BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Pelaksanaan kerja magang di Astra Credit Companies pada divisi *Data And Automation* dibimbing oleh Ibu Mutiara Caesagusta Yosadhie selaku *Techno Academy Analyst* yang berperan dalam mengawasi, membimbing, memberikan informasi selama kerja magang. Selama pengembangan *Robotic Process Automation* (RPA), Bapak Kevin Antariksa berperan sebagai *supervisor* tim RPA.

3.2 Tugas yang Dilakukan

Cakupan kerja magang yang diberikan adalah sebagai berikut:

1. Mengembangkan *dashboard* untuk aplikasi Monitoring Formulir Pemesanan Technocenter.
2. Mengikuti *Basic Training* dan *Detailed Training*.
3. Mempresentasikan hasil kerja.
4. Mengembangkan RPA (*Robotic process automation*) untuk *project Restart Server SOA*.
5. Mengembangkan RPA untuk *project* Asuransi AAB.
6. Mengikuti rapat mingguan dan rapat dengan pengguna.
7. Mengisi pekerjaan harian di situs *Monday* Astra Credit Companies.

3.3 Uraian Pelaksanaan Magang

Tabel 3.1. Uraian Pelaksanaan Magang

Minggu Ke -	Pekerjaan yang dilakukan
1-2	Sambutan dari tim ACC, pengenalan aplikasi *Monday*, menginstall software-software yang akan digunakan, dan mengikuti *Basic Training* aplikasi-aplikasi yang digunakan di ACC
3-5	Mengikuti *Basic Training*, Mengikuti *test* untuk penentuan divisi, Mengikuti *Detailed Training* untuk divisi *Data And Automation*.
6-10	Mengembangkan aplikasi Technocenter Formulir Order Monitoring dengan tim, Mengikuti *Training Soft Skills*, dan Mempersiapkan materi presentasi mengenai aplikasi Technocenter Formulir Order Monitoring yang dikembangkan.
11	Melakukan latihan presentasi dengan tim, Mempresentasikan hasil aplikasi yang dikembangkan kepada tim ACC, Mengikuti rapat tentang pembagian tim *Real Project*.
12-13	- *Meeting* dengan tim *RPA Oracle*. - Pengerjaan RPA untuk *Restart Server SOA*. - *Update progress* ke pengguna. - *testing* RPA yang dikembangkan.
14-15	- *Meeting* mingguan dengan tim *RPA Oracle*. - Mempelajari video proses *Restart Server SOA*. - *Update progress* ke pengguna. - Melakukan pengubahan pada *flow* RPA berdasarkan video yang ditinjau. - Mengimplementasikan bot telegram pada RPA. *testing* RPA yang dikembangkan.

16-17	- *Meeting* mingguan dengan tim *RPA Oracle*. - *Update progress* ke pengguna. - Memindahkan *file* RPA ke server. - Melakukan pengubahan pada *flow* RPA dikarenakan terjadi perubahan pada website. - Menambahkan fitur *try catch* dan *summary* pada RPA yang dikembangkan. - *testing* RPA yang dikembangkan.
18-19	- *Update progress* ke pengguna. - *Meeting* mingguan dengan tim *RPA Oracle*. - Memulai pengembangan *log bot* untuk RPA *Restart Server SOA* - Mempersiapkan ppt untuk presentasi *Progress*. - Mengoptimalkan RPA *Restart Server SOA* - Mendokumentasikan RPA yang dikembangkan - *testing* RPA yang dikembangkan.
20-21	- *Meeting* mingguan dengan tim *RPA Oracle*. - Mengubah pesan pada *error* dan *summary* yang terkirim ke telegram - Mempersiapkan ppt untuk presentasi *Progress*. - Mempresentasikan *progress project* ke tim ACC - Mempelajari alur kerja Asuransi AAB - Mengembangkan RPA untuk Asuransi AAB - Melanjutkan mengoptimalkan RPA *Restart Server SOA*
22	- *Meeting* mingguan dengan tim *RPA Oracle*. - Melanjutkan pengembangan RPA Asuransi AAB - Melanjutkan dokumentasi RPA yang dikembangkan - Mempresentasikan *progress project* ke tim dosen pembimbing kampus merdeka. - Mengikuti acara *Closing Bootcamp Batch 3*

3.4 Persiapan

Berikut merupakan *software* dan *hardware* yang digunakan selama proses kerja magang berlangsung:

1. *Hardware* - TUF FX505GE

(a) *Operation System*: *Windows* 10 *Home Single Language*, 64-bit

(b) *Processor*: Intel(R) Core(TM) i7-8750H CPU @2.20GHz (12CPUs)

(c) *Random Access Memory* (RAM): 16GB

(d) *Graphic Card*: NVIDIA GeForce GTX 1050 Ti 4GB

2. *Software*

(a) Microsoft Power BI

(b) UIPath

(c) Postman

(d) Visual Studio Code

(e) Google Chrome

3.5 *Training*

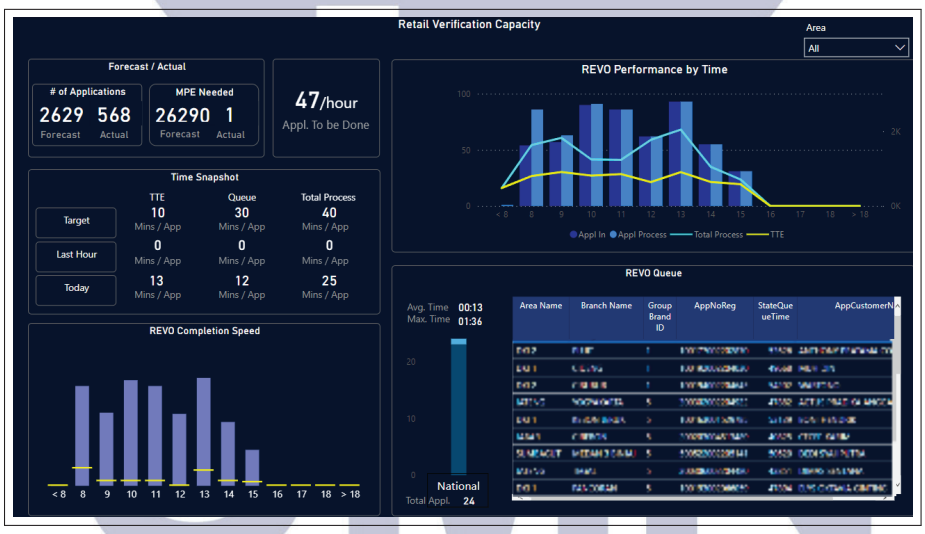
Pelaksanaan *training* di ACC dibagi menjadi dua, yaitu *Basic Training* dan *Detailed Training*. Pada *Basic Training*, diajarkan fungsi-fungsi dasar dari aplikasi yang digunakan di ACC. Setelah *Basic Training*, dilakukan tes serta memilih divisi yang diinginkan. Kemudian setelah memilih divisi, dilakukan *Detailed Training* untuk latihan penggunaan aplikasi yang digunakan di divisi *Data and Automation*.

Pada *Detailed Training*, diajarkan penggunaan aplikasi Power BI, UIPath, dan Informatica Big Data Developer. Selama masa pelatihan, hal yang dilakukan yaitu mempelajari teori penggunaan aplikasi, kemudian melakukan praktek penggunaan aplikasi.

3.5.1 Microsoft Power BI

Microsoft Power BI digunakan untuk menampilkan data yang telah diolah sehingga mudah dipahami oleh pengguna. Pada tahap pelatihan, pelajaran yang didapatkan addalah menampilkan data menggunakan visualisas pada aplikasi Power BI dan penggunaan DAX Query untuk mengolah data. Tugas yang diberikan pada masa pelatihan yaitu pengembangan *dashboard* seperti pada gambar 3.1, dan hasil *dashboard* yang dikembangkan pada Gambar 3.2

Gambar 3.1. Contoh *dashboard*
Sumber: Dokumen Internal Perusahaan

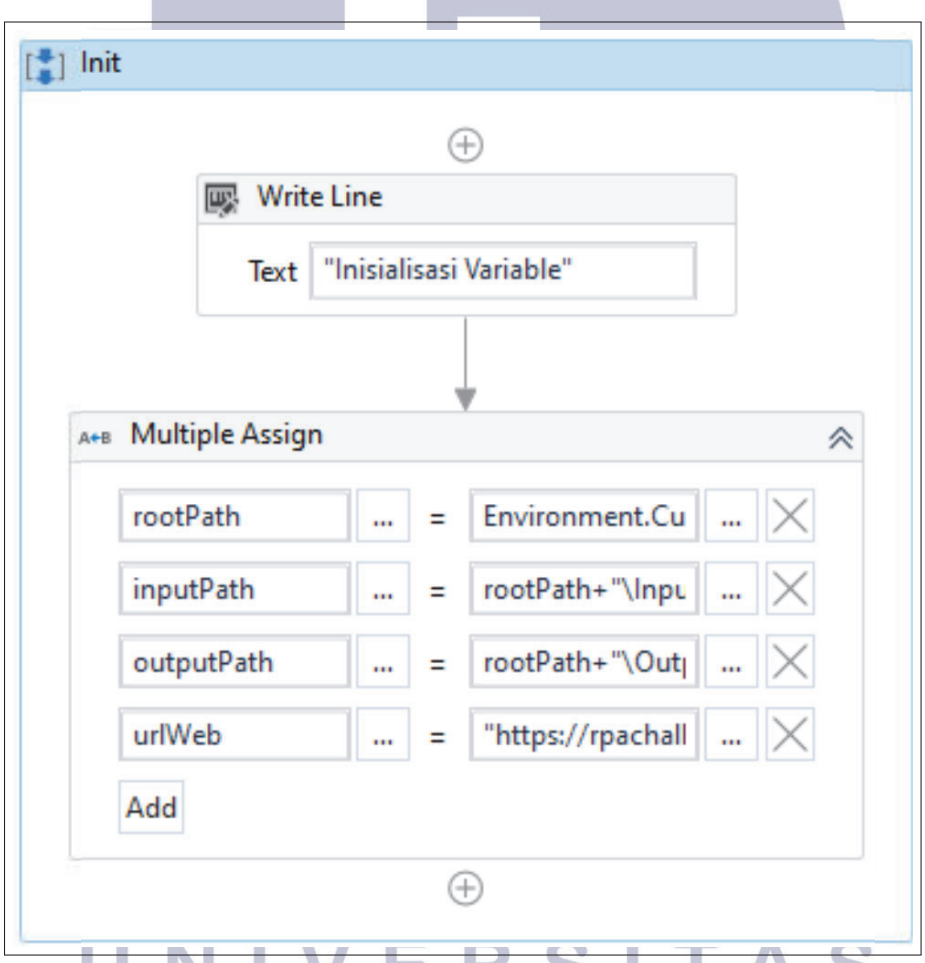


Gambar 3.2. Hasil *dashboard* yang dikembangkan

Dalam pengembangan *dashboard* pada Gambar 3.2, terdapat dua tambahan visualisasi yang didapatkan dari *AppSource* Power BI, yaitu *Advance Card* dan *Bullet Chart by OKVIZ*. Angka yang ditampilkan pada *dashboard* menggunakan *card* dan untuk menampilkan bagian *"Appl. To Be Done"* menggunakan *advance card*. Bar chart pada bagian *"Revo Completion Speed"* ditampilkan menggunakan *bullet chart* untuk menambahkan bagian *target* berwarna kuning, bar chart *line and clustered column chart* digunakan untuk menampilkan bagian *"Revo Performance By Time"*, dan bar chart *stacked column chart* digunakan pada bagian *"National"*.

