* .ini untuk Masukan Perangkat Lunak Perekaman Kehadiran Daring Otomatis

```

6  [database_config]
7  1 = open https://studentportal.unpar.ac.id
8  2 = click #login-button
9  3 = sendkeys #username 2017730035@student.unpar.ac.id
10 4 = quit
11 5

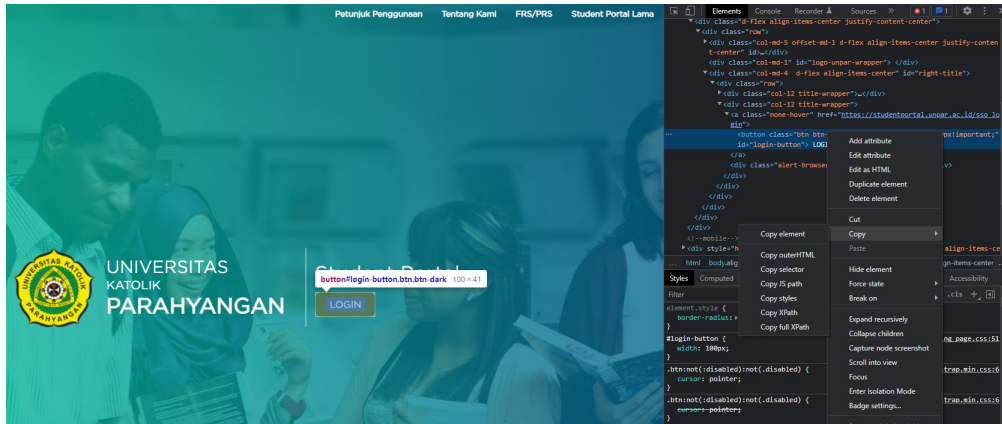
```

Berikut ini penjelasan dari isi dari contoh *file* .ini:

- Baris pertama berisi nama *section* untuk isi *file* .ini.
- *keys* pada *file* .ini ini pasti berupa angka yang terurut agar perangkat lunak dapat menjalankannya secara terurut.
- Terdapat 4 fungsi kata dari Selenium WebDriver, yaitu *open*, *click*, *sendkeys*, dan *quit*.
- *Keys* 1 pasti diisi oleh fungsi *open*, lalu diisi situs web (<https://studentportal.unpar.ac.id>), karena langkah pertama setelah berhasil membuka browser adalah menuju pada situs web yang akan diotomatisasi.
- *Keys* 2 memiliki fungsi *click* untuk menekan tombol secara otomatis dan diisi elemen “#login-button” yang diambil berdasarkan CSS Selector.
- *Keys* 3 memiliki fungsi *sendkeys* untuk memasukkan suatu nilai ke dalam elemen yang dipilih, yaitu elemen “#username” dan isinya adalah “2017730035@student.unpar.ac.id”.
- *Keys* 4 memiliki fungsi *quit* untuk menutup browsernya.

Elemen yang dipakai dalam *file* .ini ini diambil dengan cara melakukan *inspect element* pada web yang ingin dilakukan otomatisasi. Pada Gambar 4.3 adalah cara yang dilakukan untuk mendapat elemen yang ingin digunakan untuk melakukan otomatisasi. Untuk mendapatkan elemen tersebut,

- 1 perlu melakukan klik kanan pada bagian elemen yang ingin diambil, lalu pilih “inspect”. Setelah
- 2 melakukan “inspect” maka akan muncul dokumen HTML yang dapat dilihat pada bagian kanan
- 3 Gambar 4.3, sehingga dapat melakukan pengambilan elemen yang diperlukan.



Gambar 4.3: Tampilan Melakukan *Inspect Element*

## BAB 5

### IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi Implementasi Perangkat Lunak dan Pengujian Perangkat Lunak. Bagian implementasi terdiri dari penjelasan lingkungan pengembangan perangkat lunak dan hasil implementasi. Bagian pengujian terdiri dari hasil pengujian fungsional dan eksperimental terhadap perangkat lunak yang telah dibangun.

