SKRIPSI

PEREKAMAN KEHADIRAN DARING OTOMATIS

Reinalta Sugianto

NPM: 2017730035

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

UNDERGRADUATE THESIS

AUTOMATIC ONLINE ATTENDANCE RECORDING

Reinalta Sugianto

NPM: 2017730035

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2022

LEMBAR PENGESAHAN

PEREKAMAN KEHADIRAN DARING OTOMATIS

Reinalta Sugianto

NPM: 2017730035

Bandung, «tanggal» «bulan» 2022

Menyetujui,

Pembimbing

Pascal Alfadian Nugroho, S.Kom., M.Comp.

Ketua Tim Penguji	Anggota Tim Penguji
Luciana Abednego, S.Kom., M.T.	Dr. Ir. Veronica Sri Moertini, MT

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

PEREKAMAN KEHADIRAN DARING OTOMATIS

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal «tanggal» «bulan» 2022

Reinalta Sugianto
NPM: 2017730035

ABSTRAK

Portal Akademik Mahasiswa merupakan sebuah situs web yang digunakan mahasiswa untuk menunjang kegiatan akademik. Portal Akademik Mahasiswa digunakan untuk melakukan pengisian formulir rencana studi (FRS) serta melihat infomasi nilai, pembayaran, dan jadwal kuliah. Akibat dari pandemi Covid-19 menyebabkan kegiatan perkuliahan dilakukan secara daring, sehingga Portal Akademik Mahasiswa yang terbaru sudah dapat melakukan perekaman kehadiran daring untuk setiap mata kuliah yang diambil. Mahasiswa biasanya melakukan perekaman kehadiran daring secara manual dan membutuhkan waktu yang cukup lama untuk berinteraksi dengan Portal Akademik Mahasiswa. Oleh karena itu, dibuatlah program yang dapat melakukan perekaman kehadiran daring secara otomatis menggunakan selenium agar dapat mengurangi waktu interaksi dengan situs web.

Program perekaman kehadiran daring otomatis akan dibangun dengan bahasa pemrograman Python menggunakan *framework* Selenium untuk dapat mengotomatisasi browser Google Chrome. Selenium merupakan *open-source framework* untuk pengujian otomatisasi untuk browser web. Selenium dapat mengontrol browser diperantarai oleh WebDriver yang merupakan bagian inti selenium untuk melakukan otomatisasi pada browser. Agar selenium dapat berkomunikasi dengan browser maka dibutuhkan sebuah komponen, yaitu Driver dari browser itu sendiri. Tujuan dari program ini agar dapat melakukan perekaman kehadiran daring menjadi lebih mudah. Program yang akan melakukan perekaman kehadiran daring secara otomatis, sehingga pengguna dapat mengurangi waktu interaksi dengan *browser*.

Pengujian program perekaman kehadiran daring otomatis dilakukan terhadap beberapa mahasiswa pada situs Portal Akademik Mahasiswa maupun dosen pada situs AKUHADIR. Berdasarkan hasil pengujian eksperimental terhadap mahasiswa dan dosen menunjukan bahwa program berjalan dengan baik dan dapat melakukan perekaman kehadiran daring secara otomatis. Hasil pengujian fungsional program perekaman kehadiran daring otomatis dari penulis selaku mahasiswa dan dosen pembimbing berjalan baik. Hasil pengujian program perekaman kehadiran daring otomatis membuktikan bahwa telah berhasil melakukan perekaman kehadiran daring secara otomatis dan mengurangi waktu interaksi pengguna dengan *browser*.

Kata-kata kunci: Portal Akademik Mahasiswa, Selenium, WebDriver, Driver, AKUHADIR

ABSTRACT

Portal Akademik Mahasiswa is a website used by students to support academic activities. Portal Akademik Mahasiswa is used to fill out the study plan form (FRS) and view information on grades, payments, and class schedules. As a result of the Covid-19 pandemic, lectures are conducted online, so the latest Portal Akademik Mahasiswa can record online attendance for each course taken. Students usually record online attendance manually and it takes a long time to interact with the Portal Akademik Mahasiswa. Therefore, a program was created that can automatically record online attendance using selenium in order to reduce interaction time with websites.

Automatic online attendance recording program will be built in Python programming language using Selenium *framework* to automate Google Chrome browser. Selenium is an *open-source framework* for testing automation for web browsers. Selenium can control browsers mediated by WebDriver which is a core part of selenium to perform browser automation. In order for selenium to communicate with the browser, a component is needed, namely the driver from the browser itself. The purpose of this program is to make online attendance recording easier. A program that will automatically record online attendance, so users can reduce interaction time with *browser*.

Testing of the online attendance recording program was automatically carried out on several students on the Portal Akademik Mahasiswa site and lecturers on the AKUHADIR site. Based on the results of experimental tests on students and lecturers, it shows that the program runs well and can record online attendance automatically. The results of the functional testing of the automatic online attendance recording program from the author as a student and supervisor went well. The test results of the automatic online attendance recording program prove that it has succeeded in automatically recording online attendance and reducing user interaction time with *browser*.

Keywords: Portal Akademik Mahasiswa, Selenium, WebDriver, Driver, AKUHADIR

Tuhan Yesus Allah yang Maha Esa, Keluarga, Teknik Informatika UNPAR, teman-teman, dan diri sendiri

KATA PENGANTAR

Puji syukur kepada Tuhan Yesus Kristus Allah yang Maha Esa atas seluruh berkat yang diberikan kepada penulis sehingga dapat menyelesaikan skripsi dengan judul **Perekaman Kehadiran Daring Otomatis** dengan baik dan tepat waktu. Penulis juga berterima kasih kepada pihak-pihak yang telah memberikan dukungan dan bantuan kepada penulis dalam menyelesaikan skripsi ini. Penulis ingin mengucapkan terima kasih kepada:

1. Orang tua dan Keluarga yang selalu memberikan dukungan dan doa.
2. Bapak Pascal Alfadian Nugroho selaku dosen pembimbing yang selalu membimbing hingga penulis berhasil menyelesaikan skripsi ini dengan baik.
3. Ibu Luciana Abednego, S.Kom., M.T. dan Ibu Dr. Veronica Sri Moertini, MT selaku dosen penguji yang telah memberikan kritik, saran, serta masukan yang membangun sehingga penelitian ini menjadi lebih baik.
4. Seluruh mahasiswa Teknik Informatika UNPAR angkatan 2017-2020 yang telah membantu dalam skripsi ini.
5. Seluruh dosen Teknik Informatika UNPAR yang telah memberikan ilmu dari awal kuliah sampai penulis berhasil menyelesaikan skripsi ini.
6. Teman-teman 'GRBLG' yang sejak awal telah setia bersama berjuang mengarungi masa-masa senang, sulit, dan pahit di Teknik Informatika UNPAR khususnya Reynard Rafferty Susilo, Dio Antares, David Christopher Sentosa, Sterenlie, Rio Aurelio Sumantri, Fritz Humphrey Silalahi, Leonard Wang, dan Juan Nandrissa Redemptino.
7. Teman-teman Teknik Informatika UNPAR angkatan 2017 yang telah berbagi ilmu, memberikan dukungan, dan membantu penulis menyelesaikan skripsi ini.
8. Pihak-pihak lain yang telah membantu penulisan skripsi ini, yang terus memberikan doa dan semangat kepada penulis.

Penulis menyadari bahwa skripsi ini masih jauh dari kata sempurna karena keterbatasan pengetahuan dan kemampuan penulis. Oleh karena itu, penulis memohon maaf jika terdapat kekurangan pada skripsi ini. Penulis juga mengharapkan kritik dan saran yang membangun untuk menyempurnakan skripsi ini. Semoga skripsi ini dapat bermanfaat bagi pembaca atau pihak yang meneruskan penelitian ini.

Bandung, «bulan» 2022

Penulis

DAFTAR ISI

Kata Pengantar **xv**

Daftar Isi **xvii**

Daftar Gambar **xix**

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 . . . 3
1.6 Sistematika Pembahasan . . . 3

2 Landasan Teori **5**
2.1 Selenium . . . 5
2.1.1 WebDriver . . . 5
2.1.2 Interaksi *Browser* . . . 6
2.1.3 Menemukan elemen . . . 6
2.1.4 Interaksi Elemen . . . 8
2.1.5 Waits . . . 9
2.2 *Cascading Style Sheets* (CSS) *Selector* . . . 10
2.3 *Configuration File Parser* (ConfigParser) . . . 12
2.4 *Library* OS . . . 12

3 Analisis **13**
3.1 Analisis Masalah . . . 13
3.2 Analisis Sistem Kini . . . 13
3.2.1 Portal Akademik Mahasiswa 2018 . . . 13
3.2.2 Fitur Tambahan Perekaman Kehadiran Portal Akademik Mahasiswa . . . 19
3.2.3 AKUHADIR 2.1 . . . 24
3.3 Analisis Sistem Sejenis . . . 28
3.3.1 Selenium IDE . . . 28
3.4 Analisis Sistem Usulan . . . 30
3.4.1 Analisis Hasil Survei Perekaman Kehadiran Daring dan Luring . . . 30
3.4.2 Perekaman Kehadiran Daring ke dalam Selenium . . . 34
3.4.3 Arsitektur Sistem . . . 34
3.4.4 Pemodelan Diagram *Use Case* . . . 35

4 Perancangan **37**
4.1 Masukan Program . . . 37
4.1.1 Perancangan Masukan Program . . . 37
4.1.2 Konstruksi Masukan Program . . . 38

4.2 Aktivitas Sistem 39
4.3 Perancangan Algoritma 41

5 Implementasi dan Pengujian **43**
5.1 Implementasi 43
5.1.1 Lingkungan Implementasi 43
5.1.2 Hasil Implementasi 43
5.2 Pengujian 44
5.2.1 Pengujian Fungsional Mahasiswa 44
5.2.2 Pengujian Fungsional Dosen 45
5.2.3 Pengujian Eksperimental 46

6 Kesimpulan dan Saran **49**
6.1 Kesimpulan 49
6.2 Saran 49

Daftar Referensi **51**

A *File* Masukan Untuk Perangkat Lunak **53**
A.1 *File* Konfigurasi Mahasiswa 53
A.2 *File* Konfigurasi Dosen 53

B Kode Program Perangkat Lunak Perekaman Kehadiran Daring Otomatis **55**

C Hasil Pengujian Eksperimental **57**
C.1 Hasil Pengujian Eksperimental Mahasiswa 57
C.2 Hasil Pengujian Eksperimental Dosen 58

DAFTAR GAMBAR

2.1 Alur Komunikasi WebDriver dengan *browser* 5
2.2 Contoh File Konfigurasi Sederhana 12

3.1 Diagram Use Case Portal Akademik Mahasiswa 2018 14
3.2 Tampilan halaman awal Portal Akademik Mahasiswa 14
3.3 Tampilan halaman untuk memasukan *email* Portal Akademik Mahasiswa 15
3.4 Tampilan halaman untuk memasukan *password* Portal Akademik Mahasiswa . . . 15
3.5 Tampilan halaman utama 16
3.6 Tampilan halaman profil mahasiswa 17
3.7 Tampilan halaman pembayaran bagian Tagihan Pembayaran 17
3.8 Tampilan halaman pembayaran bagian Riwayat Pembayaran 18
3.9 Tampilan halaman pembayaran bagian Keterangan 18
3.10 Tampilan halaman nilai bagian Nilai per Semester 19
3.11 Tampilan halaman nilai bagian Riwayat Index Prestasi 19
3.12 Diagram Use Case Fitur Tambahan Portal Akademik Mahasiswa 20
3.13 Diagram Aktivitas Portal Akademik Mahasiswa 2020 21
3.14 Tampilan halaman awal Portal Akademik Mahasiswa 22
3.15 Tampilan halaman Portal Akademik Mahasiswa untuk memasukan *email* 22
3.16 Tampilan halaman Portal Akademik Mahasiswa untuk memasukan *password* . . . 22
3.17 Tampilan peringatan pada halaman Portal Akademik Mahasiswa 23
3.18 Tampilan halaman Portal Akademik Mahasiswa setelah Berhasil *Login* 23
3.19 Tampilan halaman Portal Akademik Mahasiswa untuk Melakukan Absen 24
3.20 Tampilan Pemberitahuan Absensi Berhasil 24
3.21 Diagram Use Case AKUHADIR 24
3.22 Tampilan awal halaman AKUHADIR 26
3.23 Tampilan menu WFH 26
3.24 Tampilan konfirmasi check in AKUHADIR 27
3.25 Tampilan halaman check out AKUHADIR 27
3.26 Tampilan Menu Awal Selenium IDE 28
3.27 Tampilan Memasukan Nama Proyek 28
3.28 Tampilan Memasukan Situs Web 29
3.29 Tampilan Otomatisasi pada Selenium IDE 29
3.30 Histogram Waktu Perekaman Kehadiran Daring Mahasiswa 30
3.31 Histogram Waktu Perekaman Kehadiran Luring Mahasiswa 31
3.32 Histogram Waktu Perekaman Kehadiran Daring Dosen 32
3.33 Histogram Waktu Perekaman Kehadiran Luring Dosen 33
3.34 Diagram Arsitektur 34
3.35 Diagram *Use Case* Sistem Usulan 35

4.1 Gambar Rancangan Untuk Masukan Program 37
4.2 Tampilan Melakukan *Inspect Element* 39
4.3 Diagram Aktivitas Program Absen Daring Otomatis 41

5.1 Tampilan *Command Prompt* dengan *Directory File* 44
5.2 Histogram Waktu Perekaman Kehadiran Daring Otomatis Mahasiswa 46

C.1 Jawaban responden(Mahasiswa) untuk pertanyaan pertama. 57
C.2 Jawaban responden(Mahasiswa) untuk pertanyaan kedua. 57
C.3 Jawaban responden(Mahasiswa) untuk pertanyaan ketiga. 58
C.4 Jawaban responden(Mahasiswa) untuk pertanyaan keempat. 58
C.5 Jawaban responden(Dosen) untuk pertanyaan pertama. 58
C.6 Jawaban responden(Dosen) untuk pertanyaan kedua. 59
C.7 Jawaban responden(Dosen) untuk pertanyaan ketiga. 59
C.8 Jawaban responden(Dosen) untuk pertanyaan keempat. 59

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Perkuliahan di UNPAR biasanya membutuhkan perekaman kehadiran untuk mengetahui kehadiran mahasiswa dan dosen. Perekaman kehadiran bagi mahasiswa UNPAR 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 lebih 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 mumbutuhkan 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.

Terdapat berbagai macam *framework* untuk pengujian otomatisasi, seperti Katalon Studi, Selenium, Appium, dan TestComplete. Pada skripsi ini akan digunakan *framework* Selenium untuk membuat program perekaman kehadiran daring otomatis. Selenium adalah *open-source framework* untuk pengujian otomatisasi pada 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 mengoperasi *browser*, padahal dijalankan secara otomatis langsung oleh WebDriver tersebut. Kelebihan Selenium adalah tersedia untuk berbagai bahasa pemrograman Ruby, Java, Python, C#, dan JavaScript. Selenium dapat dijalankan di berbagai sistem operasi seperti Windows, macOS, Linux, dan Solaris. Pembuatan Perekaman kehadiran daring otomatis ini akan menggunakan Selenium dengan bahasa pemrograman Python.

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

1. Membuka browser.

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

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 program yang dapat melakukan perekaman kehadiran otomatis dengan sistem menerima rangsangan satu "klik" yang dapat menjalankan langkah-langkah perekaman kehadiran daring secara otomatis pada situs https://studentportal.unpar.ac.id, sehingga program yang dibuat ini menjalankan perintah yang biasa dilakukan mahasiswa secara manual untuk melakukan perekaman kehadiran daring menjadi otomatis dilakukan oleh program. Program ini bertujuan agar mahasiswa dapat melakukan perekaman kehadiran secara online di situs web Portal Akademik Mahasiswa UNPAR dengan lebih mudah. Mahasiswa hanya perlu menjalankan program tersebut untuk melakukan perekaman kehadiran daring serta mengurangi waktu yang dibutuhkan untuk berinteraksi dengan aplikasi atau situs web. Program yang dibuat bukan untuk mempercepat waktu agar kehadiran terekam, tetapi 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 perekeman kehadiran milik dosen.