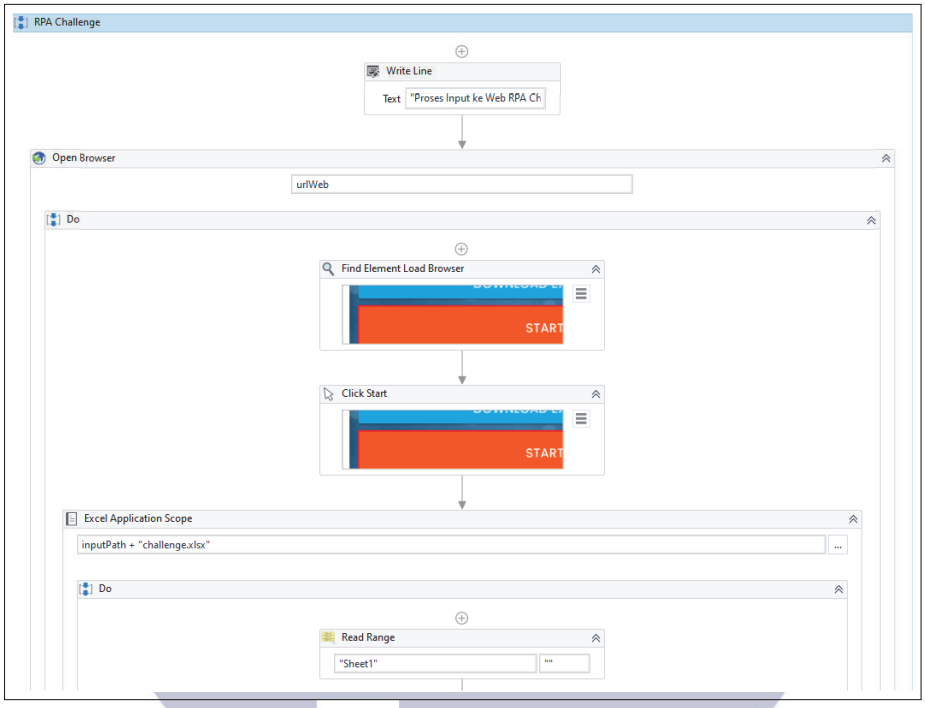
3.5.2 UIPath

UIPath digunakan untuk mengembangkan *Robotic Process Automation* (RPA) untuk proses yang dilakukan secara berulang-ulang di perusahaan. Pada tahap pelatihan, hal yang diajarkan berfokus pada penggunaan *activity* yang ada di aplikasi UIPath. Tugas yang diberikan pada masa pelatihan yaitu mengembangkan RPA untuk *website* yang dinamis. Hasil pengembangan dapat dilihat pada Gambar 3.3, Gambar 3.4, Gambar 3.6, dan Gambar 3.7.



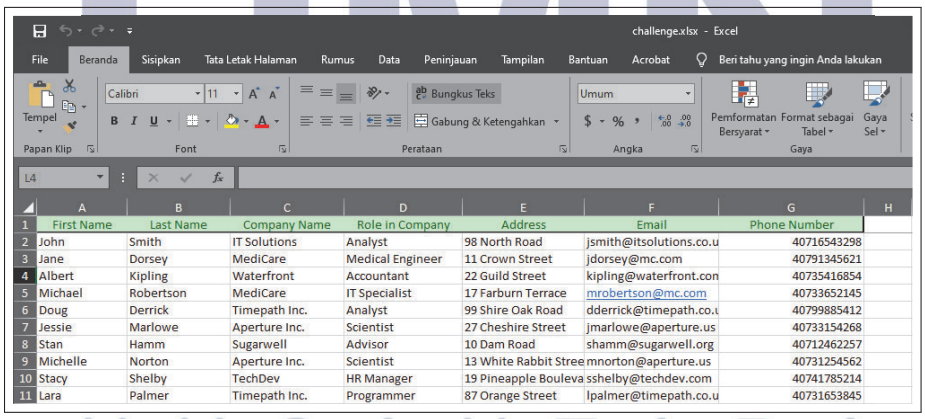
Gambar 3.3. Inisialisasi variabel

Hal pertama yang dilakukan adalah melakukan inisialisasi variabel yang diperlukan seperti *rootPath* yang berisi *path folder* RPA, *inputPath* dan *outputPath* berisi *path folder* input dan *output*, dan *urlWeb* berisi *link website*.



Gambar 3.4. Hasil tugas UIPath 1

Pada Gambar 3.4, hal pertama yang dilakukan adalah membuka *browser* Google Chrome dan membuka *link* variabel *urlWeb*, kemudian menunggu halaman *website* terbuka sepenuhnya lalu meng-klik tombol *start*. Setelah itu masuk ke proses membaca data pada *file* excel bernama challenge.xlsx. Isi data pada excel dapat dilihat pada Gambar 3.5

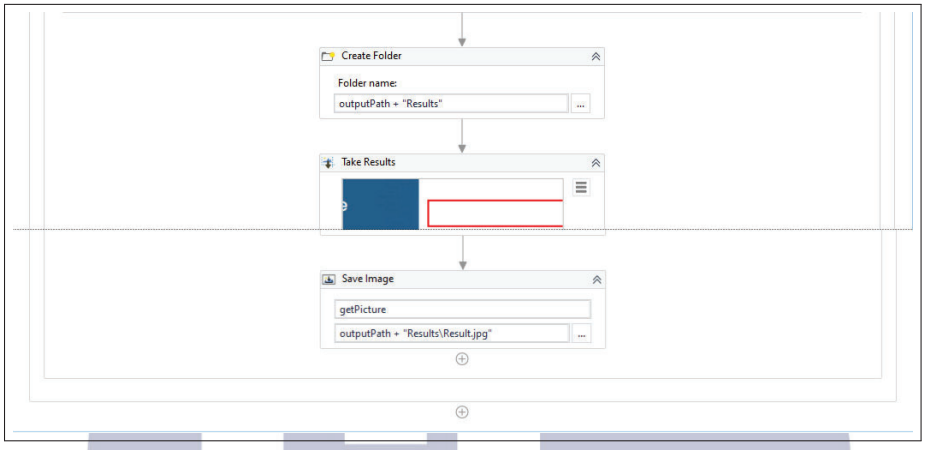


	A	B	C	D	E	F	G	H
1	First Name	Last Name	Company Name	Role in Company	Address	Email	Phone Number	
2	John	Smith	IT Solutions	Analyst	98 North Road	jsmith@itsolutions.co.u	40716543298	
3	Jane	Dorsey	MediCare	Medical Engineer	11 Crown Street	jdorsey@mc.com	40791345621	
4	Albert	Kipling	Waterfront	Accountant	22 Guild Street	kipling@waterfront.con	40735416854	
5	Michael	Robertson	MediCare	IT Specialist	17 Farburn Terrace	mrobertson@mc.com	40733652145	
6	Doug	Derrick	Timepath Inc.	Analyst	99 Shire Oak Road	dderrick@timepath.co.u	40799885412	
7	Jessie	Marlowe	Aperture Inc.	Scientist	27 Cheshire Street	jmarlowe@aperture.us	40733154268	
8	Stan	Hamm	Sugarwell	Advisor	10 Dam Road	shamm@sugarwell.org	40712462257	
9	Michelle	Norton	Aperture Inc.	Scientist	13 White Rabbit Stree	mnorton@aperture.us	40731254562	
10	Stacy	Shelby	TechDev	HR Manager	19 Pineapple Bouleva	sshelby@techdev.com	40741785214	
11	Lara	Palmer	Timepath Inc.	Programmer	87 Orange Street	lpalmer@timepath.co.u	40731653845	

Gambar 3.5. Isi *file* excel

Gambar 3.6. Hasil tugas UIPath 2

Pada Gambar 3.6, adalah proses melakukan pengulangan untuk pengisian data-data pada *website*. Pengisian data menggunakan *activity* bernama *anchor* sehingga pengisian data akan secara tepat pada *field* yang ditentukan kemudian mengklik tombol *submit*. Proses pengisian data dilakukan sebanyak 10 kali.



Gambar 3.7. Hasil tugas UIPath 3

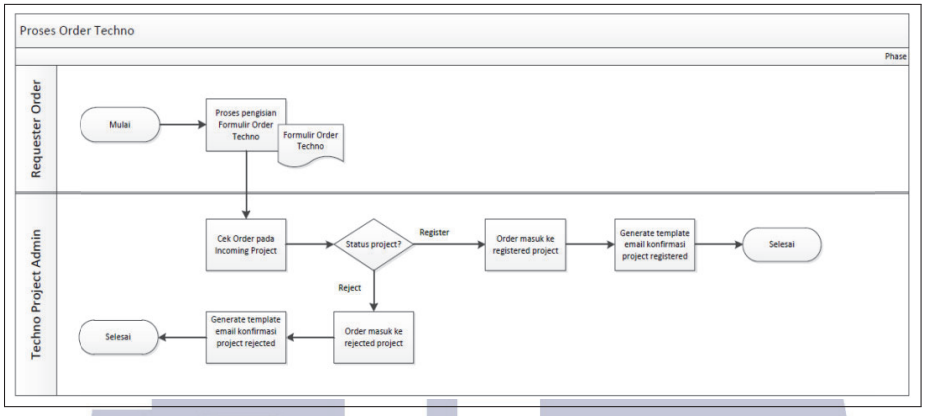
Gambar 3.7 merupakan proses penyimpanan rekor kecepatan RPA menjalankan proses pengisian data. Folder bernama *results* akan dibuat untuk menampung foto rekor kecepatan, kemudian rekor kecepatan disimpan menggunakan *activity* bernama *Take Screenshot*. Hasil *screenshot* disimpan pada folder bernama *results* dengan nama *results.jpg* (disimpan dalam bentuk jpg)

3.6 Aplikasi Technocenter Formulir Order Monitoring

Apikasi Monitoring Formulir Pemesanan Technocenter merupakan aplikasi untuk melakukan order/request project ke Technocenter. Saat ini, proses order ke Technocenter masih dilakukan secara manual dengan menggunakan aplikasi Monday. Dikarenakan adanya keterbatasan fitur pada aplikasi Monday, maka aplikasi ini dikembangkan dengan harapan dapat memaksimalkan kebutuhan proses *order/request* project ke Technocenter.