#### 5.1 Implementasi

##### 5.1.1 Lingkungan Implementasi

Implementasi perangkat lunak ini dilakukan pada komputer penulis dengan spesifikasi berikut:

1. *Processor*: Intel Core i5 9400F
2. *Random Access Memory* (RAM): 16 GB DDR4
3. Sistem Operasi: Windows 10
4. Versi Python : Python 3.8.5

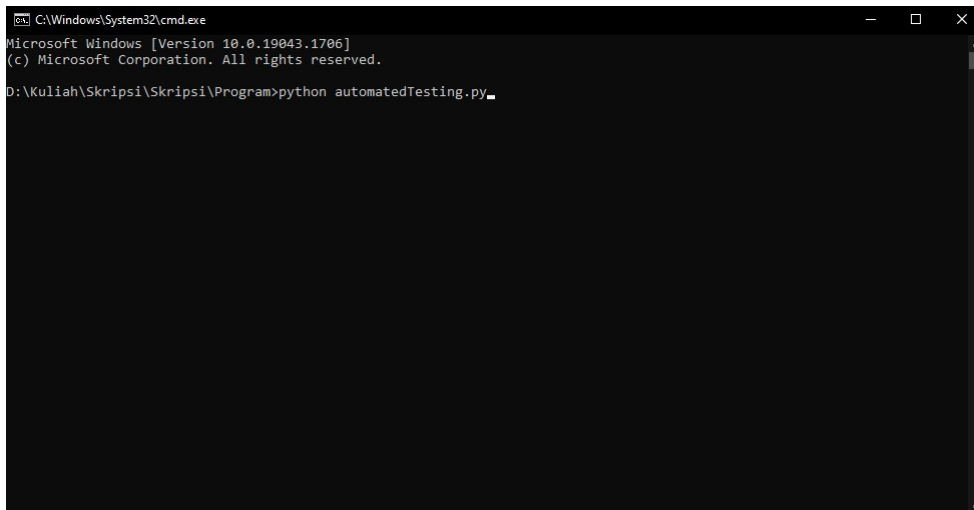
##### 5.1.2 Hasil Implementasi

Hasil implementasi berupa sebuah perangkat lunak perekaman kehadiran daring otomatis dengan bahasa pemrograman python. Sebelum menjalankan perangkat lunak untuk perekaman kehadiran daring otomatis, terdapat *file* .ini yang merupakan sebuah masukan untuk perangkat lunak. *File* .ini dibahas pada Subbab 4.2. Contoh *file* .ini dapat dilihat pada 5.1.

Kode 5.1: Contoh *file* .ini untuk Masukan Perangkat Lunak Perekaman Kehadiran Daring Otomatis

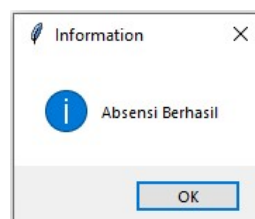
```
[database_config]
1 = open https://studentportal.unpar.ac.id
2 = click #login-button
3 = sendkeys #username 2017730035@student.unpar.ac.id
4 = click #next_login
5 = sendkeys #password 12345
6 = click #appPass>div.login__form>button
7 = or a[href='https://studentportal.unpar.ac.id/jadwal'] .swal-button.swal-button--confirm.swal-button--danger
8 = click a[onclick="absenPerkuliahan(this)"]
9 = click .swal-button.swal-button--confirm.swal-button--danger
10 = quit
```

Perekaman kehadiran daring otomatis dapat dilakukan dengan menjalankan perangkat lunak. Pengguna perlu membuka *Command Prompt* pada komputer maupun laptop dengan *directory* file automatedTesting.py berada dan menuliskan perintah “python automatedTesting.py” atau “py automatedTesting.py” pada *Command Prompt*, seperti pada tampilan Gambar 5.1