1.2 Rumusan Masalah

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

1. Bagaimana menganalisis kebutuhan program perekaman kehadiran daring otomatis?
2. Bagaimana memodelkan program perekaman kehadiran daring otomatis?
3. Bagaimana mengimplementasikan program perekaman kehadiran daring otomatis dengan *framework* selenium?
4. Bagaimana cara mengurangi waktu interaksi dengan aplikasi atau situs web untuk merekam kehadiran secara otomatis?
5. Bagaimana cara membangun program perekaman kehadiran daring otomatis?

1.3 Tujuan

Tujuan yang ingin dicapai dari penulisan skripsi ini sebagai berikut :

1. Hasil analisis kebutuhan untuk program perekaman kehadiran daring otomatis.
2. Menemukan model untuk program perekaman kehadiran daring otomatis.
3. Mengimplementasikan *framework* selenium untuk program perekaman kehadiran daring otomatis.
4. Membuat program yang mampu mengurangi waktu interaksi dengan aplikasi atau situs web untuk merekam kehadiran secara otomatis.
5. Membangun program perekaman kehadiran daring otomatis dengan Selenium WebDriver.

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.
2. Pengguna harus melakukan *install* Python3 dan mengeksekusi progamnya melalui *Command Prompt*.

1.5 Metodologi

Metodologi yang dilakukan pada skripsi ini adalah sebagai berikut :

1. Melakukan analisis kebutuhan program perekaman kehadiran daring otomatis.
2. Melakukan studi mengenai Selenium.
3. Melakukan studi mengenai *Cascading Style Sheets (CSS) Selector*.
4. Menganalisis web Student Portal UNPAR.
5. Merancang model program perekaman kehadiran daring otomatis.
6. Membangun program perekaman kehadiran daring otomatis dengan *framework* selenium.
7. Melakukan pengujian dan eksperimen.
8. Menulis dokumen skripsi.

1.6 Sistematika Pembahasan

Sistematika penulisan setiap bab skripsi ini adalah sebagai berikut :

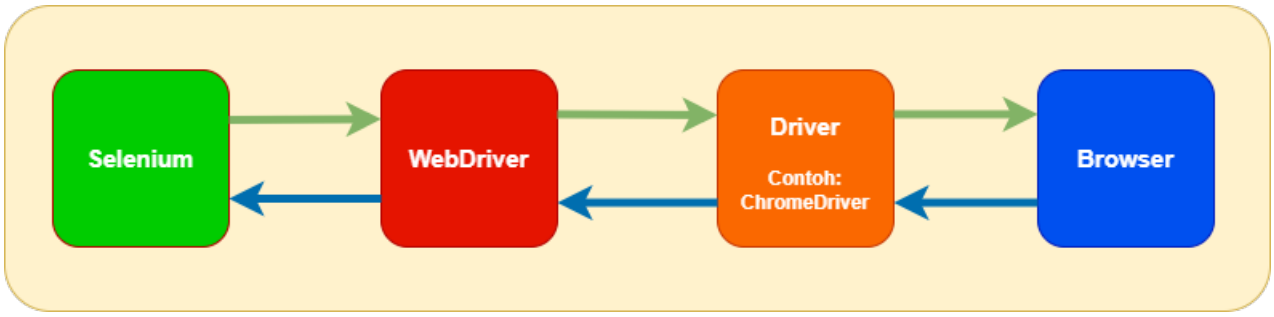
1. Bab 1 Pendahuluan
   Bab ini berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan yang digunakan untuk menyusun skripsi ini.
2. Bab 2 Dasar Teori
   Bab ini berisi teori-teori yang digunakan dalam pembuatan skripsi ini. Teori yang digunakan yaitu Portal Akademik Mahasiswa, AKUHADIR, *Cascading Style Sheets Selector*, Selenium, WebDriver, dan *Library* Python.
3. Bab 3 Analisis Masalah
   Bab ini berisi analisis yang digunakan pada skripsi ini, yaitu analisis hasil survei perekaman kehadiran daring dan luring, analisis alur perekaman kehadirang daring, cara menerjemahkan perekaman kehadiran daring ke dalam selenium, dan analisis program sejenis.
4. Bab 4 Perancangan
   Bab ini berisi perancangan program, meliputi masukan program dan diagram aktivitas.
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 Selenium

Selenium merupakan *open-source framework* untuk pengujian otomatisasi pada *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 2.1: Alur Komunikasi WebDriver dengan *browser*

Pada Gambar 2.1 ini merupakan ilustrasi cara kerja komunikasi Selenium dengan *browser*. Selenium dapat melakukan otomatisasi terhadap suatu *browser* dibantu dengan WebDriver. WebDriver ini berisi perintah-perintah untuk melakukan otomatisasi pada *browser*. Suatu perintah dari WebDriver akan diteruskan kepada *browser* melalui driver *browser* yang digunakan. Driver *browser* ini yang akan meneruskan perintah dari WebDriver kepada *browser* sehingga *browser* dapat diotomatisasi oleh Selenium. Setelah melakukan suatu perintah otomatisasi pada WebDriver maka driver akan mengirimkan kembali hasil informasinya ke pada Selenium melalui alur yang sama. Cara kerja tersebut akan terus berlanjut hingga sudah tidak ada lagi perintah untuk melakukan otomatisasi pada *browser*.

2.1.1 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 mengoperasi *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. Pada subbab ini dijelaskan bagian dari WebDriver yang digunakan untuk pembuatan program perekaman kehadiran daring otomatis.

2.1.2 Interaksi *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*

```
1    driver.title
2    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

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

2.1.3 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[2]. 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 disimpen 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")
```

2.1.4 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 memasukan "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()
```

2.1.5 Waits

Waits merupakan API pemblokiran yang dimiliki WebDriver. *Waits* ini memiliki fungsi untuk menunggu suatu perintah saat melakukan proses otomatisasi terhadapa 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 menunggunya 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.

2.2 *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*[3]. CSS sudah menjadi bagian penting untuk sebuah web untuk suatu elemen dapat ditampilkan pada sebuah web dengan tata letak tertentu, warna tertentu, atau jenis *font* tertentu. Pada contoh kode 2.17 merupakan contoh kode CSS yang dimana elemen "h1" diberi warna merah dan posisi teksnya di tengah serta elemen "p" dengan jenis huruf verdana dan ukuran hurufnya 10 pixel.

Kode 2.17: Contoh kode CSS