3.6.1 Flowchart Aplikasi

Alur kerja utama untuk aplikasi Technocenter Formulir Order Monitoring dapat dilihat pada gambar 3.8

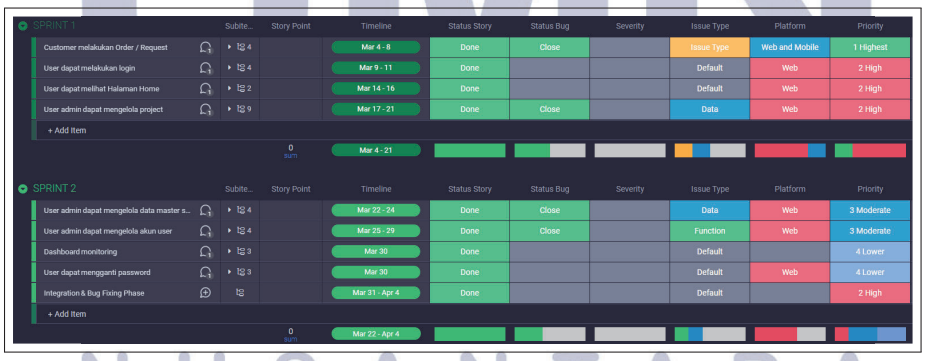


Gambar 3.8. Flowchart proses pemesanan technocenter
Sumber: Dokumen Internal Perusahaan

Proses dimulai dari pengguna yang ingin memesan dengan cara mengisi formulir pemesanan. Kemudian *admin* dapat melakukan pengecekan pesanan yang masuk pada bagian *incoming project* dan *admin* dapat memilih untuk menerima atau menolak pesanan yang dikirim pengguna. Setelah *admin* memilih, *admin* dapat mengirimkan *email* kepada pengguna untuk memberitahukan jika pesanan yang mereka kirim diterima atau ditolak.

3.6.2 Waktu Pengerjaan

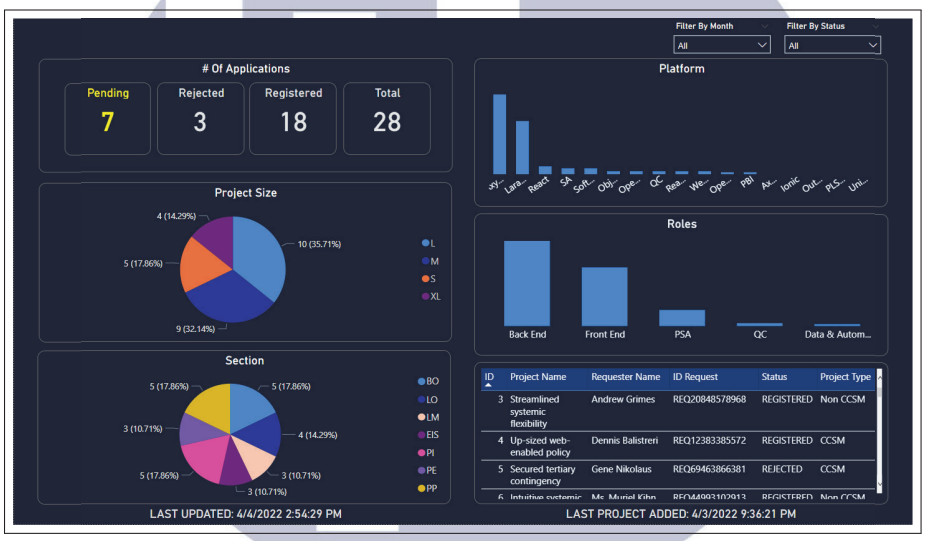
Aplikasi ini mulai dikembangkan pada tanggal 1 Maret 2022 dan selesai dikembangkan pada tanggal 4 April. Pengembangan aplikasi ini dibagi menjadi dua *sprint* yang dapat dilihat pada gambar 3.9



Gambar 3.9. *Sprint*

3.6.3 *Dashboard Monitoring*

Dashboard monitoring untuk aplikasi Technocenter Formulir Order Monitoring dikembangkan dengan menggunakan aplikasi *Microsoft Power BI*. Hasil akhir *dashboard monitoring* dapat dilihat pada gambar 3.10.



Gambar 3.10. Tampilan *dashboard monitoring*

Bagian *filtering* terletak di bagian kanan atas *dashboard*. Terdapat 2 jenis *filtering* yang dapat dilakukan pada *dashboard* ini. *Filter by Month* digunakan untuk mem-*filter* data berdasarkan bulan dan tahun data dimasukkan. *Filter by Status* digunakan untuk mem-*filter* data berdasarkan statusnya.

Bagian *# of applications* menampilkan banyaknya pemesanan yang terdaftar di sistem. Banyaknya pemesanan di bagi menjadi 4 bagian yaitu *project* yang masih belum di proses, *project* yang ditolak, *project* yang diterima, dan jumlah seluruh *project* yang terdaftar di sistem.

Bagian *project size* dan *section* menggunakan *pie chart* untuk menampilkan data. *Project size* menampilkan persentase serta jumlah *project* berdasarkan ukuran dari project. Ukuran dari project dibagi menjadi 4 yaitu S, M, L, atau XL. *Section* menampilkan persentase serta jumlah *project* berdasarkan bagiannya.

Bagian *platform* dan *roles* menggunakan *bar chart* untuk menampilkan data. *Platform* menampilkan perbandingan *software* yang paling banyak digunakan di seluruh project. *Roles* menampilkan perbandingan divisi yang paling banyak digunakan di seluruh project.

Bagian tabel dapat digunakan untuk melihat rincian project, dan dapat digu-

nakan untuk melakukan *filtering*. Pada bagian bawah *dashboard* terdapat *Last Updated* yang memberikan informasi kapan data pada *dashboard* diperbarui dan *Last Project Added* yang memberikan informasi mengenai kapan penambahan *project* terbaru.

A. Data pada *dashboard*

Data yang terdapat pada bagian *dashboard* masih menggunakan data *dummy* dan data diperbarui secara manual dengan mengeksekusi kode *python* untuk mengambil data menggunakan *API (Application Programming Interface)*.

```
1   import requests
2   import pandas as pd
```

Gambar 3.11. *Import libraries*

Langkah pertama yang dilakukan untuk pengambilan data adalah ada *import libraries* yang dibutuhkan seperti pada gambar 3.11

```
url = "http://xyz/restv2/accepted/login"
header= {"AuthToken" : "c3596c77-0171-409a-9c2e-6cfe5fd2f9c9"}
body = {
    "loginData": {
        "npk": "10",
        "password": "1"
    }
}

responseData = requests.post(url,headers=header,json=body)

data = responseData.json()["OUT_DATA"]
```

Gambar 3.12. *Login* untuk mendapatkan token autentikasi

Gambar 3.12 adalah langkah melakukan *request API* untuk mendapatkan token autentikasi yang disimpan pada variabel bernama ”data” dan akan digunakan ketika pengambilan data.

```
url = "http://xyz/restv2/accepted/getAllResourceNeed"
header= {"AuthToken" : data[0]["AuthToken"]}
body = {
    "status":"ALL"
}

responseData = requests.post(url,headers=header,data=body)
ResourceNeed = pd.DataFrame(responseData.json()["OUT_DATA"])


url = "http://xyz/restv2/accepted/getDataProject"
responseData = requests.post(url,headers=header,data=body)

DataProject = pd.DataFrame(responseData.json()["OUT_DATA"])
```

Gambar 3.13. Potongan kode untuk pengambilan data

Pada gambar 3.13, kode autentikasi yang didapatkan pada langkah sebelumnya akan dimasukkan ke variabel ”header”. variabel ”responseData” digunakan untuk menyimpan *response* dari *request API*. Variabel ”responseData” kemudian diolah menggunakan fungsi *pd.DataFrame* agar hasilnya berupa tabel yang kemudian dapat di *import* ke *Microsoft Power BI*.

B. DAX Query

DAX Query adalah bahasa yang digunakan di Power BI untuk memproses data.

```
A.1 Total Project =
VAR a = COUNT(DataProject[idProject])
return IF(ISBLANK(a),0,a)
```

Gambar 3.14. Potongan kode DAX Query 1

```
A.4 Rejected Project =
var a = COUNTX(FILTER(DataProject,DataProject[status]="REJECTED"),DataProject[idProject])
return IF(ISBLANK(a),0,a)
A.2 Pending Project =
var a = COUNTX(FILTER(DataProject,DataProject[status]="PENDING"),DataProject[idProject])
return IF(ISBLANK(a),0,a)
A.3 Registered Project =
var a = COUNTX(FILTER(DataProject,DataProject[status]="REGISTERED"),DataProject[idProject])
return IF(ISBLANK(a),0,a)
```

Gambar 3.15. Potongan kode DAX Query 2

Gambar 3.14 digunakan untuk mendapatkan jumlah banyak project yang terdaftar di tabel DataProject dengan cara menghitung kolom idProject pada DataProject, dan Gambar 3.15 jumlah banyak project berdasarkan status. Hasil didapatkan dengan cara melakukan *filter* status kemudian menghitung idProject hasil dari *filter*.

```
B.2 LO =
VAR a = COUNTX(FILTER(DataProject,DataProject[section]="LO"),DataProject[idProject])
return IF(ISBLANK(a),0,a)
B.5 PI =
VAR a = COUNTX(FILTER(DataProject,DataProject[section]="PI"),DataProject[idProject])
return IF(ISBLANK(a),0,a)
B.6 PE =
VAR a = COUNTX(FILTER(DataProject,DataProject[section]="PE"),DataProject[idProject])
return IF(ISBLANK(a),0,a)
B.7 PP =
VAR a = COUNTX(FILTER(DataProject,DataProject[section]="PP"),DataProject[idProject])
return IF(ISBLANK(a),0,a)
B.1 BO =
VAR a = COUNTX(FILTER(DataProject,DataProject[section]="BO"),DataProject[idProject])
return IF(ISBLANK(a),0,a)
B.4 EIS =
VAR a = COUNTX(FILTER(DataProject,DataProject[section]="EIS"),DataProject[idProject])
return IF(ISBLANK(a),0,a)
B.3 LM =
VAR a = COUNTX(FILTER(DataProject,DataProject[section]="LM"),DataProject[idProject])
return IF(ISBLANK(a),0,a)
```

Gambar 3.16. Potongan kode DAX Query 3

Gambar 3.16 digunakan untuk mendapatkan jumlah project berdasarkan section yang terdaftar pada tabel DataProject. Hasil didapatkan dengan cara melakukan *filter section* kemudian melakukan perhitungan banyaknya data dengan cara menghitung idProject hasil *filter*.

```
Last Project Added = "LAST PROJECT ADDED: " & MAX(DataProject[dateAdded])
```

Gambar 3.17. Potongan kode DAX Query 4

Gambar 3.17 digunakan untuk mendapatkan tanggal project terbaru yang masuk ke dalam tabel DataProject. Hasil didapatkan dengan menggunakan fungsi *MAX* pada kolom dateAdded di tabel DataProject.