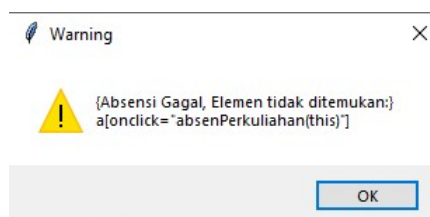


Gambar 5.1: Tampilan *Command Prompt* dengan *Directory File*

1     Setelah pengguna menekan “Enter” pada *Command Prompt* maka perangkat lunak akan  
2     melakukan perekaman kehadiran daring secara otomatis, bagi mahasiswa maka perangkat lunak  
3     akan melakukan perekaman kehadiran daring pada Portal Akademik Mahasiswa secara otomatis,  
4     dimana perangkat lunak akan menjalankan secara otomatis tahap-tahap perekaman kehadiran daring  
5     secara manual yang biasa dilakukan mahasiswa seperti yang dibahas pada Subbab 3.2, sedangkan  
6     bagi dosen maka perangkat lunak akan melakukan perekaman kehadiran daring pada AKUHADIR  
7     seperti yang dibahas pada Subbab 2.2. Setelah berhasil melakukan perekaman kehadiran daring  
8     maka akan muncul notifikasi bahwa perekaman berhasil dilakukan, seperti pada tampilan Gambar  
9     5.2, selain itu akan muncul notifikasi berupa peringatan bahwa absensi gagal, seperti pada tampilan  
10    Gambar 5.3. Absensi gagal terjadi karena tidak ada jadwal kuliah lagi bagi mahasiswa, atau sudah  
11    melakukan absensi sehingga tidak ada yang bisa lagi untuk melakukan perekaman kehadiran.



Gambar 5.2: Tampilan Notifikasi Berhasil Absen



Gambar 5.3: Tampilan Notifikasi Gagal Absen

## 5.2 Pengujian

### 5.2.1 Pengujian Fungsional

Pengujian fungsional dilakukan untuk mengetahui kesesuaian reaksi perangkat lunak dengan reaksi yang diharapkan berdasarkan aksi pengguna terhadap perangkat lunak. Tabel 5.1 merupakan tabel hasil pengujian perangkat lunak pada komputer penulis dengan spesifikasi berikut:

1. *Processor*: Intel Core i5 9400F
2. *Random Access Memory* (RAM): 16 GB DDR4
3. Sistem Operasi: Windows 10
4. Versi Python : Python 3.8.5

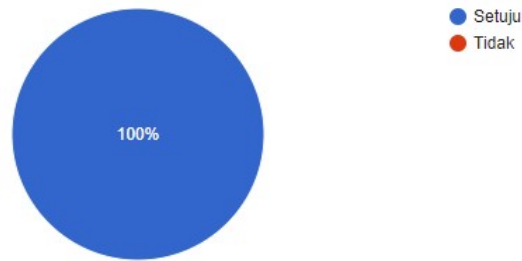
Tabel 5.1: Tabel Pengujian Fungsional

No.	Aksi Pengguna	Reaksi yang diharapkan	Reaksi Perangkat Lunak
1.	Mahasiswa menjalankan perangkat lunak	Browser Google Chrome terbuka	Sesuai
		Browser menuju situs Portal Akademik Mahasiswa	Sesuai
		Browser menuju halaman web untuk perekaman kehadiran daring	Sesuai
		Melakukan perekaman kehadiran daring secara otomatis	Sesuai
2.	Dosen menjalankan perangkat lunak	Browser Google Chrome terbuka	Sesuai
		Browser menuju situs AKUHA-DIR	Sesuai
		Browser menuju halaman web untuk perekaman kehadiran	Sesuai
		Melakukan perekaman kehadiran daring secara otomatis	Sesuai

### 5.2.2 Pengujian Eksperimental

Pengujian eksperimental dilakukan terhadap beberapa mahasiswa Universitas Katolik Parahyangan jurusan Teknik Informatika yang sudah memiliki Google Chrome dan Python3. Metode pengujian dilakukan dengan cara menyebarkan perangkat lunak yang dapat diunduh melalui Google Drive. Setelah menjalankan perangkat lunak tersebut, mahasiswa diminta untuk mengisi Google Form untuk mengetahui kelancaran perangkat lunak ketika dijalankan dan mengetahui lama waktu yang dibutuhkan hingga program berhasil melakukan perekaman kehadiran. Berikut ini hasil yang didapatkan dari pengisian survei:

- Dari 7 responden yang telah mengisi survei, memberi respons bahwa perangkat lunak berjalan dengan baik dan dapat melakukan perekaman kehadiran daring secara otomatis. Hasil diagram lingkaran pada Gambar 5.4 menunjukkan bahwa semua responden menyatakan setuju program tidak mengalami *error* atau *crash*.