```
1   h1 {
2       color: white;
3       text-align: center;
4   }
5
6   p {
7       font-family: verdana;
8       font-size: 10px;
9   }
```

CSS *Selector* digunakan untuk menemukan atau memilih elemen HTML yang diinginkan berdasarkan *style* CSS[1]. 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*.

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

Selector	Contoh	Penjelasan
element element	div p	Memilih semua elemen <p> yang berada di dalam elemen .
element > element	div > p	Memilih semua elemen <p> yang menjadi anak bagian elemen .
element + element	div + p	Memilih elemen <p> pertama setelah elemen .
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*.

Selector	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.

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*.

Selector	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>.

[1] *CSS Selector* https://www.w3schools.com/css/css_selectors.asp

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*.

Selector	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".

2.3 *Configuration File Parser* (ConfigParser)

ConfigParser merupakan sebuah modul yang sudah tersedia pada bahasa pemrograman Python dan mengimplementasikan bahasa konfigurasi dasar[4]. 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 2.2: Contoh File Konfigurasi Sederhana

Pada Gambar 2.2 merupakan contoh file konfigurasi sederhana. Tulisan "[Database]" pada gambar 2.2 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 2.2:

- 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".

2.4 *Library* OS

Library OS adalah modul sistem operasi pada python untuk program 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 *enviroment variable* milik pengguna.
- os.getcwd(): Mengembalikan direktori kerja yang digunakan.
- os.name(): Menampilkan nama modul sistem operasi yang diimpornya.
- os.rename(): Mengganti nama *file* yang lama dan digantikan dengan nama *file* yang baru.
- os.listdir(): Mengembalikan semua daftar file dan direktori yang ditentukan.
- os.rmdir(): Menghapus suatu direktori yang telah ditentukan.

BAB 3

ANALISIS

Bab ini berisi analisis yang digunakan pada skripsi ini, yaitu analisis masalah, analisis sistem kini, analisis sistem sejenis, dan analisis sistem usulan.

3.1 Analisis Masalah

Pada penelitian ini, masalah yang ingin coba diselesaikan adalah membuat perekaman kehadiran daring secara manual menjadi perekaman kehadiran daring secara otomatis yang dapat mengurangi waktu interaksi pengguna dengan browser. Perekaman kehadiran dibagi menjadi dua jenis, yaitu perekaman kehadiran luring dan daring. Waktu perekaman kehadiran daring lebih lama dibandingkan perekaman kehadiran luring. Perekaman kehadiran daring butuh waktu yang lebih lama karena perlu interaksi dengan *browser* dalam melakukan perekaman kehadiran daring.

Mahasiswa biasanya dapat melakukan perekaman kehadiran daring melalui *handphone* ataupun komputer. Perekaman kehadiran daring melalui komputer bagi mahasiswa sering kali butuh waktu yang lama dalam melakukan perekaman kehadiran. Penyebabnya adalah mahasiswa perlu melakukan perekaman kehadiran melalui *browser* dan berinteraksi dengan *browser*, seperti :

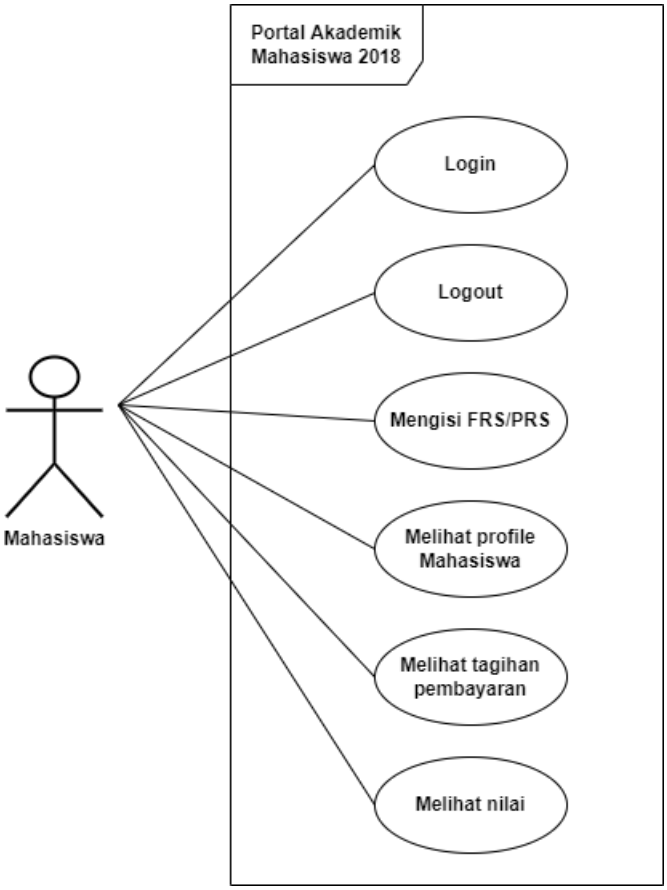
- Membuka browser.
- Membuka situs untuk melakukan perekaman kehadiran.
- Melakukan *login*, termasuk memasukan *email* dan *password*.
- Menuju halaman web bagian perekaman kehadiran daring.
- Melakukan perekaman kehadiran daring.

Pada penelitian ini akan menghasilkan program yang mampu melakukan perekaman kehadiran daring secara otomatis melalui komputer dan mengurangi waktu interaksi dengan *browser*. Program menggunakan *framework* selenium untuk dapat melakukan otomatisasi pada *browser* Google Chrome.

3.2 Analisis Sistem Kini

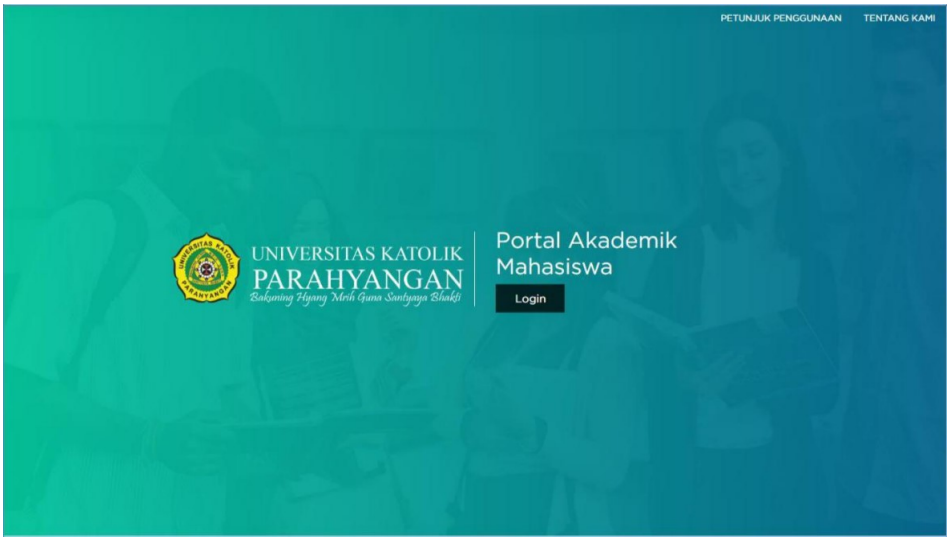
3.2.1 Portal Akademik Mahasiswa 2018

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[5]. Portal Akademik Mahasiswa dapat diakses melalui `https://studentportal.unpar.ac.id/`. Diagram *use case* dapat dilihat di Gambar 3.1.



Gambar 3.1: Diagram Use Case Portal Akademik Mahasiswa 2018

Setelah penggambaran *use case* diagram perlu dijelaskan skenario dari *use case* diagram tersebut. Skenario use case merupakan alur jalannya proses *use case* dari sisi aktor maupun sistemnya. Berikut ini merupakan skenario *use case* yang disajikan dalam bentuk tabel.

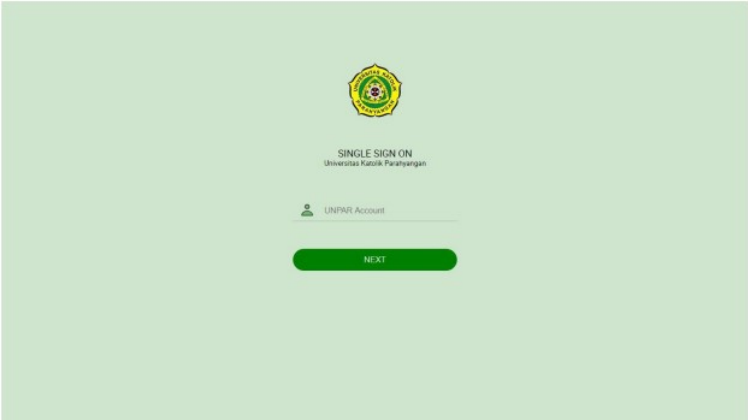


Gambar 3.2: Tampilan halaman awal Portal Akademik Mahasiswa

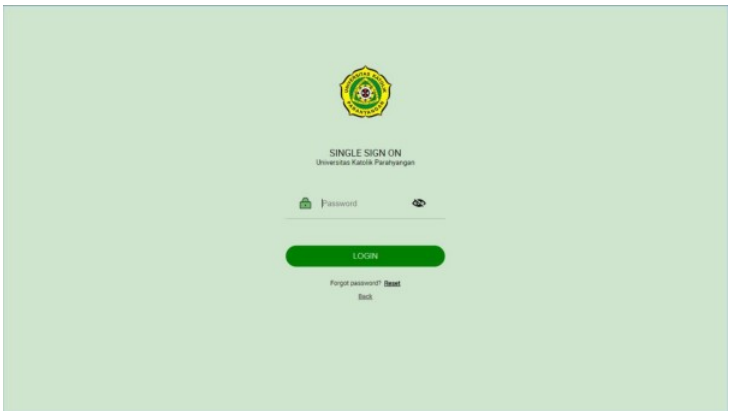
Pada Gambar 3.2 adalah tampilan awal ketika masuk ke halaman https://studentportal.unpar.ac.id/. Fitur-fitur yang tersedia pada Portal Akademik Mahasiswa sebagai berikut:

1. *Login*: Untuk dapat menggunakan situs Portal Akademik Mahasiswa, mahasiswa UNPAR harus *login* menggunakan *email* dan *password* milik mahasiswa tersebut.
   - Nama Use Case: *Login*
   - Aktor: Mahasiswa
   - Deskripsi: *Login* ke Portal Akademik Mahasiswa.
   - Kondisi awal: Mahasiswa belum *login*.
   - Kondisi akhir: Halaman utama Portal Akademik Mahasiswa.
   - Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Mahasiswa mengakses Portal Akademik Mahasiswa dan menekan tombol "Login"	Sistem menampilkan halaman *login*.
2	Mahasiswa mengisi *email* dan menekan tombol "Next"	Sistem menampilkan halaman *input password*.
3	Mahasiswa mengisi *password* dan menekan tombol "LOGIN"	Sistem menampilkan halaman utama Portal Akademik Mahasiswa.



Gambar 3.3: Tampilan halaman untuk memasukan *email* Portal Akademik Mahasiswa



Gambar 3.4: Tampilan halaman untuk memasukan *password* Portal Akademik Mahasiswa

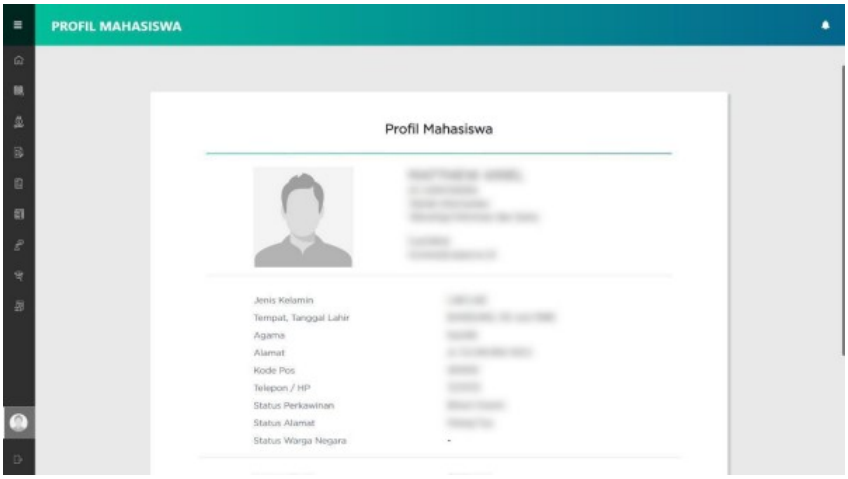
Gambar 3.5: Tampilan halaman utama

2. Fitur FRS/PRS: Untuk mengisi form rencana semester (FRS) atau melakukan perubahan rencana studi (PRS) secara online.
   - Nama Use Case: FRS/PRS
   - Aktor: Mahasiswa
   - Deskripsi: Melakukan FRS/PRS Portal Akademik Mahasiswa.
   - Kondisi awal: Mahasiswa telah *login*.
   - Kondisi akhir: Halaman FRS/PRS pada Portal Akademik Mahasiswa.
   - Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Mahasiswa memilih menu “FRS/PRS”	Sistem menampilkan halaman FRS/PRS.

3. Fitur Profil Mahasiswa: Untuk melihat profil milik mahasiswa itu sendiri.
   - Nama Use Case: Profil
   - Aktor: Mahasiswa
   - Deskripsi: Melihat data diri mahasiswa pada Portal Akademik Mahasiswa.
   - Kondisi awal: Mahasiswa telah *login*.
   - Kondisi akhir: Halaman Profil pada Portal Akademik Mahasiswa.
   - Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Mahasiswa memilih menu Profil	Sistem menampilkan halaman Profil.

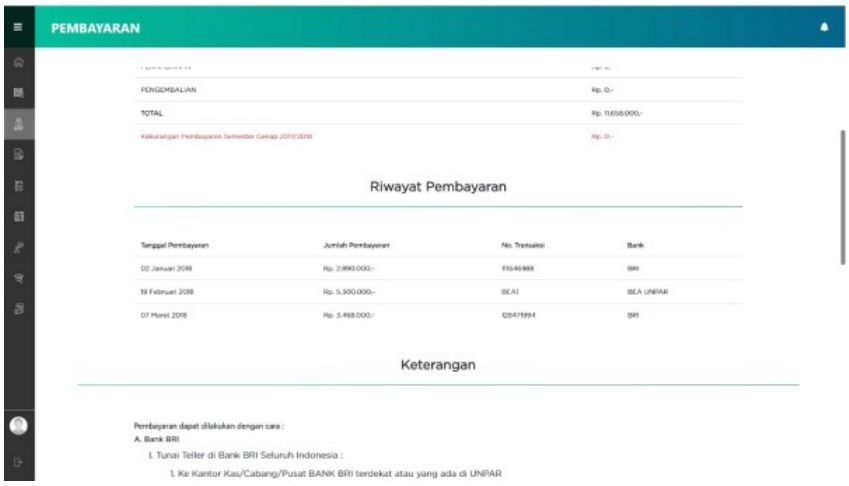


Gambar 3.6: Tampilan halaman profil mahasiswa

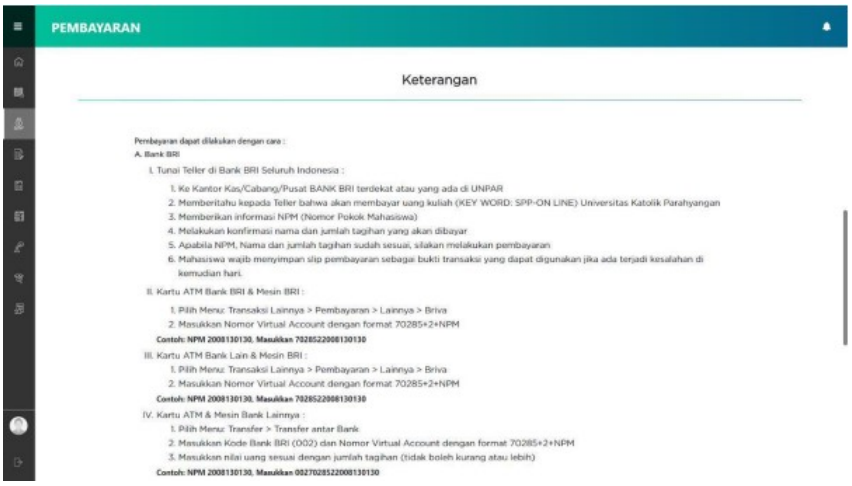
4. Fitur Pembayaran: Menampilkan jenis tagihan dan jumlah tagihan dari setiap jenis tagihan yang ada.
   - Nama Use Case: Pembayaran
   - Aktor: Mahasiswa
   - Deskripsi: Melihat jenis tagihan dan jumlah tagihan pembayaran semester milik mahasiswa pada Portal Akademik Mahasiswa.
   - Kondisi awal: Mahasiswa telah *login*
   - Kondisi akhir: Halaman pembayaran pada Portal Akademik Mahasiswa.
   - Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Mahasiswa memilih menu Pembayaran	Sistem menampilkan halaman Pembayaran.

Gambar 3.7: Tampilan halaman pembayaran bagian Tagihan Pembayaran



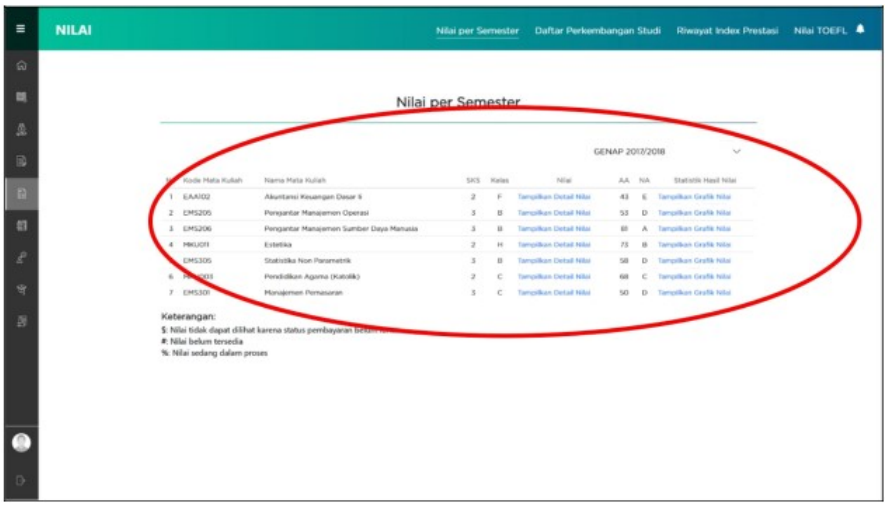
Gambar 3.8: Tampilan halaman pembayaran bagian Riwayat Pembayaran



Gambar 3.9: Tampilan halaman pembayaran bagian Keterangan

5. Fitur Nilai: Melihat semua nilai milik mahasiswa dari setiap semester.
   - Nama Use Case: Nilai
   - Aktor: Mahasiswa
   - Deskripsi: Melihat nilai milik mahasiswa.
   - Kondisi awal: Mahasiswa telah *login*
   - Kondisi akhir: Halaman nilai mahasiswa pada Portal Akademik Mahasiswa.
   - Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Mahasiswa memilih menu Nilai	Sistem menampilkan halaman Nilai per semester.
2	Mahasiswa memilih menu Riwayat Index Prestasi	Sistem menampilkan halaman Riwayat Index Prestasi.



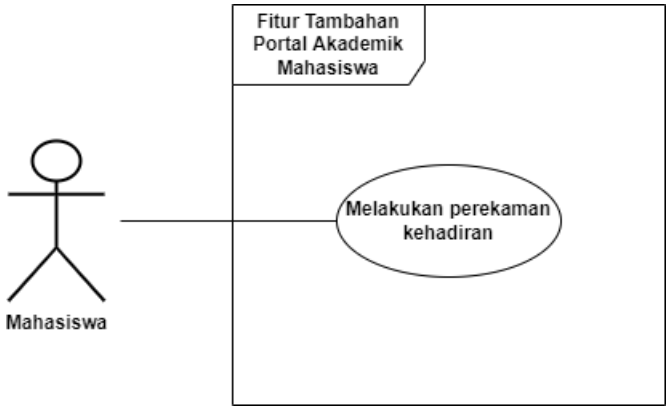
Gambar 3.10: Tampilan halaman nilai bagian Nilai per Semester

Gambar 3.11: Tampilan halaman nilai bagian Riwayat Index Prestasi

Portal Akademik Mahasiswa 2018 belum memiliki fitur perekaman kehadiran secara daring. Hal tersebut dikarenakan belum adanya pembelajaran secara daring, sehingga perekaman kehadiran dilakukan secara luring yang dimana perekaman kehadiran dilakukan secara langsung saat pembelajaran tatap muka.

3.2.2 Fitur Tambahan Perekaman Kehadiran Portal Akademik Mahasiswa

Portal Akademik Mahasiswa Universitas Katolik Parahyangan yang terbaru sejak 2020 sudah memiliki fitur perekaman kehadiran daring. Fitur tersebut dapat melakukan perekaman kehadiran secara daring untuk setiap mata kuliah yang diambil. Diagram *use case* dilihat di Gambar 3.12.



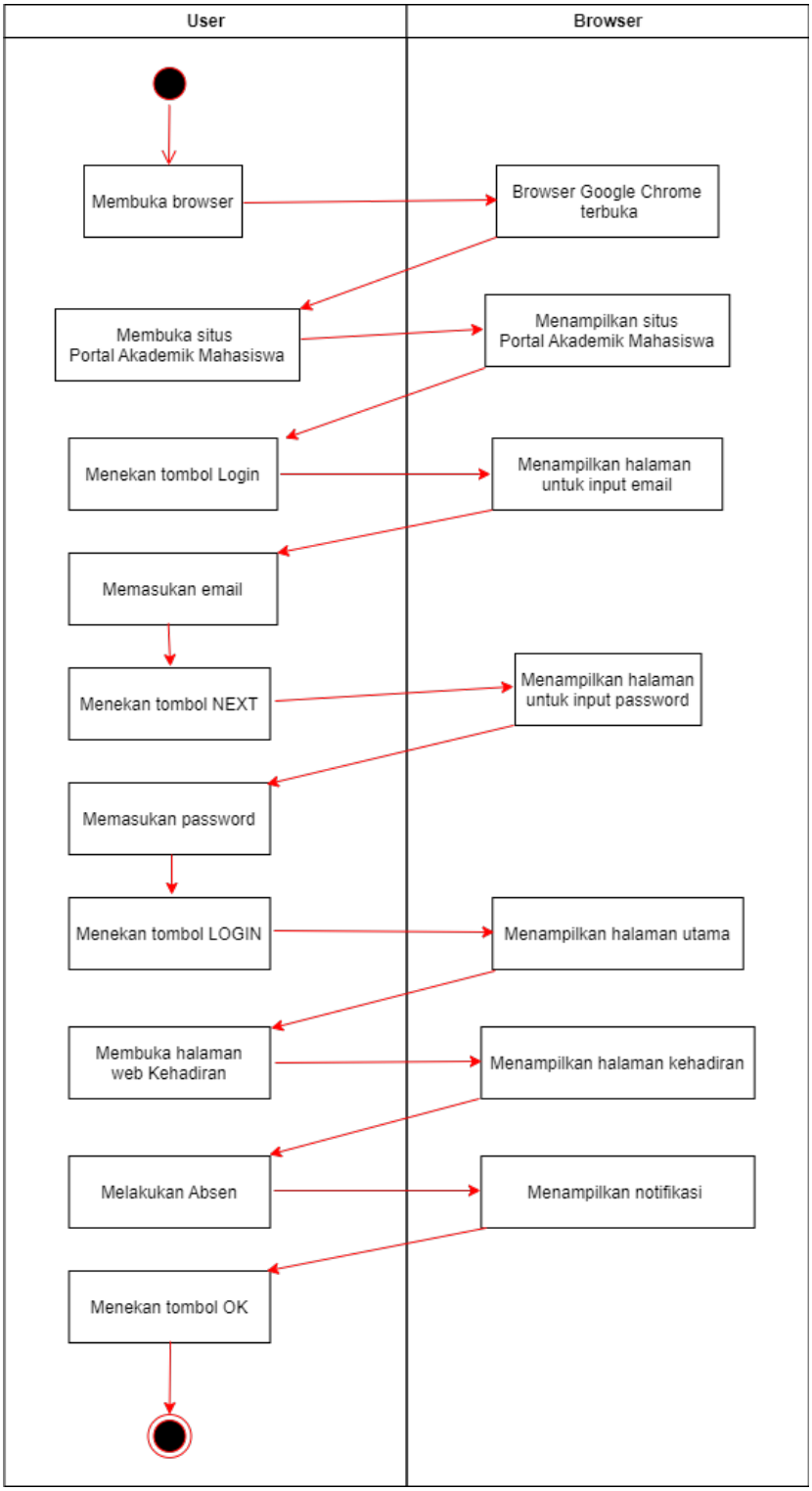
Gambar 3.12: Diagram Use Case Fitur Tambahan Portal Akademik Mahasiswa

Skenario *use case* ini merupakan tambahan lanjutan fitur dari use case pada subbab 3.2.1 yang tidak memiliki fitur perekaman kehadiran daring.

- Nama Use Case: Jadwal & Kehadiran
- Aktor: Mahasiswa
- Deskripsi: Melihat jadwal mata kuliah dan melakukan absensi daring pada Portal Akademik Mahasiswa.
- Kondisi awal: Mahasiswa telah *login*
- Kondisi akhir: Halaman Jadwal & Kehadiran dan mahasiswa berhasil melakukan absensi daring.
- Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Mahasiswa memilih menu Jadwal & Kehadiran	Sistem menampilkan halaman Jadwal & Kehadiran per-semester (Gambar 3.19).
2	Mahasiswa menekan tombol berwarna merah pada kolom presensi pada tabel jadwal kehadiran	Sistem menampilkan notifikasi mengenai absensi berhasil dilakukan (Gambar3.20).
3	Mahasiswa menekan tombol “OK”	Sistem menutup notifikasi.

Diagram aktivitas untuk perekaman kehadiran daring pada Portal Akademik Mahasiswa dapat dilihat di Gambar 3.13. Diagram aktifitas tersebut menunjukan bagaimana alur dalam melakukan perekaman kehadiran daring secara manual. Pada diagram aktivitas dapat dilihat bahwa banyak urutan langkah yang perlu dilakukan oleh mahasiswa untuk melakukan perekaman kehadiran daring. Diagram aktivitas ini untuk dapat dibandingkan perbedaannya dengan program yang akan dibuat, sehingga mengetahui langkah-langkah apa saja yang dipersingkat untuk diotomatisasi.

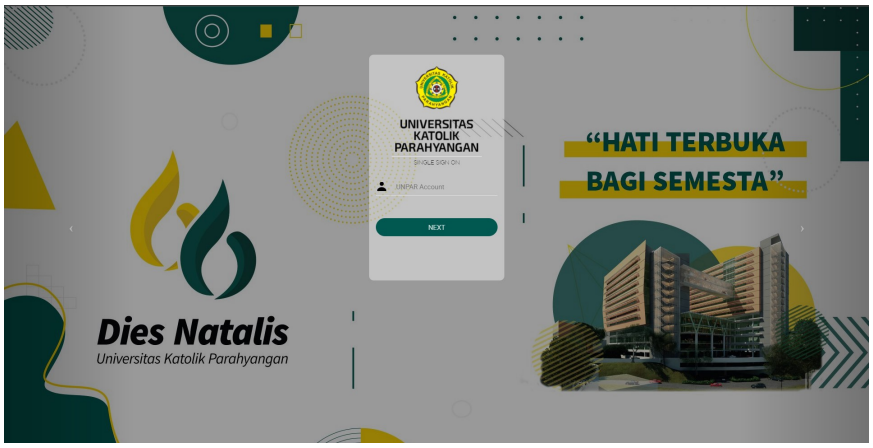


Gambar 3.13: Diagram Aktivitas Portal Akademik Mahasiswa 2020