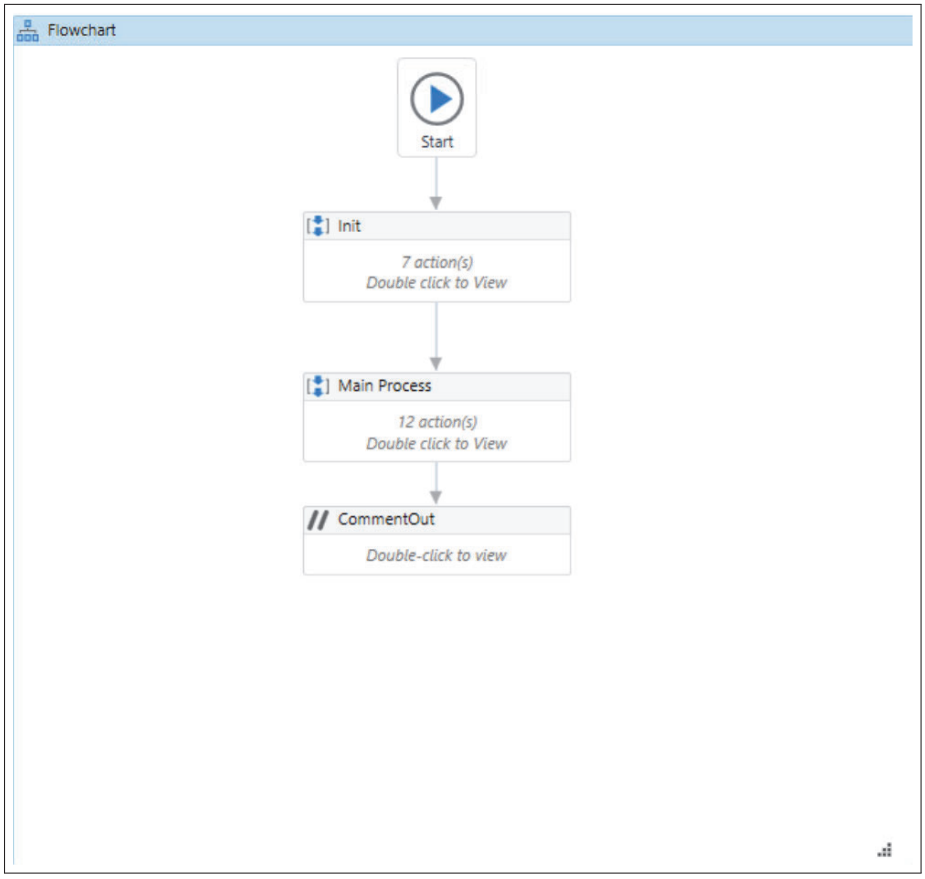
```
Resource =
var a = SUM('ResourceNeed'[quantity])
return IF(ISBLANK(a),0,a)
```

Gambar 3.18. Potongan kode DAX Query 5

Gambar 3.18 digunakan untuk menghitung total jumlah kolom *quantity* pada tabel ResourceNeed. Hasil perhitungan akan di *filter* langsung melalui pengaturan *bar chart*.

3.7 RPA *Restart Server-Oriented Architecture* (SOA)

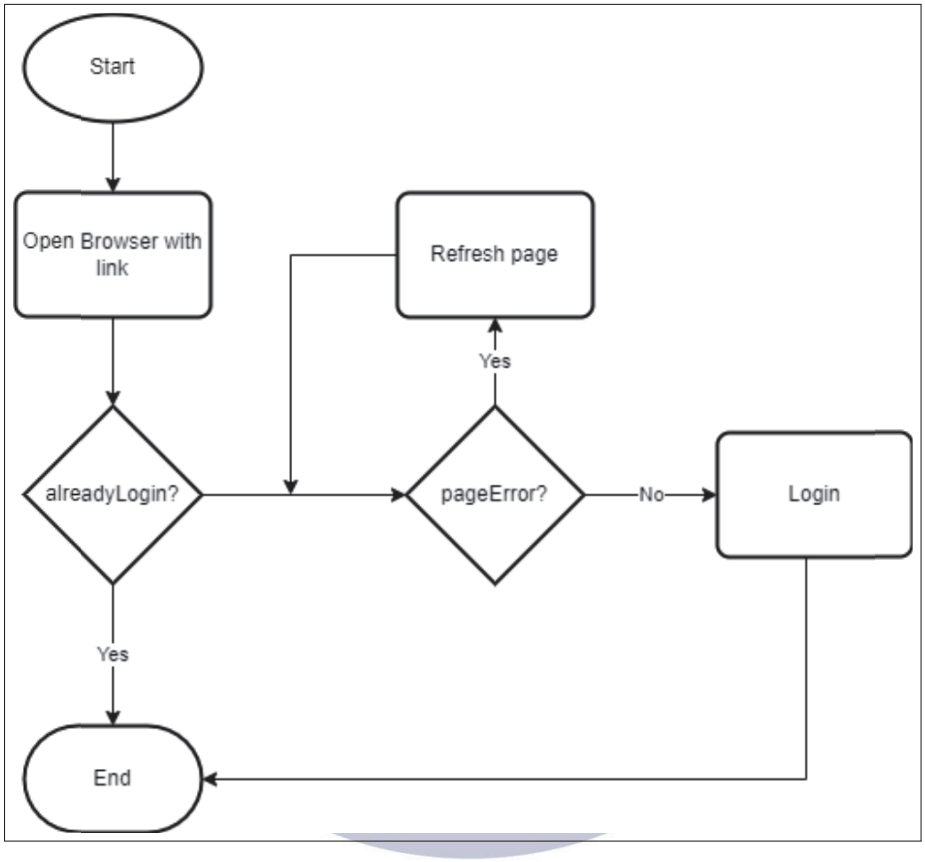
RPA ini dikembangkan untuk mengurangi *human error* ketika melakukan *restart SOA*. RPA yang dikembangkan masih menggunakan *file* .bat *dummy*. Hasil pengembangan RPA dapat dilihat pada Gambar 3.19.



Gambar 3.19. Hasil pengembangan RPA

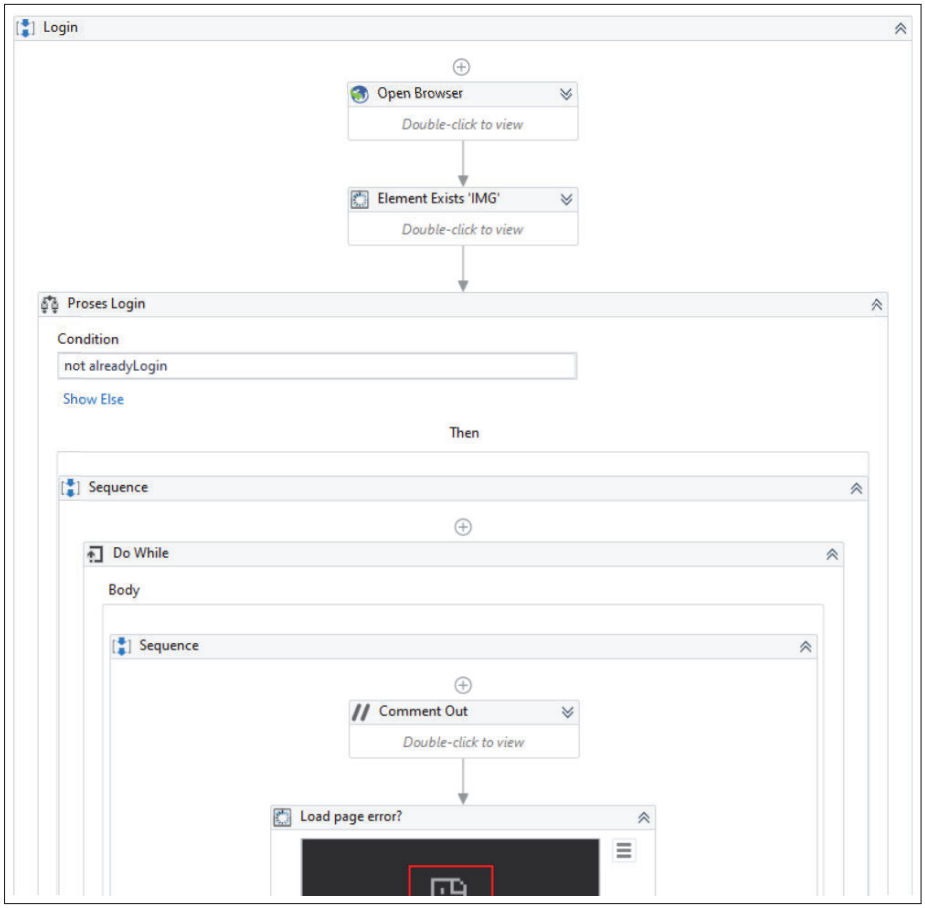
3.7.1 *Login*

alur proses *login* untuk *website SOA* dapat dilihat pada Gambar 3.20 dan hasil implementasi dapat dilihat pada Gambar 3.21 dan Gambar 3.22.