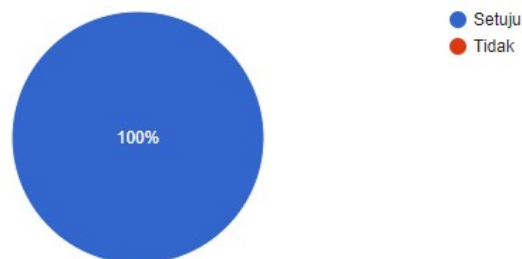
Gambar 5.4: Diagram Lingkaran Kesetujuan Perangkat Lunak Tidak *Error* atau *Crash*

- Tabel 5.2 menunjukkan waktu yang didapatkan dari 7 responden dalam menjalankan perangkat lunak untuk melakukan perekaman kehadiran daring otomatis. Hasil tabel tersebut menunjukkan bahwa dalam melakukan perekaman kehadiran daring otomatis berada di rentang waktu 11-22 detik dan hasil rata-rata waktu adalah 16,71 detik.

Tabel 5.2: Tabel Perekaman Kehadiran Otomatis

Jumlah Responden	Waktu Perekaman Kehadiran Otomatis
1 orang	11 detik
1 orang	14 detik
1 orang	15 detik
2 orang	18 detik
1 orang	19 detik
1 orang	22 detik

- Diagram lingkaran pada Gambar 5.5 menunjukkan bahwa sebanyak 7 responden menyatakan setuju bahwa perangkat lunak untuk melakukan perekaman kehadiran daring secara otomatis dapat membuat waktu interaksi dengan situs web atau browser menjadi lebih singkat.



Gambar 5.5: Diagram Lingkaran Kesetujuan Perangkat Lunak Menghemat Waktu Interaksi dengan Browser



## BAB 6

### KESIMPULAN DAN SARAN

#### 6.1 Kesimpulan

Dari hasil pembangunan perangkat lunak Perekaman Kehadiran Daring Otomatis, didapatkan kesimpulan-kesimpulan sebagai berikut:

1. Telah berhasil membangun perangkat lunak perekaman kehadiran daring secara otomatis menggunakan Selenium WebDriver.
2. Telah berhasil membuat perangkat lunak yang mampu secara otomatis melakukan tahap-tahap dalam melakukan perekaman kehadiran secara daring dengan sekali menjalankan perangkat lunak. Meskipun waktu yang dibutuhkan dalam melakukan perekaman kehadiran daring secara otomatis masih lebih lama dibandingkan perekaman secara luring, tetapi berdasarkan survei bahwa waktu interaksi dengan situs web atau browser untuk melakukan perekaman kehadiran daring menjadi lebih singkat.

#### 6.2 Saran

Dari hasil penelitian, pengujian, dan kesimpulan yang didapat, berikut ini adalah beberapa saran untuk pengembang lebih lanjut:

1. Mempercepat waktu dalam perekaman kehadiran daring otomatis sehingga dapat menyamai waktu perekaman kehadiran secara luring.