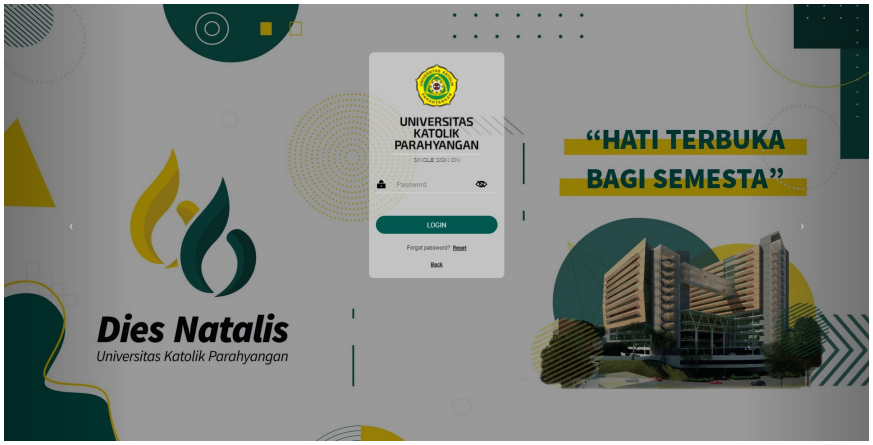
Setiap mahasiswa sebelum memulai perkuliahan pada setiap mata kuliah perlu mengakses Portal Akademik Mahasiswa yang dapat diakses melalui `https://studentportal.unpar.ac.id/` seperti pada Gambar 3.14.

Gambar 3.14: Tampilan halaman awal Portal Akademik Mahasiswa

Setelah itu mahasiswa melakukan *Login* dengan mengisi *email* serta *password* milik mahasiswa tersebut. Pada Gambar 3.15 merupakan tampilan untuk mengisi *email* dan Gambar 3.16 merupakan tampilan untuk mengisi *password*.

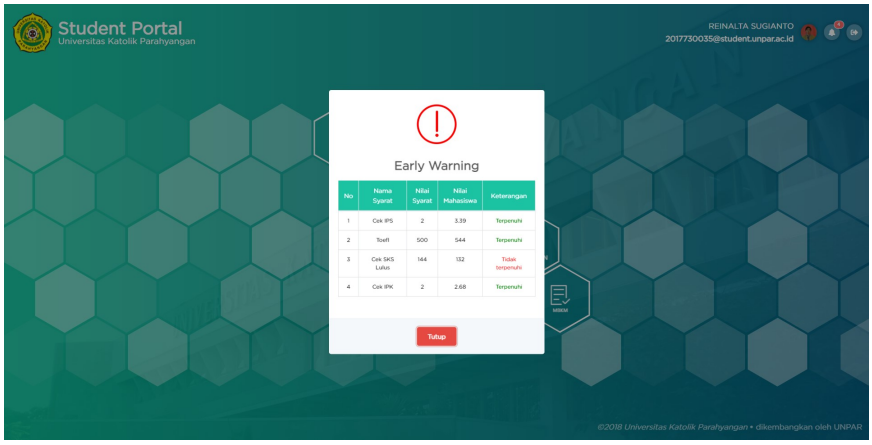


Gambar 3.15: Tampilan halaman Portal Akademik Mahasiswa untuk memasukan *email*



Gambar 3.16: Tampilan halaman Portal Akademik Mahasiswa untuk memasukan *password*

Setelah berhasil melakukan *login* akan ada dua kemungkinan yang terjadi pada halaman Portal Akademik Mahasiswa, yaitu pertama akan muncul notifkasi peringatan seperti pada Gambar 3.17 dan kedua akan masuk langsung ke halaman utama Portal Akademik Mahasiswa seperti pada Gambar 3.18. Jika muncul notifikasi terlebih dahulu maka harus menutup notifikasi tersebut untuk dapat masuk ke halaman utama.

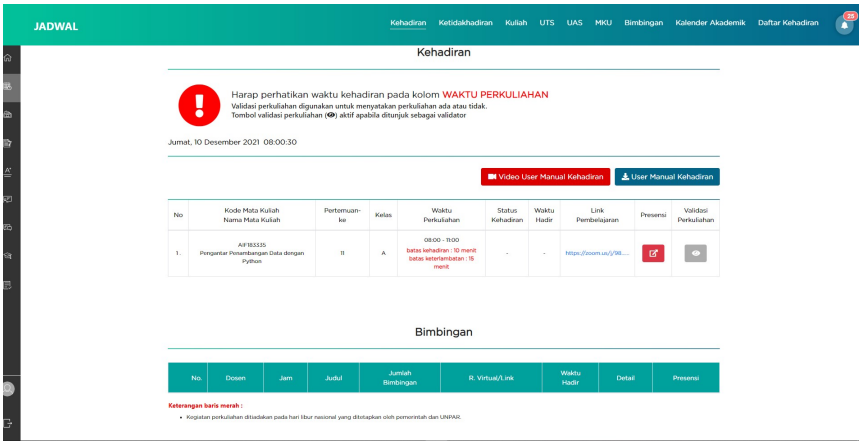


Gambar 3.17: Tampilan peringatan pada halaman Portal Akademik Mahasiswa

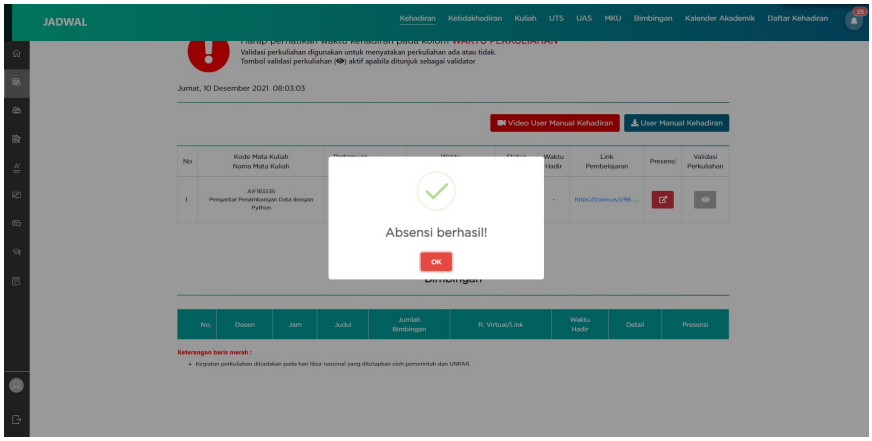
Gambar 3.18: Tampilan halaman Portal Akademik Mahasiswa setelah Berhasil *Login*

Pada Gambar 3.17 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 3.18. Jika tidak terjadi peringatan seperti pada Gambar 3.17, maka dapat langsung menekan tombol berbentuk heksagon "JADWAL & KEHADIRAN" seperti pada Gambar 3.18.

Setelah berhasil masuk ke halaman untuk perekaman kehadiran daring, mahasiswa perlu menekan tombol berwarna merah pada kolom bagian presensi pada tabel jadwal kehadiran mata kuliah (Gambar 3.19). Setelah mengklik tombol presensi maka akan muncul notifikasi bahwa absensi telah berhasil dan mahasiswa perlu menekan tombol "OK".



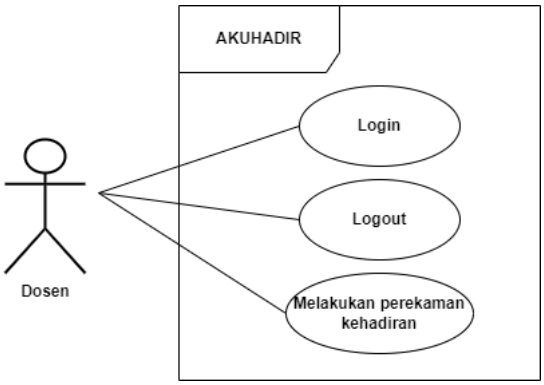
Gambar 3.19: Tampilan halaman Portal Akademik Mahasiswa untuk Melakukan Absen



Gambar 3.20: Tampilan Pemberitahuan Absensi Berhasil

3.2.3 AKUHADIR 2.1

Subbab ini ditulis dari hasil mewawancarai dosen pembimbing, karena penulis tidak memiliki akses terhadap AKUHADIR. AKUHADIR adalah sebuah portal web yang dibuat bagi pegawai UNPAR dalam melaporkan kehadiran kerja nya secara daring. Pegawai UNPAR melakukan perekaman kehadirannya setiap hari melalui portal AKUHADIR 2.1 yang dapat diakses pada `https://akuhadir.unpar.ac.id`. Hal tersebut sesuai surat edaran dari Rektor III/R/2020-07/1153 [6]. Diagram *use case* dapat dilihat pada Gambar 3.21.



Gambar 3.21: Diagram Use Case AKUHADIR

Setelah penggambaran *use case* diagram perlu dijelaskan skenario dari *use case* diagram tersebut. Skenario use case merupakan alur jalannya proses *use case* dari sisi aktor maupun sistemnya. Berikut ini merupakan skenario *use case* yang disajikan dalam bentuk tabel.

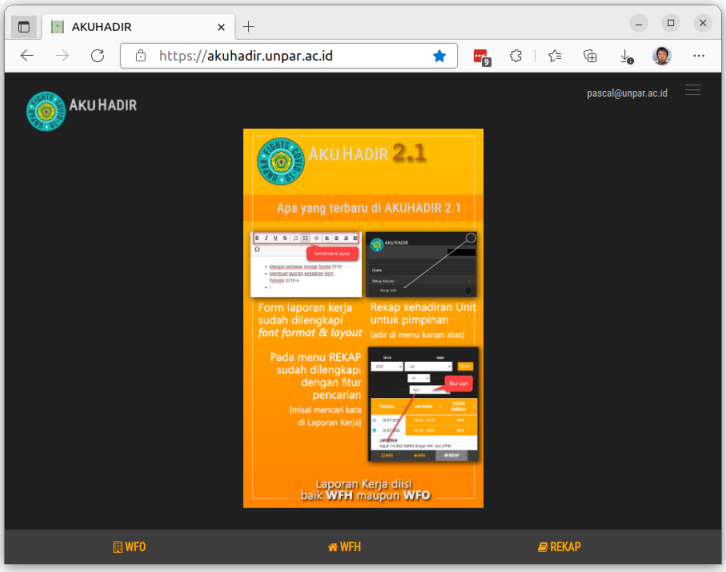
1. Fitur *Login*: Untuk dapat menggunakan situs AKUHADIR, dosen UNPAR harus *login* menggunakan *email* dan *password* milik dosen tersebut.
   - Nama Use Case: *Login*
   - Aktor: Dosen
   - Deskripsi: *Login* ke situs AKUHADIR.
   - Kondisi awal: Dosen belum *login*.
   - Kondisi akhir: Halaman utama AKUHADIR.
   - Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Dosen mengakses AKUHADIR	Sistem menampilkan halaman *login*.
2	Dosen mengisi *email* dan menekan tombol "Next"	Sistem menampilkan halaman *input password*.
3	Dosen mengisi *password* dan menekan tombol "LOGIN"	Sistem menampilkan halaman utama AKUHADIR.

2. Fitur WFH: Melakukan absen daring bagi dosen pada situs AKUHADIR.
   - Nama Use Case: *WFH*
   - Aktor: Dosen
   - Deskripsi: Melakukan absensi daring pada situs AKUHADIR.
   - Kondisi awal: Dosen telah *login*.
   - Kondisi akhir: Halaman WFH dan dosen berhasil melakukan absensi daring.
   - Skenario utama:

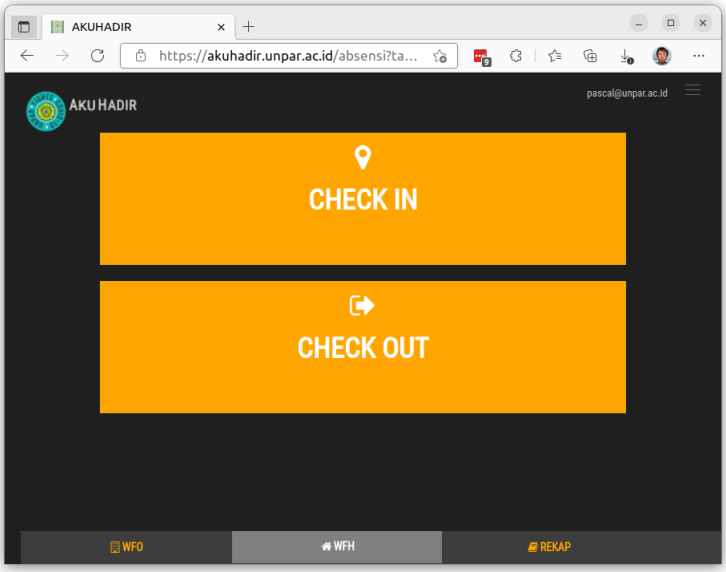
No	Aksi Aktor	Reaksi Sistem
1	Dosen memilih menu WFH	Sistem menampilkan halaman WFH.
2	Dosen menekan tombol *CHECK IN*	Sistem menampilkan pesan konfirmasi bahwa absensi dosen berhasil.

Gambar 3.22 menunjukkan halaman awal portal AKUHADIR. Pegawai UNPAR melakukan *login* melalui SSO (*Single Sign On*) UNPAR untuk dapat melakukan perekaman kehadirannya. Setelah melakukan login para pegawai UNPAR dapat melakukan perekaman kehadiran secara daring dengan memilih menu WFH pada bagian bawah layar. Menu WFH ini akan membuka halaman di mana pegawai UNPAR dapat melakukan perekaman kehadiran. Halaman tersebut terdapat dua buah tombol, yaitu: satu tombol untuk *check in* dan satu tombol lagi untuk *check out*. Halaman dari menu WFH dapat dilihat pada Gambar 3.23.

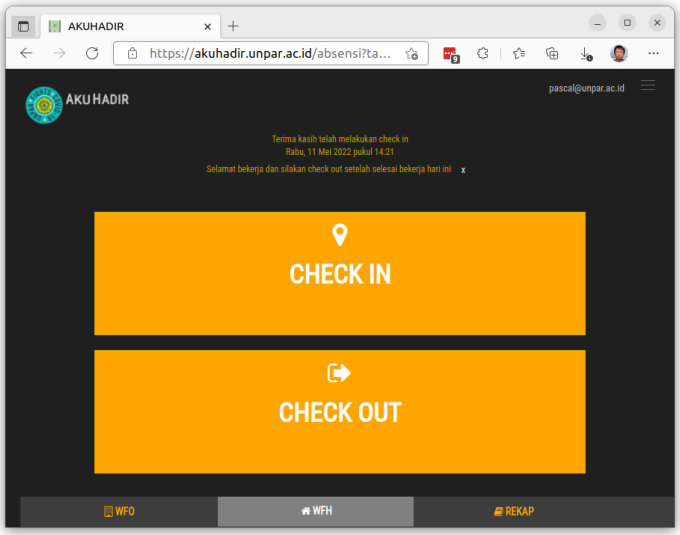


Gambar 3.22: Tampilan awal halaman AKUHADIR

Pegawai UNPAR sebelum memulai bekerja di pagi hari harus melakukan perekaman kehadiran melalui AKUHADIR. Pegawai membuka situs AKUHADIR dan *login* sesuai dengan *email* serta *password* milik pegawai. Pegawai mengklik menu WFH untuk masuk ke bagian halaman perekaman kehadiran milik situs AKUHADIR. Selanjutnya pegawai mengklik menu *Check in*. Setelah tombol tersebut diklik, akan muncul pesan konfirmasi bahwa *check in* sudah berhasil dilakukan (dapat dilihat pada Gambar 3.24).

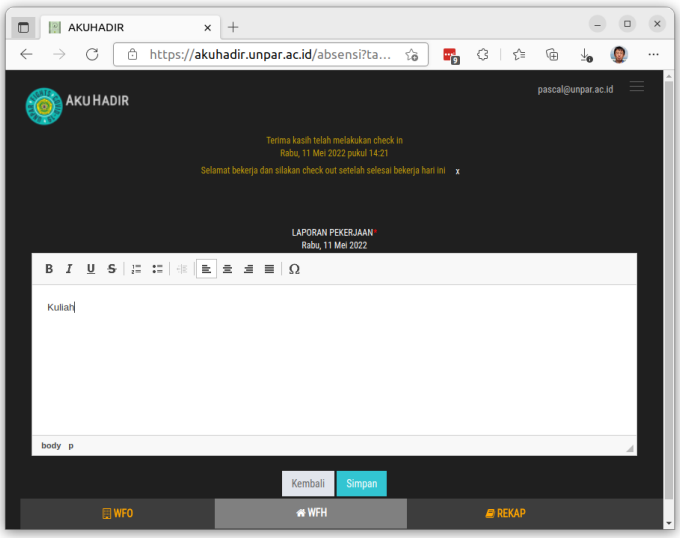


Gambar 3.23: Tampilan menu WFH



Gambar 3.24: Tampilan konfirmasi check in AKUHADIR

Setelah selesai bekerja, pegawai perlu kembali mengakses AKUHADIR dan masuk ke halaman perekaman kehadiran dari menu WFH. Lalu pegawai memilih menu *check out*. Setelah melakukan *check out*, pegawai UNPAR diminta untuk menuliskan laporan hasil pekerjaan yang sudah dilakukan pada hari tersebut. Bentuk halaman untuk menuliskan laporan hasil pekerjaan pada Gambar 3.25.



Gambar 3.25: Tampilan halaman check out AKUHADIR

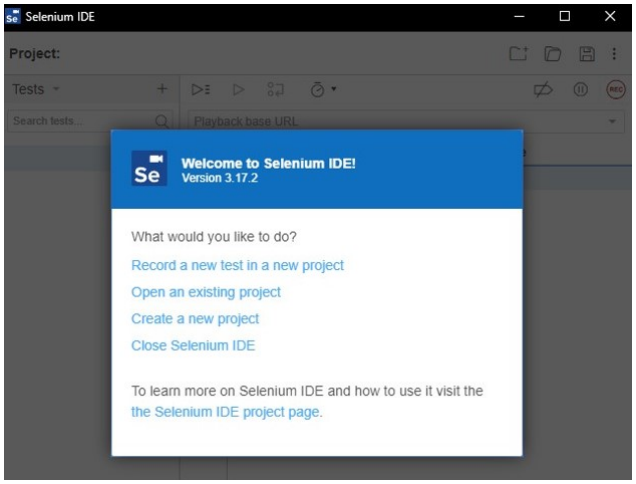
Selain fitur untuk melakukan perekaman kehadiran daring pada AKUHADIR, terdapat beberapa fitur lainnya yang tidak dijelaskan lebih mendalam di sini karena tidak terkait erat dengan penelitian yang dilakukan:

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

3.3 Analisis Sistem Sejenis

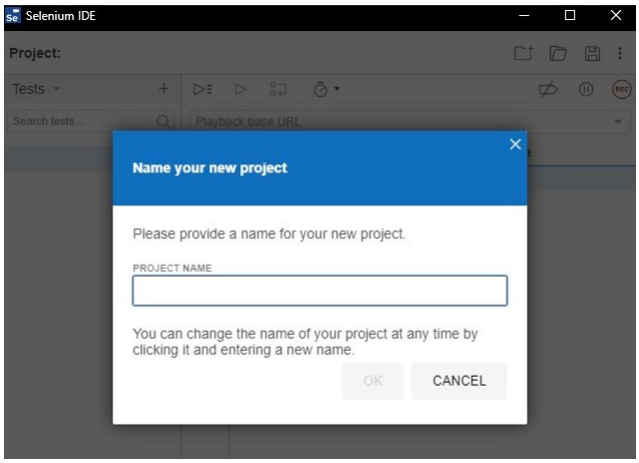
3.3.1 Selenium IDE

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. Selenium IDE tersimpan di *extensions* pada Google Chrome. Pengguna yang ingin melakukan otomatisasi menggunakan Selenium IDE perlu memilih menu *Record a new test in a new project* (merekam tes baru untuk proyek baru) seperti pada Gambar 3.26.

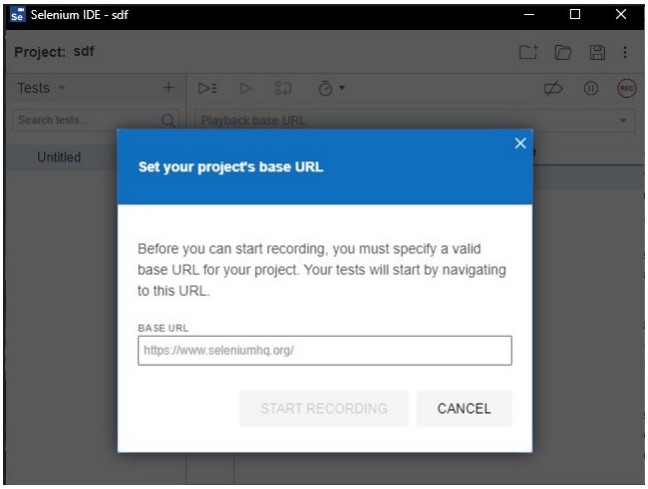


Gambar 3.26: Tampilan Menu Awal Selenium IDE

Setelah memilih menu ,pengguna perlu memasukan nama proyek sesuai keinginan pengguna (Gambar 3.27). Sehabis memasukan nama proyek, pengguna perlu memasukan situs web yang ingin diotomatisasi untuk dapat memulai melakukan otomatisasi pada situs yang dituju menggunakan Selenium IDE dan menekan tombol "*START RECORDING*". Setelah menekan tombol "*START RECORDING*" seperti pada Gambar 3.28, maka akan langsung muncul *windows* Google Chrome baru yang langsung menuju situs web yang sudah dimasukan.

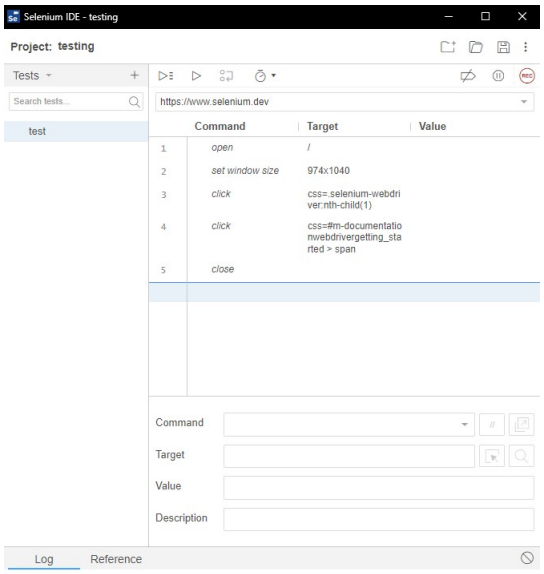


Gambar 3.27: Tampilan Memasukan Nama Proyek



Gambar 3.28: Tampilan Memasukan Situs Web

Pengguna perlu melakukan hal-hal yang ingin diotomatisasi hingga selesai pada *windows* Google Chrome baru yang sudah menuju situs web agar dapat terekam oleh Selenium IDE. Pada Gambar 3.29 menunjukan hasil yang sudah terekam dari apa yang sudah dilakukan pada situs web yang ingin diotomatisasikan.



Gambar 3.29: Tampilan Otomatisasi pada Selenium IDE

Selenium IDE ini dapat digunakan untuk melakukan perekaman kehadiran otomatis pada Portal Akademik Mahasiswa. Program yang dibuat pada skripsi ini akan sejenis dengan Selenium IDE, namun program pada skripsi ini akan dapat dijalankan langsung melalui *Command Prompt*. Pada program yang dibuat di skripsi ini membuat pengguna tidak perlu melakukan seperti pada Selenium IDE. Pengguna cukup menjalankan melalui *Command Prompt* saja.

3.4 Analisis Sistem Usulan

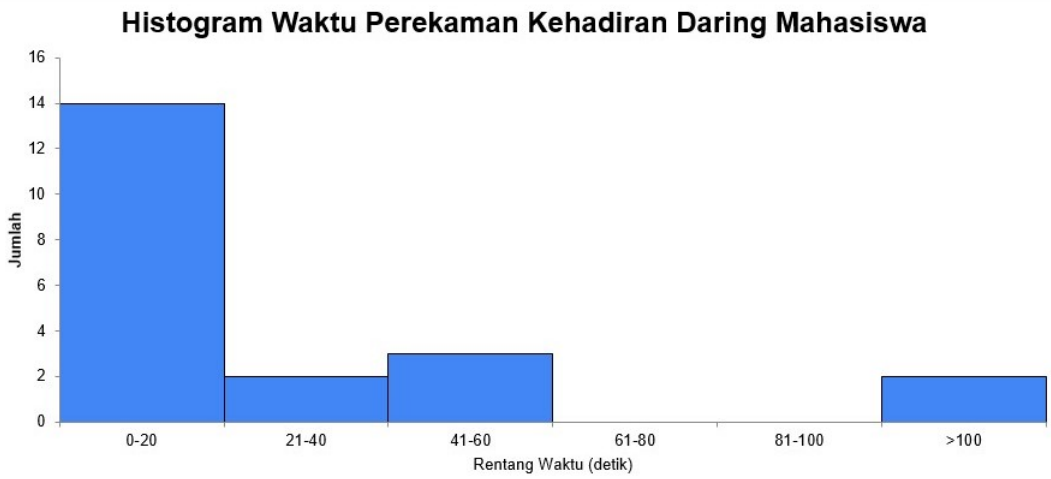
3.4.1 Analisis Hasil Survei Perekaman Kehadiran Daring dan Luring

Survei perekaman kehadiran daring dan luring dilakukan untuk mengetahui berapa lama waktu yang dibutuhkan untuk melakukan perekaman kehadiran secara daring maupun luring dan beberapa informasi dalam melakukan perekaman kehadiran daring. Survei ini diberikan kepada mahasiswa dan dosen Teknik Informatika Universitas Katolik Parahyangan. Hasil survei menunjukan 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.

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 responen 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 3.30: Histogram Waktu Perekaman Kehadiran Daring Mahasiswa

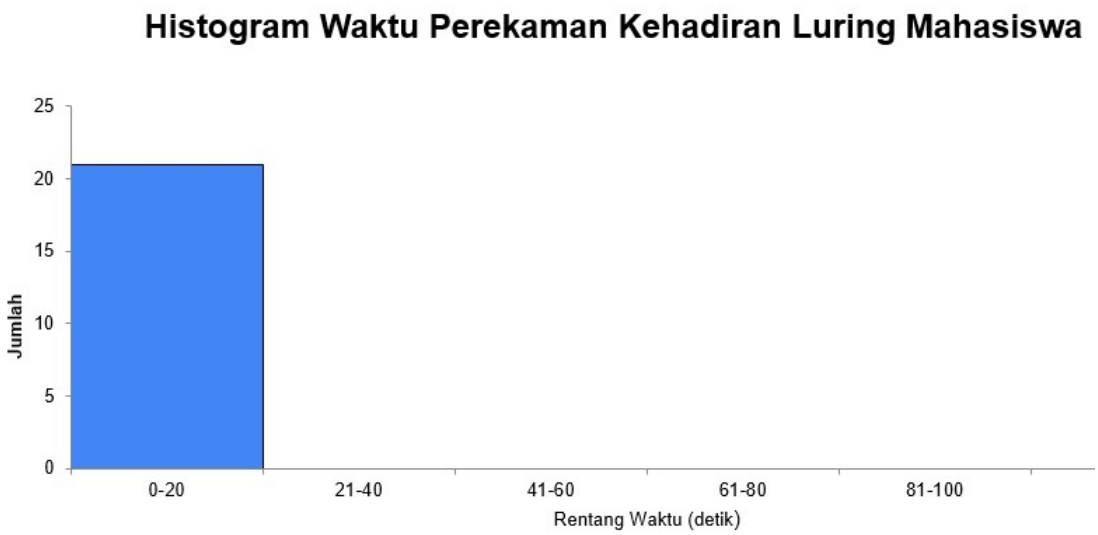
Pada Gambar 3.30 merupakan visualisasi dari hasil survei mengenai lama waktu yang dibutuhkan dari 21 mahasiswa untuk melakukan perekaman kehadiran secara daring. Histogram ini dikelompokan berdasarkan rentang waktu per 20 detik. Histogram tersebut menunjukan bahwa mayoritas mahasiswa sebanyak 14 orang memiliki rentang waktu mulai dari 0 sampai 20 detik melakukan perekaman kehadiran secara daring, sebanyak 2 orang memilik 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 perekeman 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?**



Gambar 3.31: Histogram Waktu Perekaman Kehadiran Luring Mahasiswa

Pada Gambar 3.31 merupakan visualisasi dari hasil survei mengenai lama waktu yang dibutuhkan dari 21 mahasiswa untuk melakukan perekaman kehadiran secara luring. Histogram ini dikelompokan berdasarkan rentang waktu per 20 detik. Histogram tersebut menunjukan bahwa seluruh mahasiswa sebanyak 21 orang memiliki rentang waktu mulai dari 0 sampai 20 detik melakukan perekaman kehadiran secara daring. Hasil survei perekeman kehadiran secara luring untuk setiap mahasiswa secara jelas dapat dilihat pada tabel 3.2. Jawaban dari 21 orang responden adalah mulai dari waktu tercepat 5 detik hingga waktu terlama 15 detik. Jika dihitung rata-rata waktu yang dibutuhkan untuk melakukan perekaman kehadiran luring bagi para mahasiswa adalah 7,95 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

Berdasarkan hasil survei terdapat beberapa informasi yang dirasakan mahasiswa ketika dalam melakukan perekaman kehadiran daring. Faktor yang menjadi kendala dalam melakukan perekaman kehadiran daring bagi mahasiswa, sebagai berikut:

1. Faktor waktu berpengaruh pada kecepatan dalam melakukan perekaman kehadiran daring. Perkuliahan mahasiswa pada jam-jam padat, seperti jam 7 pagi, 9 pagi, 12 siang, atau 1 siang ini biasanya mahasiswa akan mengalami kendala waktu yang lama untuk dapat melakukan perekaman kehadiran daring.
2. Faktor internet berpengaruh pada kecepatan dalam perekaman kehadiran daring. Setiap mahasiswa pasti menggunakan internet dari provider yang berbeda sehingga waktu yang dibutuhkan dalam melakukan perekaman kehadiran daring bergantung pada internet yang digunakan oleh mahasiswa.

Kesimpulan dari hasil survei mahasiswa menunjukan 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.

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 responen 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` ?**
   Pada Gambar 3.32 merupakan visualisasi dari hasil survei mengenai lama waktu yang dibutuhkan dari 6 dosen untuk melakukan perekaman kehadiran secara daring. Histogram ini dikelompokan berdasarkan rentang waktu per 20 detik. Histogram menunjukan 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.

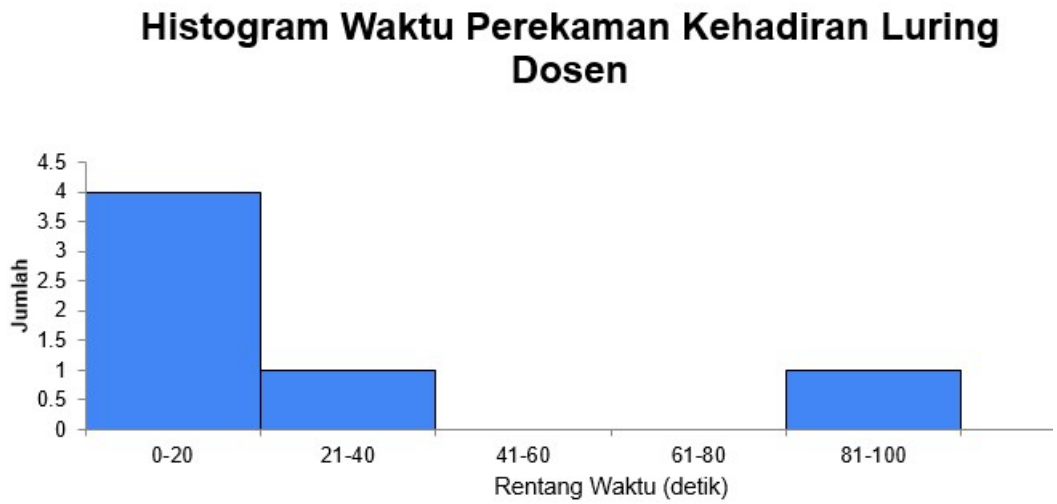
Gambar 3.32: Histogram Waktu Perekaman Kehadiran Daring Dosen

Hasil survei perekeman kehadiran daring untuk setiap dosen secara jelas dapat dilihat pada tabel 3.3. Jika dihitung rata-rata waktu yang dibutuhkan untuk melakukan perekaman kehadiran daring bagi para dosen adalah 31,83 detik.

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

2. **Berapa detik perkiraan waktu interaksi yang Anda butuhkan untuk melakukan perekaman kehadiran luring menggunakan metode fingerprint?**
Pada Gambar 3.33 merupakan visualisasi dari hasil survei mengenai lama waktu yang dibutuhkan dari 6 dosen untuk melakukan perekaman kehadiran secara luring. Histogram ini dikelompokan berdasarkan rentang waktu per 20 detik. Histogram menunjukan 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.



Gambar 3.33: Histogram Waktu Perekaman Kehadiran Luring Dosen

Jika dihitung rata-rata waktu yang dibutuhkan untuk melakukan perekaman kehadiran luring bagi para dosen adalah 24,33 detik.Hasil survei perekeman kehadiran luring untuk setiap dosen secara jelas dapat dilihat pada tabel 3.4. Kesimpulan dari hasil survei dosen menunjukan 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.

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

3.4.2 Perekaman Kehadiran Daring ke dalam Selenium

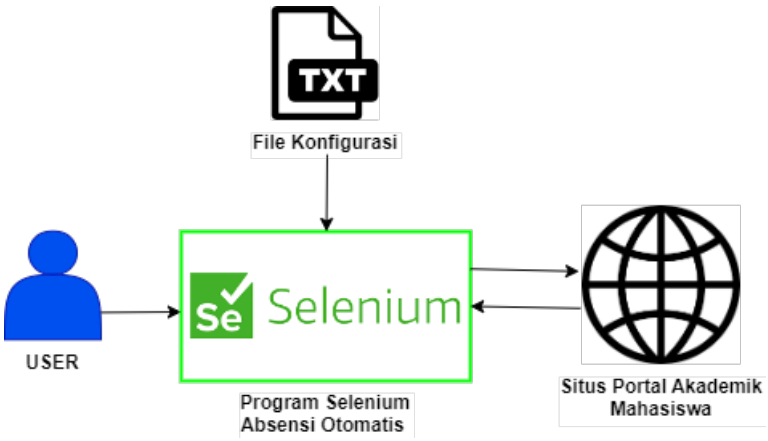
Otomatisasi perekaman kehadiran online ini akan menggunakan selenium, sehingga perlu diterjemahkan dari cara perekaman kehadiran online secara normal ke dalam selenium. Membuka situs web `https://studentportal.unpar.ac.id/` menggunakan selenium adalah dengan menggunakan *method get()*. Setiap tombol 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. Tombol yang ditekan secara otomatis menggunakan selenium perlu menggunakan *method click()*. Elemen yang ingin diambil dapat dilakukan dengan berbagai macam cara seperti yang sudah dijelaskan pada Bab 2.1. Beberapa faktor yang dapat dijadikan acuan untuk memilih cara dalam 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 2.1 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. Program akan berhenti ketika *browser* ditutup atau sudah berhasil melakukan perekaman kehadiran daring secara otomatis.

3.4.3 Arsitektur Sistem

Arsitektur sistem dibuat untuk menjelaskan gambaran umum dari sistem yang akan dibangun. Arsitektur sistem yang dibangun akan menampilkan sebuah program yang dijalankan pada komputer oleh *user*. Diagram arsitektur sistem yang dibangun tampak seperti pada gambar 3.34.

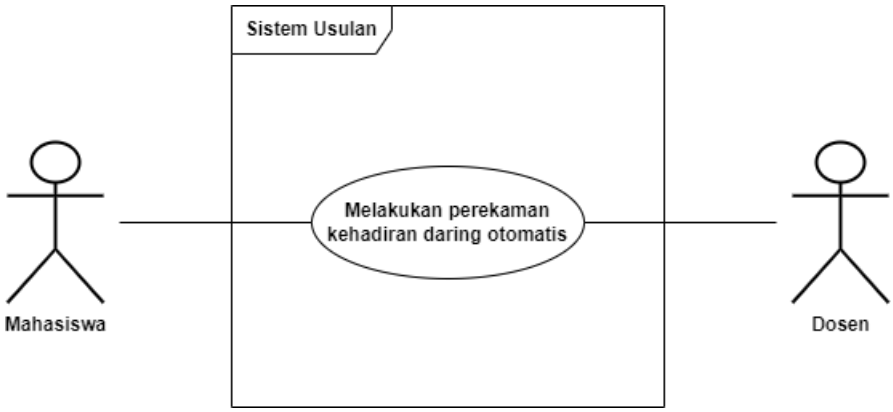


Gambar 3.34: Diagram Arsitektur

User cukup menjalankan program melalui *Command Prompt* pada komputer. Pada sistem ini, program menggunakan *framework* selenium yang berfungsi untuk melakukan otomatisasi pada *browser*. Sistem program ini memiliki sebuah masukan berupa *file* konfigurasi yang berisi perintah-perintah yang akan dijalankan oleh program. Program akan membaca dan menjalankan perintah dari *file* konfigurasi secara baris perbaris. Setiap baris yang dibaca dan dijalankan program akan diteruskan untuk mengotomatisasi *browser* dengan situs Portal Akademik Mahasiswa. Program akan terus membaca dan menjalankan setiap perintah pada *file* konfigurasi hingga akhirnya dapat melakukan perekaman kehadiran secara otomatis pada situs Portal Akademik Mahasiswa.

3.4.4 Pemodelan Diagram *Use Case*

Salah satu cara untuk mempermudah pembangunan program adalah membuat pemodelan dengan diagram *use case*. Diagram *use case* ini untuk mengetahui interaksi antara pengguna dengan programm. Pemodelan dengan diagram use case ditujukan untuk mempermudah pengembang perangkat lunak dalam memahami kebutuhan fungsional perangkat lunak. Gambar 3.35 merupakan diagram use case yang digunakan.



Gambar 3.35: Diagram *Use Case* Sistem Usulan

Setelah penggambaran diagram, diperlukan skenario *use case*. Skenario use case ini merupakan alur jalannya proses *use case* dari sisi aktor maupun sisi sistem. Berikut ini merupakan skenario use case yang disajikan dalam bentuk tabel berdasarkan diagram use case sistem usulan pada 3.35.

1. **Skenario mahasiswa melakukan perekaman kehadiran daring otomatis**.
   - Nama Use Case: Perekaman kehadiran daring otomatis.
   - Aktor: Mahasiswa
   - Deskripsi: Melakukan perekaman kehadiran daring otomatis.
   - Kondisi awal: -
   - Kondisi akhir: Program berhasil melakukan perekaman kehadiran secara otomatis.
   - Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Mahasiswa menjalankan program melalui *Command Prompt* dengan direktori *file*	Sistem membuka *browser* Google Chrome.
2		Sistem membuka situs Portal Akademik Mahasiswa dan melakukan *login*.
3		Sistem menuju halaman web perekaman kehadiran daring mahasiswa dan melakukan perekaman kehadiran daring.

2. **Skenario dosen melakukan perekaman kehadiran daring otomatis**.
   - Nama Use Case: Perekaman kehadiran daring otomatis.
   - Aktor: Dosen
   - Deskripsi: Melakukan perekaman kehadiran daring otomatis.
   - Kondisi awal: -
   - Kondisi akhir: Program berhasil melakukan perekaman kehadiran secara otomatis.
   - Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Dosen menjalankan program melalui *Command Prompt* dengan direktori *file*	Sistem membuka *browser* Google Chrome.
2		Sistem membuka situs AKUHADIR dan melakukan *login*.
3		Sistem menuju halaman web perekaman kehadiran daring dosen dan melakukan perekaman kehadiran daring.

BAB 4

PERANCANGAN

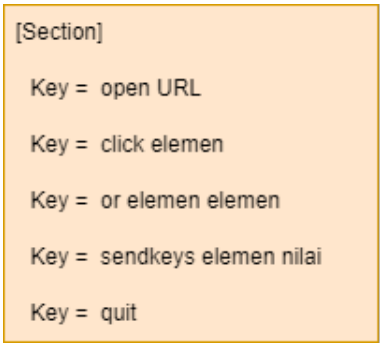
Pada bab ini akan dijelaskan perancangan program yang dibuat pada penelitian ini. Perancangan terdiri dari masukan program ,aktivitas sistem, dan perancangan algoritma.

4.1 Masukan Program

Program 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*.

4.1.1 Perancangan Masukan Program

Pada subbab ini akan dijelaskan perancangan dari *file* konfigurasi yang menjadi masukan untuk program. Gambar 4.1 merupakan rancangan untuk *file* konfigurasi.



Gambar 4.1: Gambar Rancangan Untuk Masukan Program

Penjelasan dari Gambar 4.1 sebagai berikut:

- *File* konfigurasi memiliki *section* dan nama *section* dapat diubah sesuai keinginan. Nama *section* ini yang akan dipanggil di dalam program, agar program dapat mengetahui seluruh isi *key* dan *value* dari *file* konfigurasi tersebut.
- *Key* akan berisi angka secara terurut dan dimulai dari angka satu hingga seterusnya. Diisi dengan angka secara terurut untuk memudahkan program dalam membaca langkap pertama hingga langkah terakhir.
- *Value* terdiri dari kata kunci *open*, *click*, *or*, *sendkeys*, dan *quit* yang berguna untuk program menjalankan perintah. *Value* terdiri juga dari alamat web atau URL, elemen, dan nilai.
- Kata *open* berpasangan dengan URL, karena kata *open* merupakan kata kunci dari *file* .ini bagi program untuk menjalankan fungsi *get()* dari selenium, yaitu membuka situs web yang ingin dituju.

- Kata *click* berpasangan dengan satu elemen, karena kata *click* merupakan kata kunci dari *file* .ini bagi program untuk menjalankan fungsi *click* dari selenium, yaitu menekan suatu tombol secara otomatis sesuai dengan elemen yang telah diambil pada situs web yang telah dibuka.
- Kata *sendkeys* berpasangan dengan satu elemen dan satu nilai, karena kata *sendkeys* merupakan kata kunci dari *file* .ini bagi program untuk menjalankan fungsi *sendkeys()* dari selenium, yaitu mengetik atau memasukan suatu nilai dalam bentuk teks maupun angka secara otomatis pada suatu elemen yang telah diambil.
- Kata *or* berpasangan dengan dua elemen, karena kata *or* merupakan kata kunci dari *file* .ini bagi program untuk menjalankan fungsi *click()* dari selenium. Perbedaan dengan kata *click* adalah bahwa kata *or* ini diberi dua elemen dari web yang kemungkinan dapat terjadi dua-duanya atau salah satu saja. Lalu menekan tombol secara otomatis pada satu elemen yang telah diambil atau menekan tombol secara otomatis dua elemen yang telah diambil secara bertahap.
- Kata *quit* merupakan kata kunci dari *file* .ini bagi program untuk menjalankan fungsi *quit()* dari selenium, yaitu menutup browser.

4.1.2 Konstruksi Masukan Program

Konstruksi untuk masukan program dapat disusun setelah mengetahui rancangan dari masukan program yang sudah dibahas pada subbab 4.1.1. Konstruksi untuk masukan program dibuat dengan mengikuti langkah-langkah dari perekaman kehadiran daring secara manual. Hal ini dilakukan agar program dapat menjalankan perintah sesuai dengan langkah-langkah dari perekaman kehadiran daring secara manual. Pada subbab 3.2.2 sudah dijelaskan alur untuk melakukan perekaman kehadiran daring mahasiswa secara manual dan tinggal diubah ke dalam file konfigurasi berdasarkan aturan dari rancangan dari masukan program, sehingga menghasil sebuah file konfigurasi esuai dengan alur perekaman kehadiran daring mahasiswa yang dapat dilihat pada Listing 4.1

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

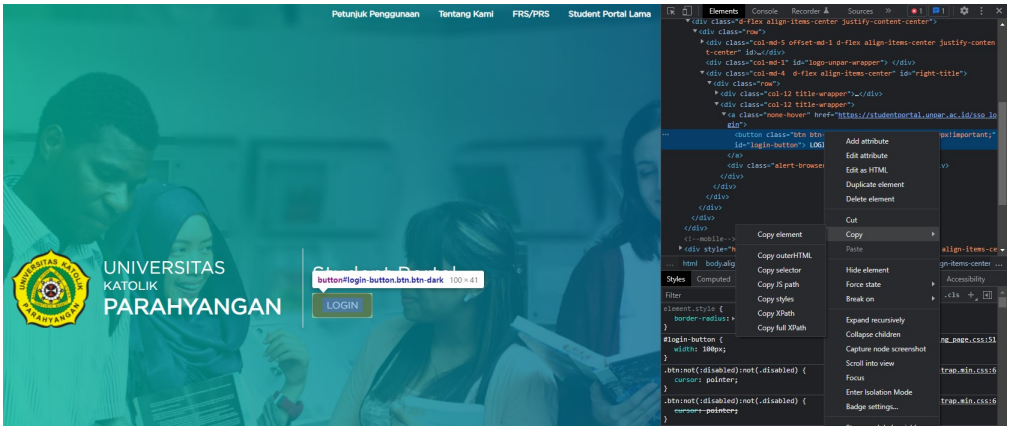
```
[database_config]
1 = open https://studentportal.unpar.ac.id
2 = click #login-button
3 = sendkeys #username 2017730035@student.unpar.ac.id
4 = click #next_login
5 = sendkeys #password 12345
6 = click #appPass>div.login__form>button
7 = or a[href='https://studentportal.unpar.ac.id/jadwal'] .swal-button.swal-button--confirm.swal-button--danger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