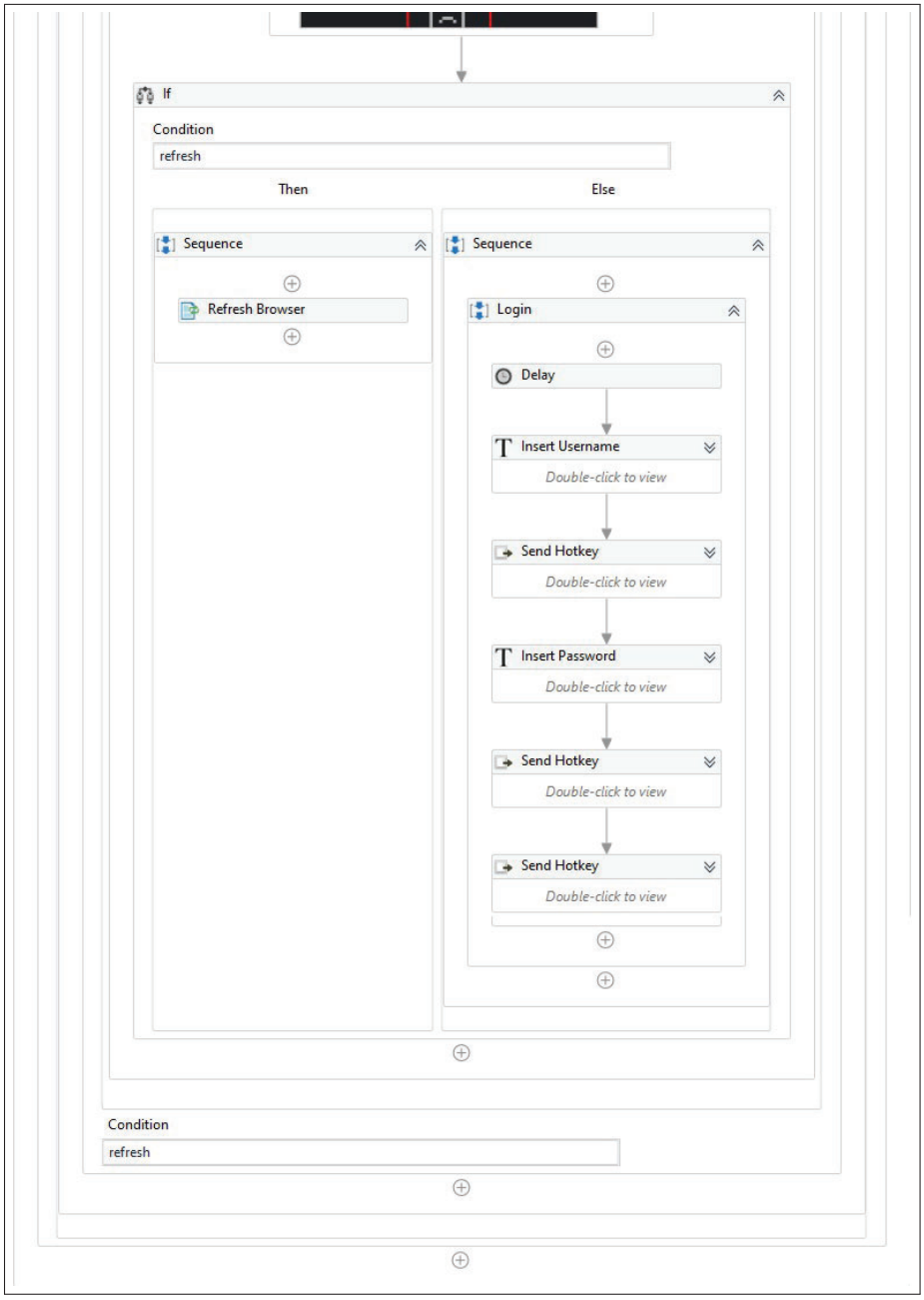


Gambar 3.20. Flowchart proses login

Proses dimulai dengan membuka *website login*, kemudian melakukan pengecekan jika pengguna sudah *login*. Jika pengguna belum melakukan *login*, maka akan melakukan pengecekan apakah halaman *web* berhasil dibuka, apabila tidak berhasil maka akan melakukan proses *refresh* ulang halaman *web*. Setelah halaman *web* berhasil dibuka, proses login akan dilakukan.



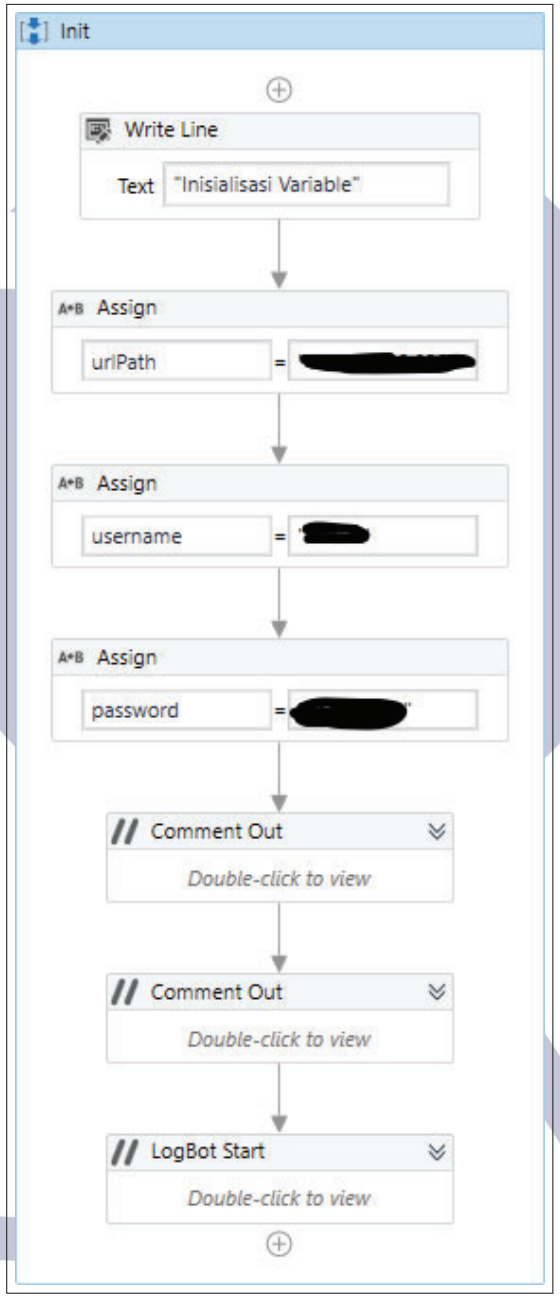
Gambar 3.21. *login* 1



Gambar 3.22. *login* 2

3.7.2 Init

Proses *init* merupakan proses inisialisasi variabel-variabel yang nantinya akan digunakan di bagian *main process*. Rincian proses *init* dapat dilihat pada Gambar 3.23

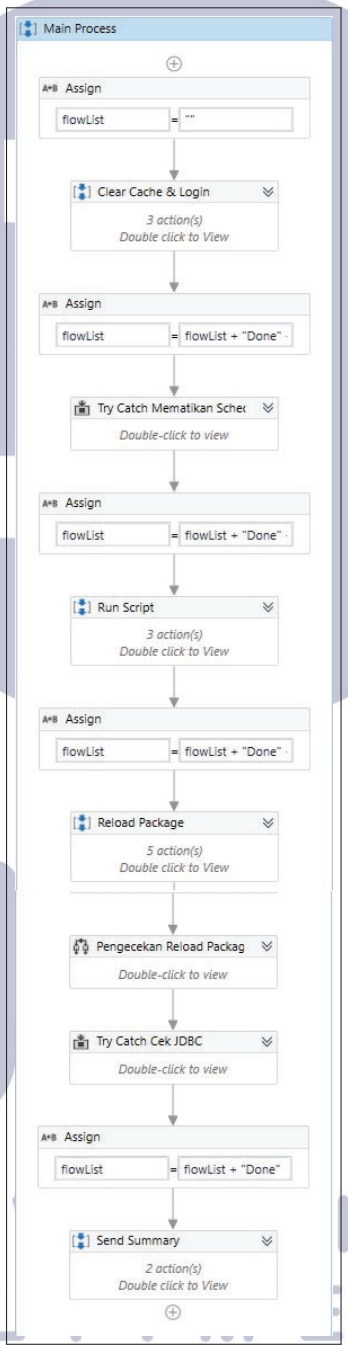


Gambar 3.23. Rincian proses inisialisasi

3.7.3 Main Process

Main process berisi langkah melakukan *restart SOA*. *Restart SOA* dimulai dengan melakukan proses *clear cache login*, mematikan *scheduler*, menjalankan *script* untuk proses *restart server*, melakukan proses *reload package*, cek *Java Database Connectivity* (JDBC), dan memberikan ringkasan proses yang telah di-

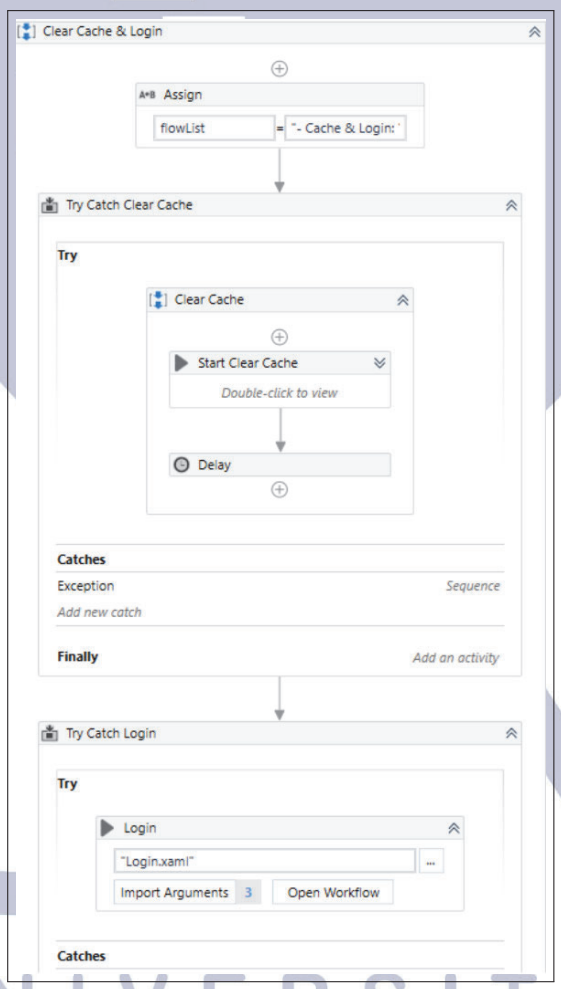
lakukan. Pada setiap langkah akan dilakukan pendataan proses melalui variabel bernama flowList. Rincian *main process* dapat dilihat pada Gambar 3.24.