## DAFTAR REFERENSI

- [1] 9f08b37 (2021) *Selenium*. Software Freedom Conservancy.
- [2] 2018, T. P. P. A. M. P. (2018) Portal akademik mahasiswa. [https://studentportal.unpar.ac.id/assets/BUKU\\_PANDUAN\\_PENGGUNAAN\\_FRS\\_GABUNGAN.pdf](https://studentportal.unpar.ac.id/assets/BUKU_PANDUAN_PENGGUNAAN_FRS_GABUNGAN.pdf). Online; diakses 15-November-2021.
- [3] Situmorang, M. (2020) Pemberlakuan sementara kebijakan bekerja dari rumah (wfh). Surat Edaran No. III/R/2020-07/1153.
- [4] Keller, M. dan Nussbaumer, M. (2010) Css code quality: A metric for abstractness; or why humans beat machines in css coding. *2010 Seventh International Conference on the Quality of Information and Communications Technology*, pp. 116–121.
- [5] Version 3.1 (2017) *XML path language (XPath) 3.1*. World Wide Web Consortium.
- [6] Version 3.10.4 (2022) *Configuration file parser*. Python Software Foundation.
- [7] Grayson, J. E. (2000) *Python and Tkinter programming*. Manning Publications Co. Greenwich.



## LAMPIRAN A

### *FILE* MASUKAN UNTUK PERANGKAT LUNAK

#### A.1 *File* Konfigurasi

*File* .ini yang digunakan sebagai *file* konfigurasi yang berguna sebagai masukan perangkat lunak perekaman kehadiran daring secara otomatis.

Kode A.1: database.ini

```
1 [database_config]
2 1 = open https://studentportal.unpar.ac.id
3 2 = click #login-button
4 3 = sendkeys #username 2017730035@student.unpar.ac.id
5 4 = click #next_login
6 5 = sendkeys #password 12345
7 6 = click #appPass>div.login__form>button
8 7 = or a[href='https://studentportal.unpar.ac.id/jadwal'] .swal-button.swal-button--confirm.swal-button--danger
9 8 = click a[onclick="absenPerkuliahah(this)"]
10 9 = click .swal-button.swal-button--confirm.swal-button--danger9
11 10 = quit
```



## LAMPIRAN B

### KODE PROGRAM PERANGKAT LUNAK PEREKAMAN KEHADIRAN DARING OTOMATIS

Kode B.1: automatedTesting.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun Mar 13 11:35:40 2022
4
5 @author: user
6 """
7 from configparser import ConfigParser
8 from selenium import webdriver
9 from selenium.webdriver.common.by import By
10 from selenium.webdriver.support import expected_conditions as EC
11 from selenium.webdriver.support.ui import WebDriverWait
12 from selenium.common.exceptions import TimeoutException
13 from tkinter import *
14 from tkinter import messagebox
15 import os
16
17 os.environ["PATH"] = os.getcwd()
18 print(os.environ["PATH"])
19
20
21 driver = webdriver.Chrome()
22
23 parser = ConfigParser()
24 parser.read('database.ini')
25
26 i = 1
27 while (i <= 1):
28     x = parser.get('database_config', str(i)).split()
29     if x[0] == "open":
30         driver.get(x[1])
31         i += 1
32     elif x[0] == "click":
33         try:
34             elemen = WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.CSS_SELECTOR, x[1])))
35             if elemen.is_displayed() and elemen.is_enabled():
36                 elemen.click()
37         except TimeoutException:
38             driver.quit()
39             root = Tk()
40             root.withdraw()
41             a = "Absensi_Gagal, Elemen_tidak_ditemukan:", x[1]
42             messagebox.showwarning("Warning", a)
43             break
44         i+=1
45     elif x[0] == "sendkeys":
46         inpt = WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.CSS_SELECTOR, x[1])))
47         inpt.send_keys(x[2])
48         i += 1
49     elif x[0] == "or":
50         try:
51             elemen1 = driver.find_element(By.CSS_SELECTOR, x[1]) #jadwal
52             elemen2 = WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.CSS_SELECTOR, x[2]))) #notif
53             if elemen2.is_displayed() and elemen2.is_enabled():
54                 elemen2.click()
55                 elemen1.click()
56         except TimeoutException:
57             elemen1.click()
58         i += 1
59     elif x[0] == "quit":
60         driver.wait(3)
61         driver.quit()
62         root = Tk()
63         root.withdraw()
64         messagebox.showinfo("Information", "Absen_Berhasil")
65         break
```