Berikut ini penjelasan dari isi pada file .ini yang merupakan masukan program perekaman kehadiran daring otomatis:

- 1 = open https://studentportal.unpar.ac.id, merupakan langkah pertama untuk melakukan absensi, yaitu membuka situs web Portal Akademik Mahasiswa dengan kata kunci *open* dan mengambil alamat situs web Portal Akademik Mahasiswa.
- 2 = click #login-button, langkah kedua dengan kata kunci *click* untuk menekan tombol *"LOGIN"* yang memiliki elemen (#login-button) berdasarkan *CSS selector* pada tombol *"LOGIN"*.
- 3 = sendkeys #username 2017730035@student.unpar.ac.id, langkah ketiga dengan dengan kata kunci *sendkeys* untuk memasukan sebuah nilai (2017730035@student.unpar.ac.id) pada elemen (#username) yang merupakan tempat untuk memasukan *email*.
- 4 = click #next_login, langkah keempat dengan kata kunci *click* untuk menekan tombol *"NEXT"* dan mengambil element (#next_login) pada tombol *"NEXT"*.
- 5 = sendkeys #password 12345, langkah kelima dengan kata kunci *sendkeys* untuk memasukan sebuah nilai (12345) pada elemen (#password) yang merupakan tempat untuk memasukan *password*.
- 6 = click #appPass>div.login___form>button, langkah keenam dengan kata kunci *click* untuk menekan tombol *"LOGIN"* yang memiliki elemen (#appPass>div.login___form>button) agar