Gambar 3.24. Rincian *main process*

A. Clear Cache & Login

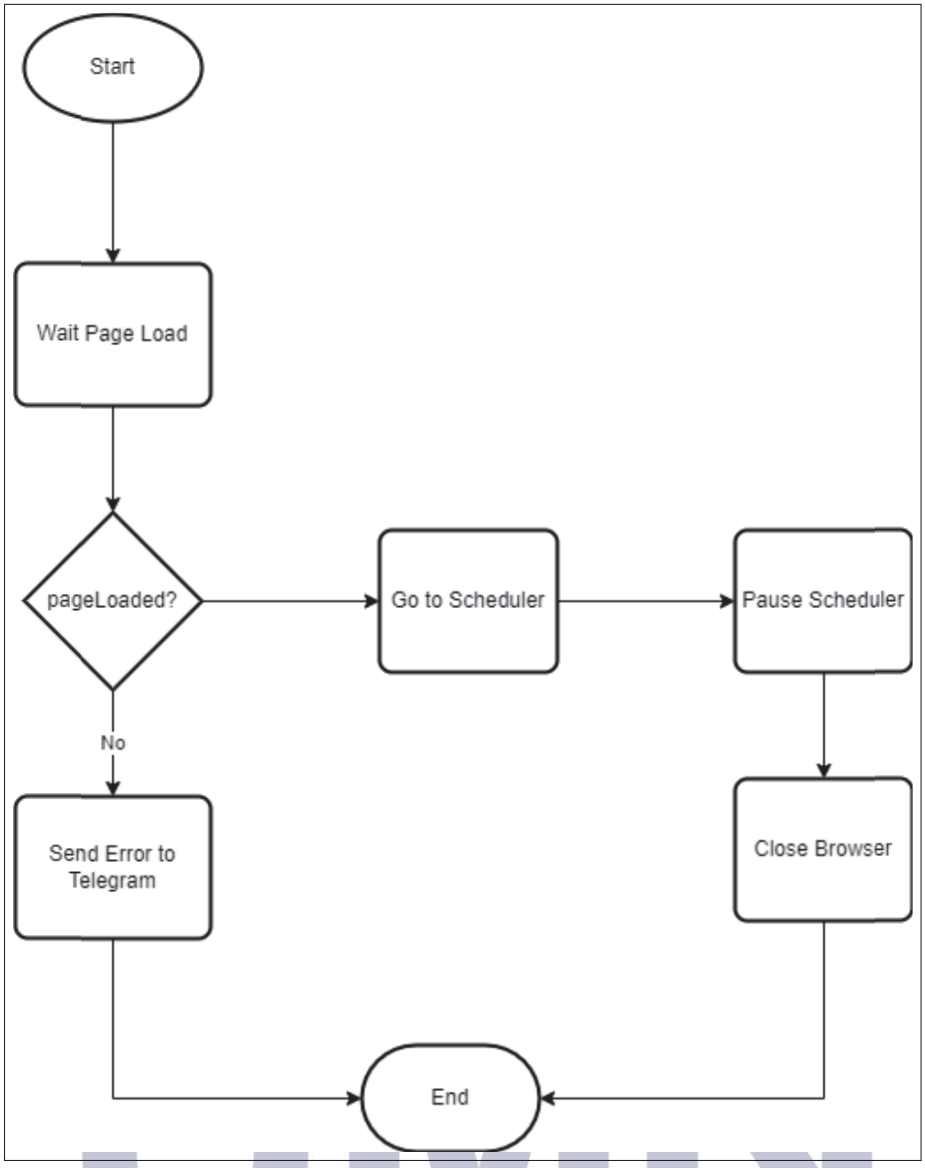
Proses ini dimulai dengan mengisi variabel flowList untuk ringkasan pendataan. Kemudian memulai proses *clear cache* dengan menjalankan *file* .bat bernama clearcache.bat. Terdapat *delay* selama 5 menit sebelum melanjutkan ke proses selanjutnya. Setelah proses *clear cache* dilakukan, proses login dimulai dengan menggunakan modul *login*. Proses ini dapat dilihat pada Gambar 3.25.



Gambar 3.25. Proses *Clear Cache & Login*

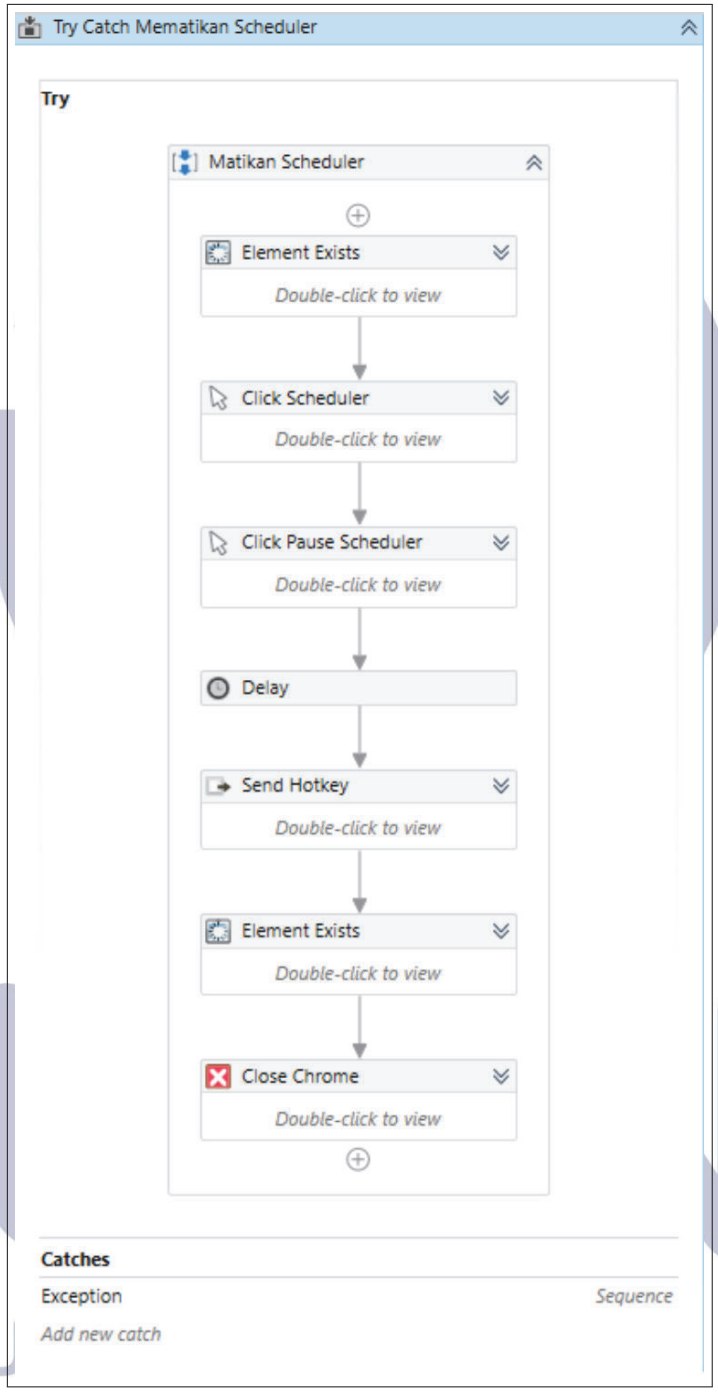
B. Mematikan *Scheduler*

Alur proses mematikan *scheduler* dapat dilihat pada Gambar 3.27 dan hasil implementasi dapat dilihat pada Gambar **??**



Gambar 3.26. Flowchart proses mematikan *scheduler*

Proses mematikan *scheduler* dimulai dengan menunggu halaman *web* berhasil dibuka, ketika tidak berhasil maka akan mengirimkan *error message* ke Telegram. Namun jika halaman *web* berhasil dibuka, maka RPA akan menuju ke halaman *Scheduler* untuk melakukan memberhentikan *scheduler* kemudian *browser* akan ditutup.

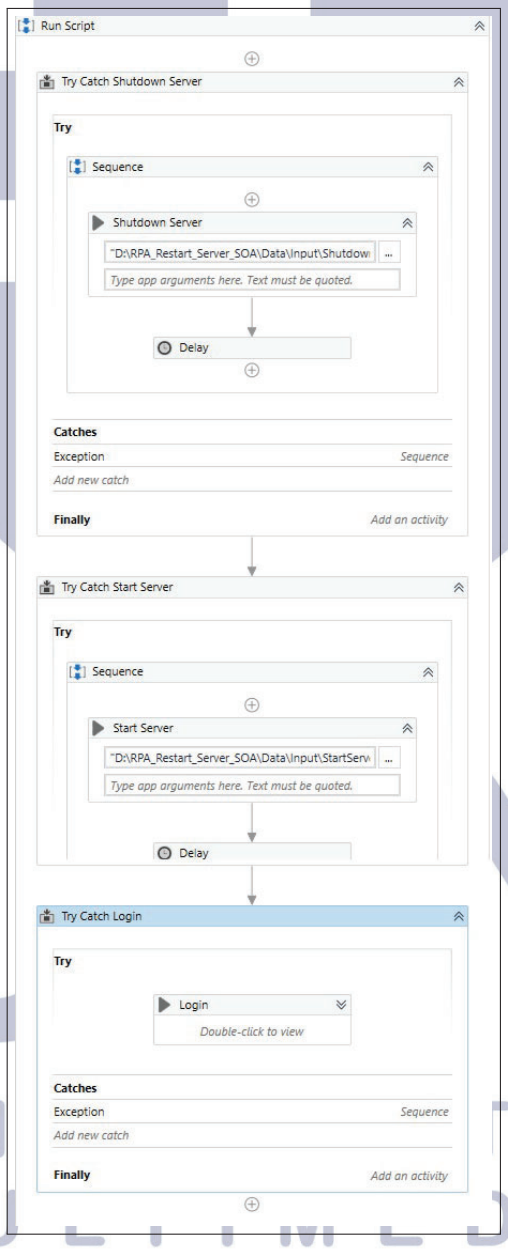


Gambar 3.27. Hasil implementasi proses mematikan *scheduler*

C. *Run Script*

Proses ini dimulai dengan mematikan *server* menggunakan *file* .bat bernama shutdown.bat. Proses mematikan *server* berlangsung selama 10 menit. Setelah

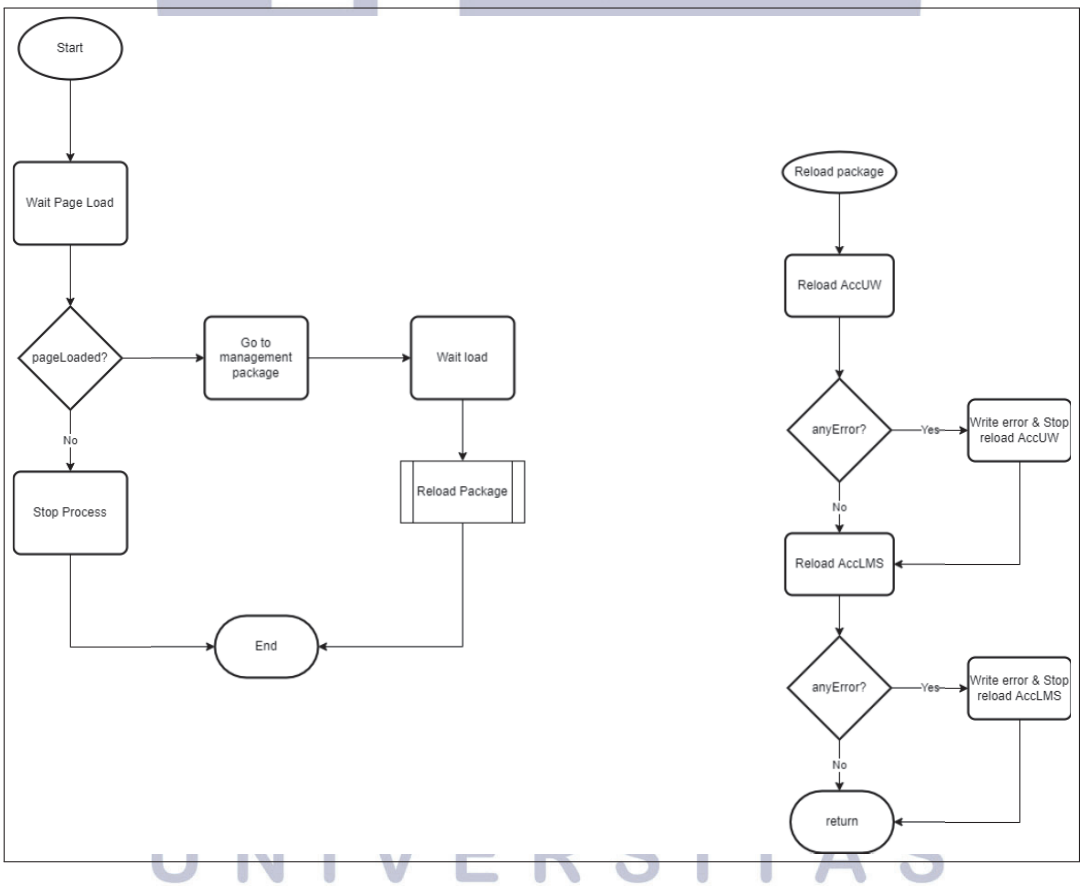
mematikan *server*, server dihidupkan kembali dengan menjalankan *file* .bat bernama starserver.bat. Proses menyalakan *server* berlangsung selama 5 menit. Kemudian dilakukan proses *login* kembali dengan menggunakan modul *login*. hasil implementasi proses menjalankan *script* dapat dilihat pada Gambar 3.28



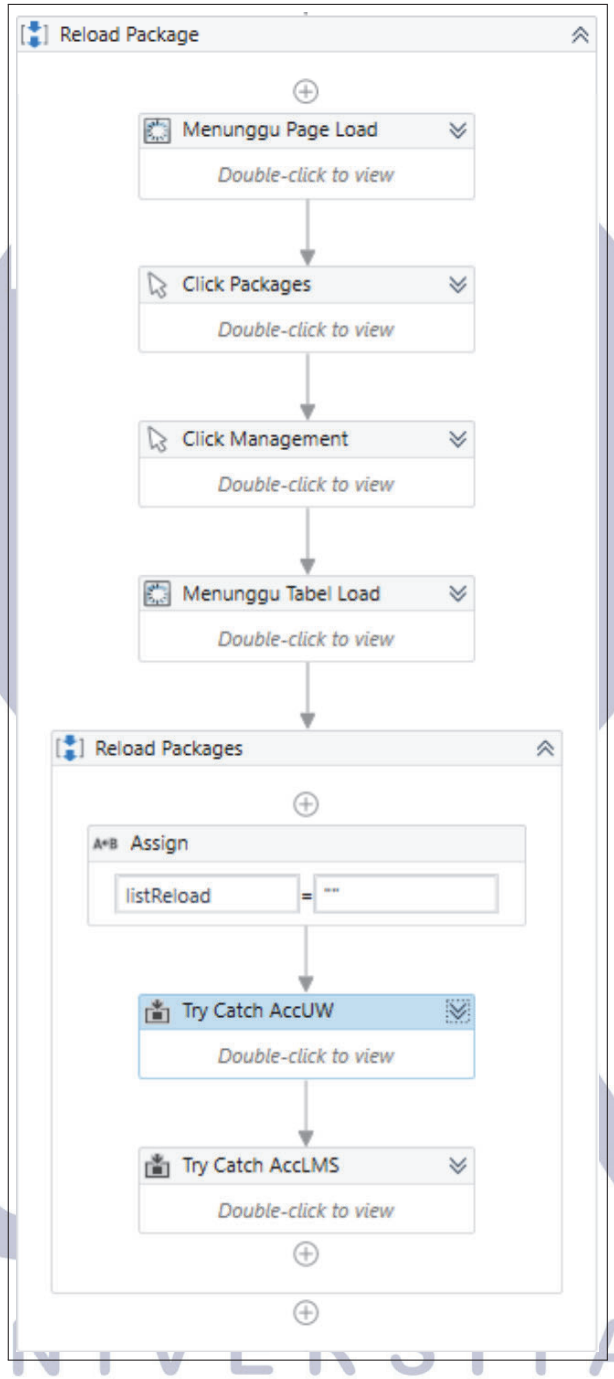
Gambar 3.28. Hasil implementasi proses *run script*

D. Reload Package

Proses *reload package* dimulai dengan menunggu halaman *web* berhasil tampil, ketika berhasil tampil maka RPA akan membuka halaman *management package* untuk melakukan proses *reload package*. Proses *reload package* dimulai dengan melakukan *reload* untuk *package* bernama AccUW kemudian *package* AccLMS. Jika terdapat *error* pada proses *reload*, maka akan dilakukan pendataan proses yang berhasil dan gagal di *reload*, kemudian proses akan lanjut ke tahap berikutnya. Alur proses *reload package* dapat dilihat pada Gambar 3.29 dan hasil implementasi dapat dilihat pada Gambar 3.30.



Gambar 3.29. Flowchart *reload package*

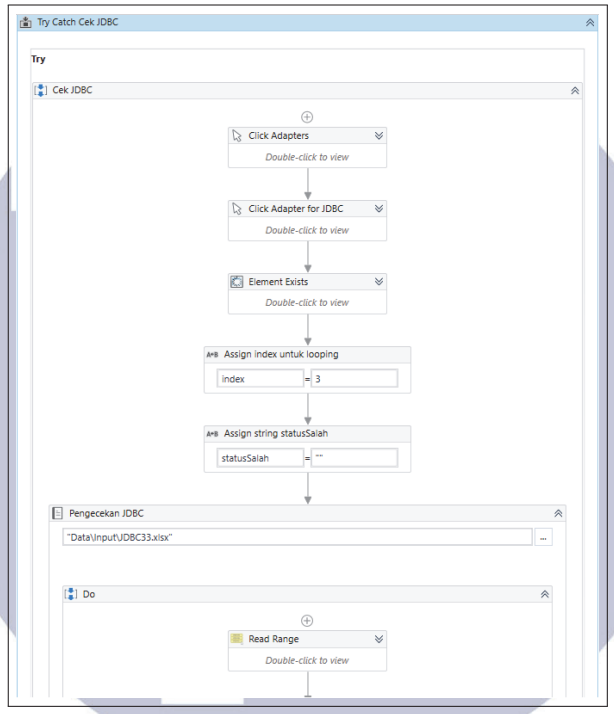


Gambar 3.30. Hasil implementasi proses *reload package*

E. Cek Java Database Connectivity (JDBC)

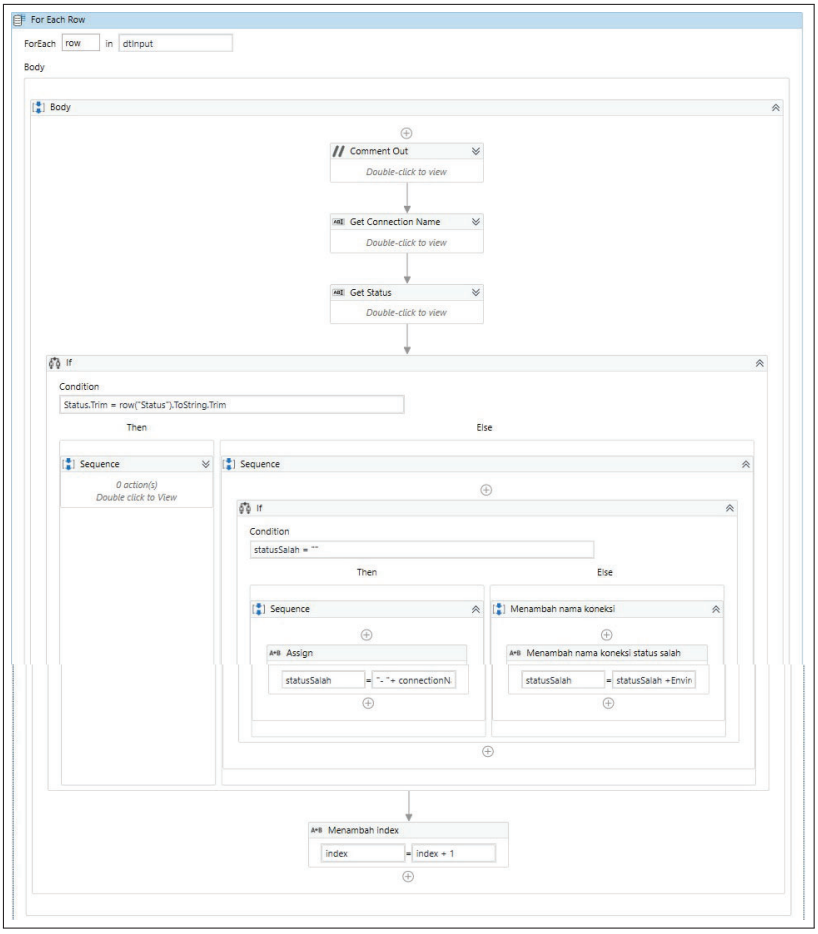
Proses cek JDBC dimulai dengan membuka halaman *Adapter for JDBC*, kemudian menginisialisasi variabel *index* dan statusSalah. Setelah melakukan proses inisialisasi dilakukan proses pembacaan data *file* excel bernama JDBC33.xlsx.

Proses ini dapat dilihat pada Gambar 3.31.



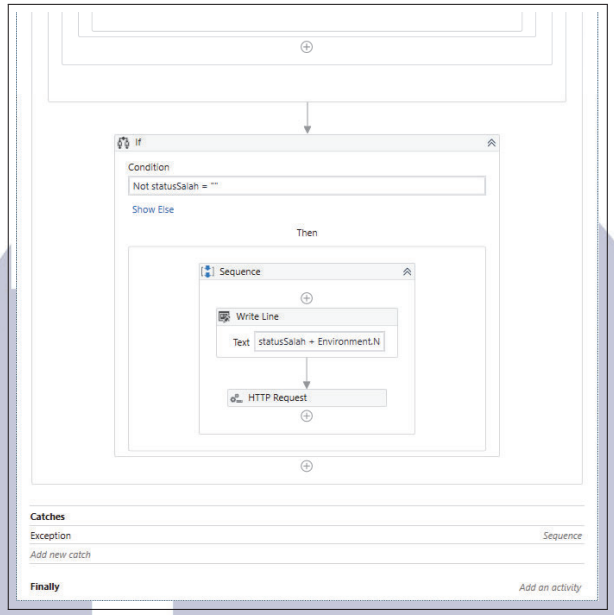
Gambar 3.31. Hasil implementasi proses cek JDBC 1

Proses selanjutnya adalah melakukan pengulangan untuk pengecekan status JDBC pada halaman *web* dengan data yang ada di excel. Jika status JDBC berbeda maka nama koneksi dengan status yang berbeda akan dimasukkan ke variabel statusSalah. Index digunakan untuk berpindah baris pada tabel halaman *web*. Proses ini dapat dilihat pada Gambar 3.32.