dapat masuk ke dalam halaman utama situs Portal Akademik Mahasiswa.

- 7 = or a[href='https://studentportal.unpar.ac.id/jadwal'] .swal-button.swal-button–confirm.swal-button–danger, langkah ketujuh dengan kata kunci *or* untuk membuat program melihat kondisi jika notifikasi peringatan muncul maka menekan tombol "TUTUP" dengan elemen (.swal-button.swal-button–confirm.swal-button–danger) lalu menekan tombol untuk masuk ke bagian halaman absensi dengan elemen (a[href='https://studentportal.unpar.ac.id/jadwal']), jika notifikasi peringatan tidak muncul maka langsung menekan tombol untuk masuk ke bagian halaman absensi dengan elemen (a[href='https://studentportal.unpar.ac.id/jadwal']).
- 8 = click a[onclick="absenPerkuliahan(this)"], langkah kedelapan dengan kata kunci *click* untuk menekan tombol absensi yang memiliki elemen (a[onclick="absenPerkuliahan(this)"]).
- 9 = click .swal-button.swal-button–confirm.swal-button–danger9, langkah kesembilan dengan kata kunci *click* untuk menekan tombol *"OK"* yang memiliki elemen (.swal-button.swal-button–confirm.swal-button–danger9) dari notifikasi yang menunjukan absensi telah berhasil.
- 10 = quit, langkah terakhir dengan kata kunci *quit* untuk keluar dari browser.



Gambar 4.2: Tampilan Melakukan *Inspect Element*

Elemen yang dipakai dalam *file* .ini ini diambil dengan cara melakukan *inspect element* pada web yang ingin dilakukan otomatisasi. Elemen yang dipilih berdasarkan *CSS Selector* yang sudah dibahas pada subbab 3.4.2. Pada Gambar 4.2 adalah cara yang dilakukan untuk mendapat elemen yang ingin digunakan untuk otomatisasi. Untuk mendapatkan elemen tersebut, perlu melakukan klik kanan pada bagian elemen yang ingin diambil, lalu pilih "inspect". Setelah melakukan "inspect" maka akan muncul dokumen HTML yang dapat dilihat pada bagian kanan Gambar 4.2, sehingga dapat melakukan pengambilan elemen yang diperlukan.

4.2 Aktivitas Sistem

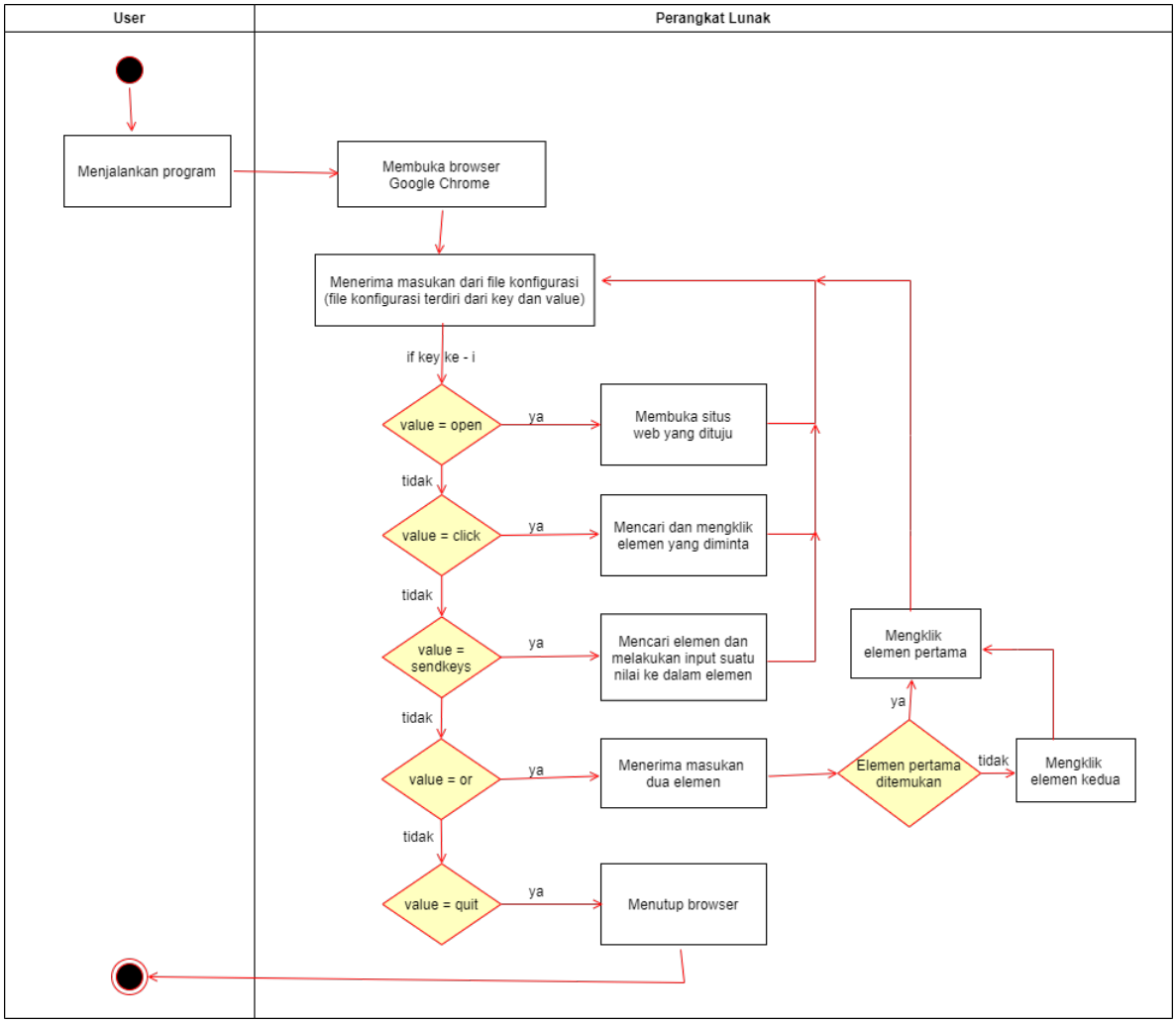
Program perekaman absen daring otomatis adalah program yang digunakan untuk melakukan absensi secara otomatis bagi mahasiswa UNPAR. Program ini menggunakan Selenium WebDriver sebagai *tools* yang berguna untuk melakukan otomatisasi pada browser web. Program ini juga membutuhkan masukan dari sebuah file konfigurasi untuk menjalankannya. Berikut ini adalah langkah-langkah yang harus dilakukan jika ingin menggunakan program ini:

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 sesuai versi dari browser Google Chrome, 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 program untuk melakukan perekaman absen daring otomatis.
5. Pengguna menyimpan hasil perubahan yang telah dilakukan pada file konfigurasi.

Diagram Aktivitas untuk program perekaman kehadiran daring otomatis dapat dilihat pada Gambar 4.3. Berikut ini adalah penjelasan pada diagram aktivitas:

1. Pengguna menjalankan langsung programnya.
2. Program akan membuka browser Google Chrome saat pertama kali program dijalankan.
3. Program menerima masukan dari *file* konfigurasi yang telah di-*setup* oleh pengguna.
4. Program akan menjalankan perintah sesuai masukan *file* konfigurasi secara baris perbaris dimulai dari baris pertama.
5. Setiap program selesai menjalankan satu baris perintah dari masukan *file* konfigurasi maka program akan kembali melakukan cek baris berikutnya pada *file* konfigurasi dan menjalankannya hingga baris terakhir.
6. Jika masukan dari value *file* konfigurasi terdapat kata *open* maka program diberi perintah untuk membuka situs web yang dituju pada browser Google Chrome secara otomatis.
7. Jika masukan dari value *file* konfigurasi terdapat kata *click* maka program diberi perintah untuk menekan suatu tombol yang diminta berdasarkan elemen yang telah diambil.
8. Jika masukan dari value *file* konfigurasi terdapat kata *sendkeys* maka program diberi perintah untuk melakukan *input* suatu nilai berupa angka atau kata ke dalam suatu elemen yang telah dipilih.
9. Jika masukan dari value *file* konfigurasi terdapat kata *or* maka program akan menerima dua elemen secara langsung, jika elemen pertama telah ditemukan maka program langsung diberi perintah untuk menekan tombol berdasarkan elemen pertama, jika elemen kedua yang ditemukan terlebih dahulu maka program diberi perintah untuk menekan tombol berdasarkan elemen kedua terlebih dahulu lalu menekan tombol berdasarkan elemen pertama.
10. Jika masukan dari value *file* konfigurasi terdapat kata *quit* maka program diberi perintah untuk menutup menutup browser dan program selesai dijalankan.



Gambar 4.3: Diagram Aktivitas Program Absen Daring Otomatis

Pada diagram aktivitas program (Gambar 4.3) hanya satu langkah saja yang dilakukan oleh pengguna untuk melakukan perekaman kehadiran daring, sedangkan pada diagram aktivitas absensi daring manual (Gambar 3.13) yang terletak pada subbab 3.2.2 cukup banyak langkah yang perlu dilakukan untuk dapat melakukan perekaman kehadirang daring. Pengguna setidaknya perlu melakukan 10 langkah untuk dapat melakukan perekaman kehadiran daring secara manual. Program perekaman kehadiran daring otomatis ini mempersingkat langkah-langkah yang harus dilakukan pengguna untuk melakukan perekaman kehadiran daring dengan cukup menjalankan programnya.

4.3 Perancangan Algoritma

Perancangan algoritma ini memiliki *input* berupa sebuah *file* konfigurasi dan *output* yang dihasilkan adalah program berhasil melakukan perekaman kehadiran daring. Algoritma ini akan melakukan perekaman kehadiran daring otomatis sesuai dari perintah masukan dari *file* konfigurasi.

Algorithm 1 Algoritma untuk Perekaman Kehadiran Daring Otomatis

Input: file_konfigurasi
Output: Perekaman daring secara otomatis pada Google Chrome

```
osPathEnvironment ← getCurrentDirectory()
driver ← getChromeDriver()                                  ▷ Membuka browser
parser ← ConfigParser()
parser.read(file_konfigurasi)
index ← 1
while TRUE do
    (command, parameters) ← parser.get(file_konfigurasi, str(index)).split()
    if command = "open" then
        driver.get(parameters[0])                           ▷ Membuka situs yang dituju
        index ← index + 1
    else if command = "click" then
        elemen ← findElementByCssSELECTOR(parameters[0])
        if elemenIsDisplayed() AND elemenIsEnabled() then
            elemen.click()
        else
            driver.quit()
            print (Absensi Gagal)
            index ← index + 1
        end if
    else if command = "sendkeys" then
        inpt ← findElementByCssSELECTOR(parameters[0])
        inpt.send_keys(parameters[1])
        index ← index + 1
    else if command = "or" then
        elemen1 ← findElementByCssSELECTOR(parameters[0])
        elemen2 ← findElementByCssSELECTOR(parameters[1])
        if elemenIsDisplayed() AND elemenIsEnabled() then
            elemen2.click()
            elemen1.click()
        else
            elemen1.click()
            index ← index + 1
        end if
    else if command = "quit" then
        driver.quit()                                       ▷ Menutup browser
        print (Absensi Berhasil)
    end if
end while
```

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

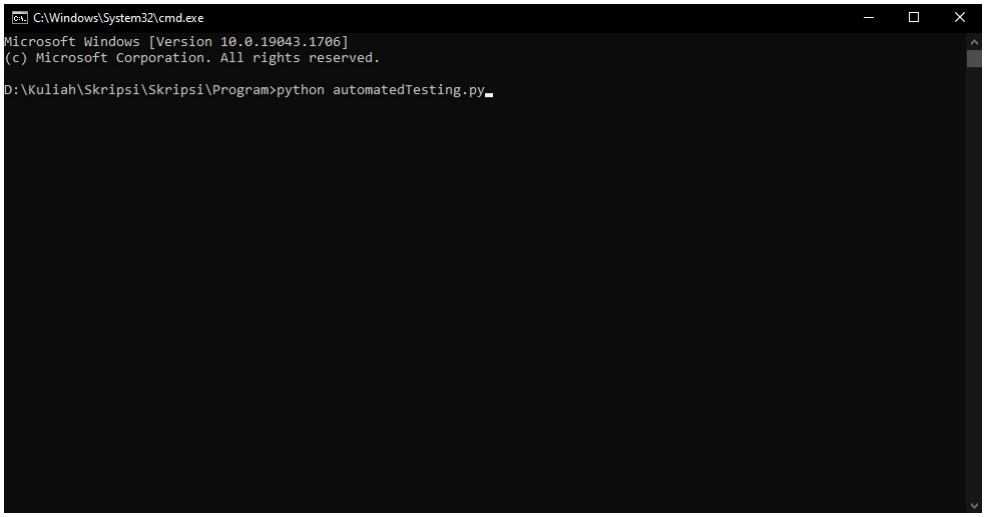
5.1.2 Hasil Implementasi

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 masukkan untuk perangkat lunak. *File* .ini dibahas pada Subbab 4.1. 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--danger9
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*

Setelah pengguna menekan "Enter" pada *Command Prompt* maka perangkat lunak akan melakukan perekaman kehadiran daring secara otomatis, bagi mahasiswa maka perangkat lunak akan melakukan perekaman kehadiran daring pada Portal Akademik Mahasiswa secara otomatis, dimana perangkat lunak akan menjalankan secara otomatis tahap-tahap perekaman kehadiran daring secara manual yang biasa dilakukan mahasiswa seperti yang dibahas pada Subbab 3.2.2, sedangkan bagi dosen maka perangkat lunak akan melakukan perekaman kehadiran daring pada AKUHADIR seperti yang dibahas pada Subbab 3.2.3. Setelah berhasil melakukan perekaman kehadiran daring maka akan muncul pemberitahuan pada *Command Prompt* dengan tulisan "Information: Absensi Berhasil!" bahwa perekaman berhasil dilakukan. Pemberitahuan absensi gagal pada *Command Prompt* akan ada tulisan "Warning! Absensi Gagal, Elemen tidak ditemukan: (*diisi dengan sesuai elemen yang tidak ditemukan*)". Absensi gagal terjadi karena tidak ada jadwal kuliah lagi bagi mahasiswa, atau sudah melakukan absensi sehingga tidak ada yang bisa lagi untuk melakukan perekaman kehadiran.

5.2 Pengujian

5.2.1 Pengujian Fungsional Mahasiswa

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

Mahasiswa melakukan perekaman kehadiran otomatis di Portal Akademik Mahasiswa. Mahasiswa perlu penyesuaian *file* `database.ini` seperti dilihat pada kode 5.2.

Kode 5.2: *File* `database.ini` Portal Akademik Mahasiswa (*password* disembunyikan)

```
[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 xxx
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
```

Setelah menjalankan program, didapatkan bahwa program dapat berjalan dan berfungsi dengan baik untuk dapat melakukan perekaman kehadiran daring otomatis pada situs Portal Akademik Mahasiswa. Kesimpulan dari pengujian fungsional mahasiswa pada program dinyatakan berhasil, dan selengkapnya dapat dilihat pada tabel 5.1.

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

5.2.2 Pengujian Fungsional Dosen

Subbab ini ditulis berdasarkan hasil wawancara terhadap dosen pembimbing ketika mencoba program perekaman kehadiran daring otomatis.