Gambar 3.32. Hasil implementasi proses cek JDBC 2

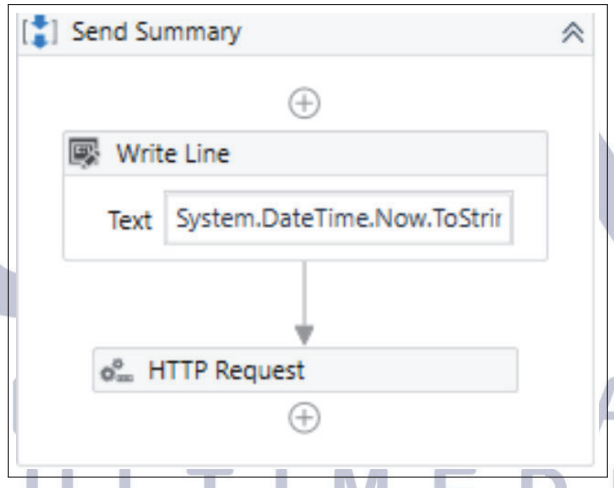
Proses yang terakhir merupakan pengecekan variabel statusSalah. Jika statusSalah tidak kosong maka pesan ke Telegram yang berisi informasi nama koneksi JDBC dengan status salah akan dikirimkan. Proses ini dapat dilihat pada Gambar 3.33.



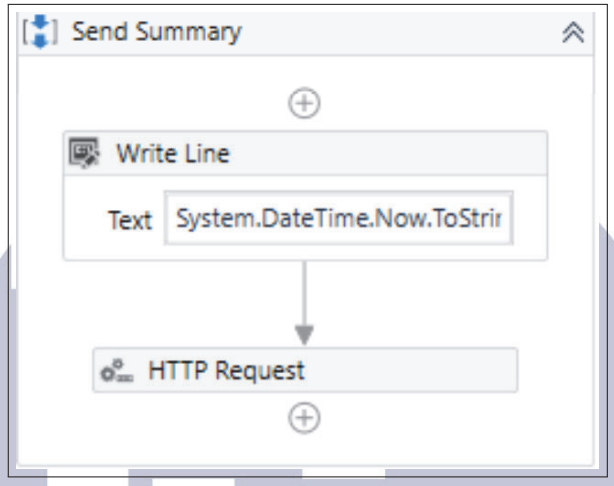
Gambar 3.33. Hasil implementasi proses cek JDBC 3

F. Summary

Proses *summary* merupakan proses pengiriman pesan ke Telegram mengenai informasi proses *restart*. Hasil implementasi dapat dilihat pada Gambar 3.34.



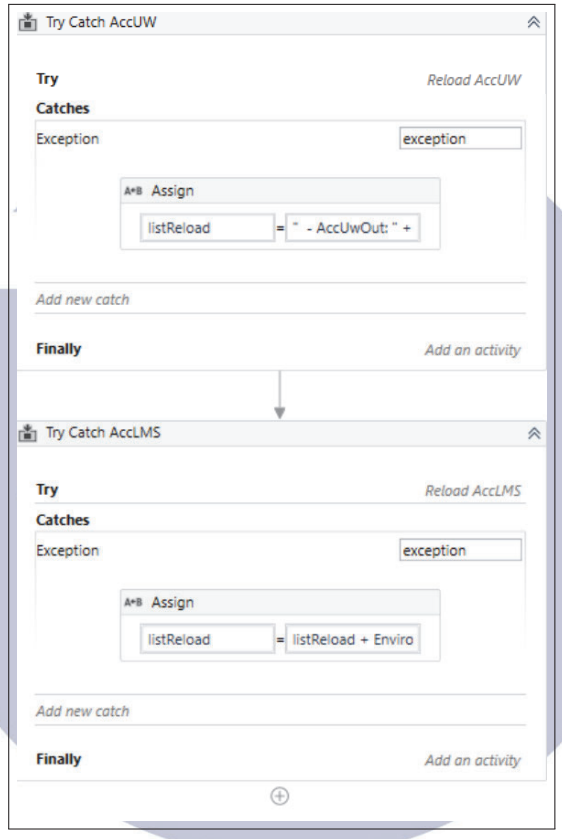
Gambar 3.34. Hasil implementasi *summary*



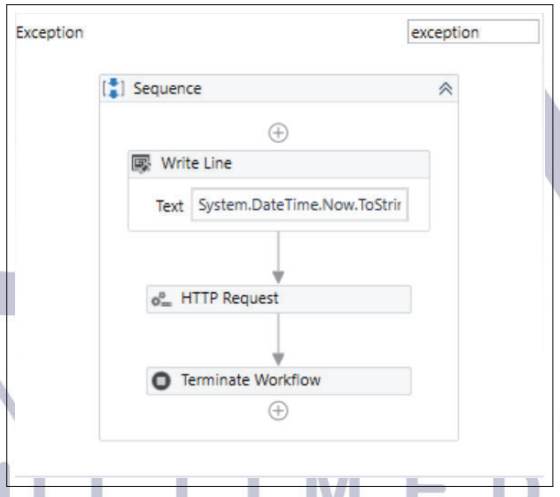
Gambar 3.35. Pesan terkirim di Telegram

G. Exception

Pada *main process* dilakukan *try catch* dimana ketika terjadi kegagalan dalam proses *restart* maka RPA akan mengirimkan pemberitahuan ke Telegram kemudian mematikan proses *restart SOA*. Namun, pada bagian *reload package*, ketika terjadi kegagalan maka proses *reload package* akan dihentikan kemudian RPA akan mendata status *reload package* dan melanjutkan ke proses berikutnya. *Exception* dapat dilihat pada Gambar 3.36 dan Gambar 3.37



Gambar 3.36. *Exception* pada *reload package*

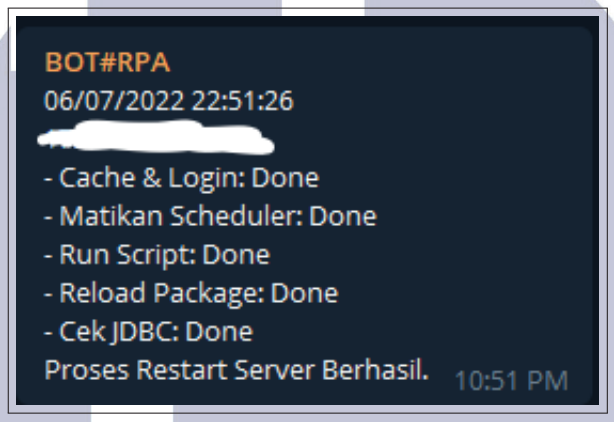


Gambar 3.37. *Exception* pada proses lain

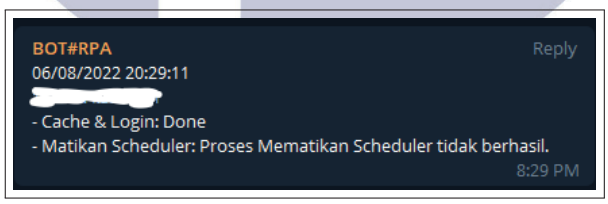
3.7.4 Output

Pesan telegram yang terkirim ketika semua proses berjalan lancar dapat dilihat pada Gambar 3.38. Ketika terdapat proses yang gagal, pesan telegram yang

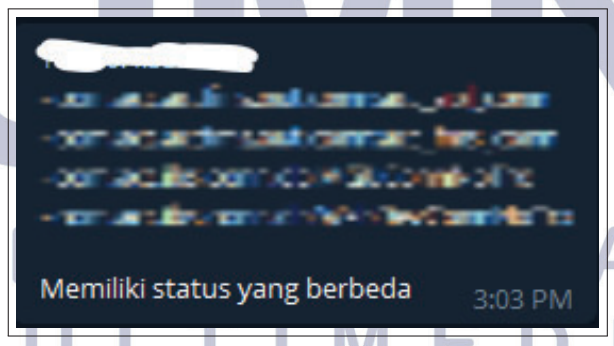
muncul dapat dilihat pada Gambar 3.39. Pesan telegram yang terkirim ketika terdapat status JDBC yang berbeda dapat dilihat pada Gambar 3.40. Gambar 3.41 merupakan pesan yang terkirim ketika terdapat kegagalan pada proses *reload package*.



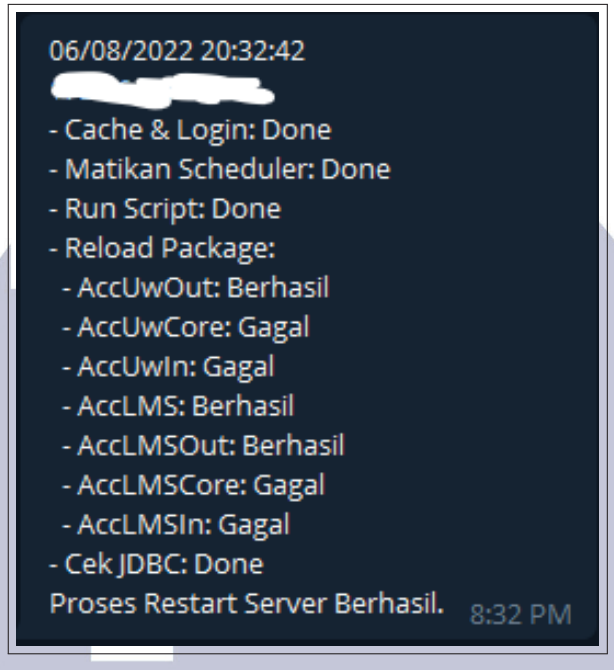
Gambar 3.38. Pesan Telegram 1



Gambar 3.39. Pesan Telegram 2



Gambar 3.40. Pesan Telegram 3



Gambar 3.41. Pesan Telegram 4

3.8 Kendala dan Solusi

Selama kerja magang, terdapat beberapa kendala yang dialami dan juga solusi yang ditemukan untuk menyelesaikan kendala yang dialami.

3.8.1 Kendala

Berikut adalah kendala-kendala yang dialami:

1. Tidak dapat melakukan pengambilan data dikarenakan tidak memiliki akses ke *database*.
2. Tidak mengetahui cara menggunakan bahasa *Python* untuk pengambilan data melalui *API*.
3. Tidak mengetahui cara menggunakan *API* Telegram untuk pengiriman pesan
4. *File* .bat untuk proses *clear cache*, mematikan dan menyalakan *server* belum diberikan pengguna.

3.8.2 Solusi

Berikut adalah solusi yang ditemukan berdasarkan kendala yang dihadapi selama pengembangan aplikasi ini:

1. Menggunakan *Python* untuk pengambilan data.
2. Mempelajari cara menggunakan *Python* dengan melakukan *searching*.
3. Mempelajari *API* Telegram melalui *website* Telegram.
4. Menggunakan *file* .bat *dummy*.