Program yang sudah dibuat diujikan juga di komputer dosen pembimbing, untuk melakukan rekam kehadiran pada portal AKUHADIR (subbab 3.2.3). Berikut adalah spesifikasi komputer dosen pembimbing yang digunakan untuk menguji:

1. *Processor*: Apple M1
2. *Random Access Memory* (RAM): 8 GB
3. Sistem Operasi: macOS Monterey
4. Versi Python : Python 3.9.10

Dosen pembimbing melakukan perekaman kehadiran otomatis di AKUHADIR. Dosen pembimbing perlu penyesuaian *file* `database.ini` seperti dilihat pada kode 5.3.

Kode 5.3: *File* `database.ini` AKUHADIR (*password* disembunyikan)

```
[database_config]
1 = open https://akuhadir.unpar.ac.id
2 = sendkeys #username pascal@unpar.ac.id
3 = click #next_login
4 = sendkeys #password xxx
5 = click button[name=submit]
6 = click a[href='https://akuhadir.unpar.ac.id/absensi?tab=tab2']
7 = click a[onclick='checkin_home()']
8 = quit
```

Setelah dosen pembimbing menjalankan program, didapatkan bahwa program dapat berjalan dan berfungsi dengan baik untuk dapat melakukan perekaman kehadiran daring otomatis pada situs AKUHADIR. Kesimpulan dari pengujian fungsional program pada komputer dosen pembimbing dinyatakan berhasil, dan selengkapnya dapat dilihat pada tabel 5.2.

Tabel 5.2: Tabel Pengujian Fungsional

No.	Aksi Pengguna	Reaksi yang diharapkan	Reaksi Perangkat Lunak
1.	Dosen menjalankan perangkat lunak	Browser Google Chrome terbuka	Sesuai
		Browser menuju situs AKUHADIR	Sesuai
		Browser menuju halaman web untuk perekaman kehadiran	Sesuai
		Melakukan perekaman kehadiran daring secara otomatis	Sesuai

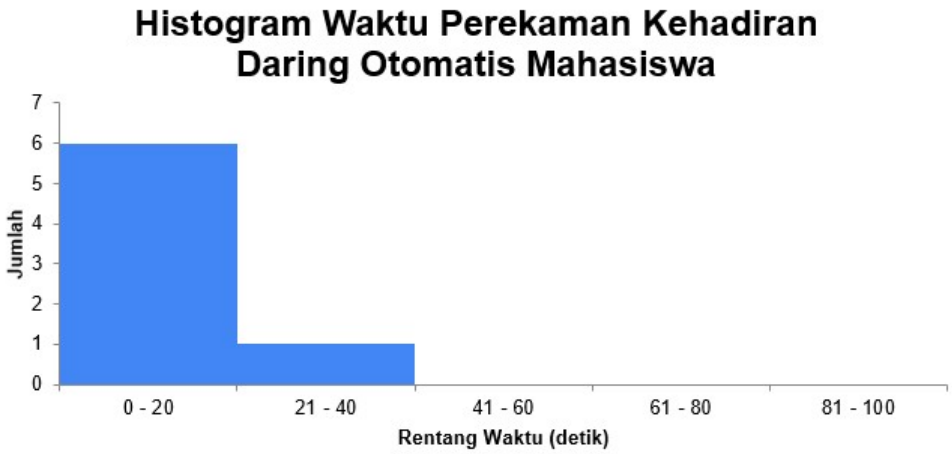
5.2.3 Pengujian Eksperimental

Pengujian eksperimental dilakukan terhadap beberapa mahasiswa dan dosen 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 dan dosen diminta untuk mengisi Google Form untuk mengetahui kelancaran perangkat lunak ketika dijalankan dan mengetahui lama waktu yang dibutuhkan hingga program berhasil melakukan perekaman kehadiran.

Hasil Survei Mahasiswa

Dari 7 responden yang telah mengisi survei, memberi respons bahwa perangkat lunak berjalan dengan baik dan dapat melakukan perekaman kehadiran daring secara otomatis. Pertanyaan yang diajukan kepada responen dan rangkuman jawaban hasil survei sebagai berikut:

1. **Apakah perangkat lunak berjalan dengan baik (tidak ada crash atau error) dan dapat melakukan perekaman kehadiran daring secara otomatis?**
   Berdasarkan jawaban dari semua responden menyatakan setuju bahwa program tidak mengalami *error* atau *crash*.
2. **Setelah menjalankan perangkat lunak perekaman kehadiran daring otomatis. Berapa lama(detik) waktu yang dibutuhkan untuk melakukan perekaman kehadiran daring menggunakan program perekaman kehadiran daring otomatis?**
   Histogram ini dikelompokan berdasarkan rentang waktu per 20 detik. Histogram menunjukan bahwa sebanyak 6 mahasiswa memiliki rentang waktu 0 sampai 20 detik dan sebanyak 1 mahasiswa memiliki rentang waktu 21 sampai 40 detik.



Gambar 5.2: Histogram Waktu Perekaman Kehadiran Daring Otomatis Mahasiswa

Hasil survei perekaman kehadiran daring otomatis untuk setiap mahasiswa secara jelas dapat dilihat pada tabel 5.3 menunjukan waktu yang didapatkan dari 7 responden dalam menjalankan perangkat lunak untuk melakukan perekaman kehadiran daring otomatis. Hasil tabel tersebut menunjukan bahwa dalam melakukan perekaman kehadiran daring otomatis memiliki rata-rata waktu adalah 16,71 detik.

Tabel 5.3: Tabel Perekaman Kehadiran Daring Otomatis Mahasiswa

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

3. **Apakah setuju dengan perangkat lunak perekaman kehadiran daring otomatis ini, membuat waktu interaksi dengan situs web/browser untuk melakukan absensi menjadi lebih singkat?**
   Berdasarkan jawaban dari semua 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.

Hasil Survei Dosen

Hasil survei kepada dosen hanya mendapat 2 respon dosen. Pertanyaan yang diajukan kepada responen dan rangkuman jawaban hasil survei sebagai berikut:

1. **Apakah perangkat lunak berjalan dengan baik (tidak ada crash atau error) dan dapat melakukan perekaman kehadiran daring secara otomatis?**
   Berdasarkan jawaban dari semua responden menyatakan setuju bahwa program tidak mengalami *error* atau *crash*.
2. **Setelah menjalankan perangkat lunak perekaman kehadiran daring otomatis. Berapa lama(detik) waktu yang dibutuhkan untuk melakukan perekaman kehadiran daring menggunakan program perekaman kehadiran daring otomatis?**
   Dua dosen yang menjadi responden menyatakan waktu yang dibutuhkan untuk melakukan perekaman kehadiran daring secara otomatis adalah 2 detik dan 15 detik. Rata-rata perekaman kehadiran daring otomatis yang dilakukan oleh dua dosen tersebut adalah 8,5 detik.
3. **Apakah setuju dengan perangkat lunak perekaman kehadiran daring otomatis ini, membuat waktu interaksi dengan situs web/browser untuk melakukan absensi menjadi lebih singkat?**
   Berdasarkan jawaban dari semua 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.

Perbandingan Hasil Pengujian

Bagian ini akan membandingkan waktu yang didapatkan perekaman kehadiran daring secara otomatis menggunakan program dengan waktu perekaman kehadiran daring dan luring bagi mahasiswa maupun dosen yang didapatkan dari survei pada subbab 3.4.1. Tabel 5.4 merupakan tabel hasil perbandingan rata-rata waktu untuk melakukan perekaman kehadiran.

Tabel 5.4: Tabel Perekaman kehadiran

Pengguna	Rata-rata waktu dengan program	Rata-rata waktu luring	Rata-rata waktu daring
Mahasiswa	16,71 detik (7 respon)	7,95 detik (21 respon)	63 detik (21 respon)
Dosen	8,5 detik (2 respon)	24,33 detik (6 respon)	31,83 detik (6 respon)

Dari hasil perbandingan pada tabel 5.4 secara angka dapat dilihat bahwa untuk mahasiswa waktu perekaman kehadiran dengan program masih lebih lama sedikit dengan waktu perekaman luring tetapi lebih cepat dibanding waktu perekaman daring dan untuk dosen perekaman kehadiran dengan perangkat lunak lebih cepat dibandingkan waktu perekaman daring maupun luring.

BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

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

1. Analisis kebutuhan untuk program perekaman kehadiran daring otomatis dengan melakukan survei terhadap beberapa mahasiswa serta dosen terhadap perekaman kehadiran secara daring.
2. Model untuk program perekaman kehadiran daring otomatis dengan menggunakan *Command Prommpt* untuk menjalankan program.
3. Telah berhasil menggunakan *framework* selenium untuk mengimplementasikan fungsi dari program perekaman kehadiran daring otomatis. Program dapat melakukan perekaman kehadiran daring secara otomatis terhadap Portal Akademik Mahasiswa maupun AKUHADIR dengan *framework* selenium.
4. Telah berhasil membuat program yang mampu secara otomatis melakukan tahap-tahap dalam melakukan perekaman kehadiran secara daring dengan sekali menjalankan perangkat lunak, sehingga waktu interaksi dengan situs webnya menjadi lebih singkat.
5. Telah berhasil membangun program perekaman kehadiran daring secara otomatis menggunakan Selenium WebDriver.

6.2 Saran

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

1. Melakukan survei lebih banyak lagi untuk perekaman kehadiran daring otomatis agar hasil perbandingannya lebih akurat lagi.
2. Mempertimbangkan program perekaman kehadiran daring otomatis untuk dosen agar dapat merekam kehadiran mahasiswa, karena terdapat fitur pada AKUHADIR agar dosen dapat melakukan perekaman kehadiran untuk mahasiswa.

DAFTAR REFERENSI

[1] 9f08b37 (2021) *Selenium*. Software Freedom Conservancy.

[2] Version 3.1 (2017) *XML path language (XPath) 3.1*. World Wide Web Consortium.

[3] 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.

[4] Version 3.10.4 (2022) *Configuration file parser*. Python Software Foundation.

[5] 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.

[6] Situmorang, M. (2020) Pemberlakuan sementara kebijakan bekerja dari rumah (wfh). Surat Edaran No. III/R/2020-07/1153.

LAMPIRAN A

FILE MASUKAN UNTUK PERANGKAT LUNAK

A.1 *File* Konfigurasi Mahasiswa

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

Kode A.1: database.ini (*password disembunyikan*)

```
[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--danger9
10 = quit
```

A.2 *File* Konfigurasi Dosen

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

Kode A.2: database.ini (*password disembunyikan*)

```
[database_config]
1 = open https://akuhadir.unpar.ac.id
2 = sendkeys #username 2017730035@student.unpar.ac.id
3 = click #next_login
4 = sendkeys #password 12345
5 = click button[name=submit]
6 = click a[href='https://akuhadir.unpar.ac.id/absensi?tab=tab2']
7 = click a[onclick='checkin_home()']
8 = quit
```

LAMPIRAN B

KODE PROGRAM PERANGKAT LUNAK PEREKAMAN KEHADIRAN DARING OTOMATIS

Kode B.1: automatedTesting.py

```
# -*- coding: utf-8 -*-
"""
Created on Sun Mar 13 11:35:40 2022

@author: user
"""
from configparser import ConfigParser
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait
from selenium.common.exceptions import TimeoutException
import os

os.environ["PATH"] = os.getcwd()
print(os.environ["PATH"])

driver = webdriver.Chrome()
parser = ConfigParser()
parser.read('database.ini')

i = 1
while (True):
    x = parser.get('database_config',str(i)).split()
    if  x[0] == "open" :
        driver.get(x[1])
        i += 1
    elif x[0] == "click":
        try:
            elemen = WebDriverWait(driver,5).until(EC.element_to_be_clickable((By.CSS_SELECTOR, x[1])))
            if elemen.is_displayed() and elemen.is_enabled() :
                elemen.click()
        except TimeoutException:
                driver.quit()
                a = "Absensi Gagal, Elemen tidak ditemukan:", x[1]
                print ("Warning!", a)
                break
        i+=1
    elif x[0] == "sendkeys":
        inpt =  WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.CSS_SELECTOR,x[1])))
        inpt.send_keys(x[2])
        i += 1
    elif x[0] =="or":
        try:
            elemen1 = driver.find_element(By.CSS_SELECTOR, x[1]) #jadwal
            elemen2 = WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.CSS_SELECTOR,x[2]))) #notif
            if elemen2.is_displayed() and elemen2.is_enabled() :
                elemen2.click()
                elemen1.click()
        except TimeoutException:
                elemen1.click()
        i +=1
    elif x[0] == "quit":
        driver.wait(5)
        driver.quit()
        print ("Information: Absensi Berhasil!")
        break
```

LAMPIRAN C

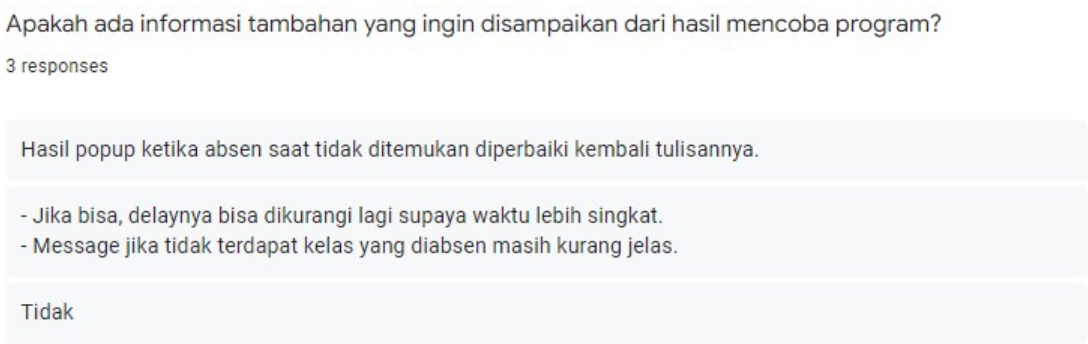
HASIL PENGUJIAN EKSPERIMENTAL

C.1 Hasil Pengujian Eksperimental Mahasiswa

Gambar C.1: Jawaban responden(Mahasiswa) untuk pertanyaan pertama.

Gambar C.2: Jawaban responden(Mahasiswa) untuk pertanyaan kedua.

Gambar C.3: Jawaban responden(Mahasiswa) untuk pertanyaan ketiga.



Gambar C.4: Jawaban responden(Mahasiswa) untuk pertanyaan keempat.

C.2 Hasil Pengujian Eksperimental Dosen

Gambar C.5: Jawaban responden(Dosen) untuk pertanyaan pertama.

Setelah menjalankan perangkat lunak perekaman kehadiran daring otomatis. Berapa lama(detik) waktu yang dibutuhkan untuk melakukan perekaman kehadiran daring menggunakan program perekaman kehadiran daring otomatis?

2 responses

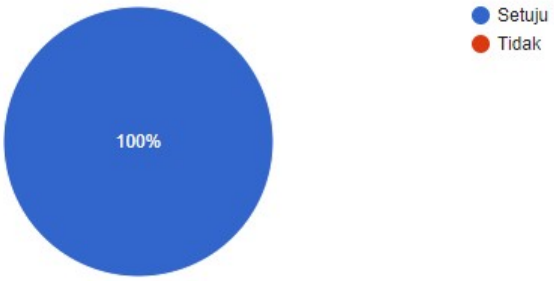
2

15

Gambar C.6: Jawaban responden(Dosen) untuk pertanyaan kedua.

Apakah setuju dengan perangkat lunak perekaman kehadiran daring otomatis ini, membuat waktu interaksi dengan situs web/browser untuk melakukan absensi menjadi lebih singkat?

2 responses



Gambar C.7: Jawaban responden(Dosen) untuk pertanyaan ketiga.

Apakah ada informasi tambahan yang ingin disampaikan dari hasil mencoba program?

2 responses

Begitu browser terbuka, halamannya berganti" dengan cepat. Jadi agak panik tadi, apa yg terjadi sama akuhadirnya. :D Tapi untungnya tidak ada error dan berhasil terabsen.

Tidak ada.

Gambar C.8: Jawaban responden(Dosen) untuk pertanyaan keempat.