Lampiran B

Software Architecture Document

PENGEMBANGAN APLIKASI PENGELOLAAN PRESENSI DAN PERIZINAN ASRAMA MAHASISWA KAMPUS POLBAN BERBASIS *WEB*

Oleh:

Kelompok TA 208

ICHWAN LATIF FAJARI	**NIM: 181511046**
IRFAN SISWARA	**NIM: 181511048**
RIZQA FAUZIYYAH NABILA	**NIM: 181511065**

Revision History

Date	Version	Description	Author
10/05/2021	1.0	Pembuatan Dokumen SAD Pertama – *Introduction dan Architectural Representation*	Rizqa Fauziyyah N Ichwan Latif F
16/05/2021	2.0	Penambahan bagian *Architectural Goals and Constraint*	Rizqa Fauziyyah N Ichwan Latif F
19/05/2021	3.0	Penambahan bagian *Logical View dan Process View*	Rizqa Fauziyyah N Ichwan Latif F
22/05/2021	4.0	Penambahan bagian *Deployment View* dan *Implementation View*	Rizqa Fauziyyah N Ichwan Latif F
02/06/2021	5.0	Perubahan *Sequence Diagram* Perubahan *Class Diagram* Perubahan *Package Diagram*	Rizqa Fauziyyah N Irfan Siswara Ichwan Latif
02/07/2021	6.0	Perubahan semua diagram perancangan Penambahan algoritma pada *Sequence Diagram*	Rizqa Fauziyyah N Irfan Siswara
17/07/2021	7.0	*Balancing document*	Rizqa Fauziyyah N

I. *Introduction*

Bagian ini menjelaskan gambaran umum mengenai tujuan penulisan dokumen, cakupan aplikasi, definisi istilah, akronim, dan singkatan yang digunakan dalam dokumen, referensi yang digunakan dalam penulisan dokuman, dan deskripsi setiap bagian pada dokumen *Software Architecture Document* (SAD) ini. Penggambaran UML pada dokumen SAD menggunakan *tools* draw.io.

I.1 *Purpose*

Dokumen ini memberikan penjelasan dan gambaran terkait arsitektur yang dirancang dan digunakan aplikasi pengelolaan presensi dan perizinan asrama Polban. Dokumen ini menggunakan beberapa pandangan arsitektur yang berbeda agar dapat mengasilkan gambaran dari berbagai aspek. Hal tersebut memiliki tujuan untuk menggambarkan arsitektur yang telah dibuat pada aplikasi, sehingga dapat memberikan kemudahan dalam proses implementasi.

I.2 *Scope*

Aplikasi pengelolaan presensi dan perizinan adalah aplikasi yang ditujukan bagi asrama Polban. Aplikasi bertujuan untuk dapat mengidentifikasi keberadaan mahasiswa dengan mengakses lokasi saat pengisian presensi kehadiran. Aplikasi juga mendukung pengajuan perizinan dan *approval* secara *online,* sehingga dapat mengefisiensikan waktu dalam proses persetujuan pengelola asrama dan Wakil Direktur (Wadir) 3.

Gambaran umum pengelolaan presensi kehadiran yaitu aplikasi dapat mengidentifikasi keberadaan mahasiswa yang sedang melakukan presensi. Aplikasi mampu mengukur titik koordinat mahasiswa dengan koordinat asrama sehingga dapat menentukan status kehadiran mahasiswa tersebut sedang berada dalam lingkungan asrama atau tidak. Terdapat perekapan data presensi yang dapat diakses oleh pengelola asrama dan Wadir 3 selaku *stakeholder* yang terlibat pada aplikasi. Gambaran umum pengelolaan perizinan pada aplikasi ini yaitu dapat melakukan pengajuan izin dan pemberian *approval*. Aplikasi mampu mengirimkan notifikasi

kepada pengelola asrama melalui *e-mail* saat mahasiswa mengajukan perizinan, begitupun mahasiswa akan menerima notifikasi *e-mail* jika perizinan yang diajukan telah ditindaklanjuti oleh Wadir 3. Terdapat perekapan data perizinan yang dapat diakses oleh pengelola asrama dan pihak manajemen (Wadir 3). Aplikasi yang dibangun berlaku untuk asrama gedung A, B, dan C. Aplikasi dibangun menggunakan arsitektur RESTful API dan teknologi pengembangan aplikasi menggunakan Laravel dan React.js. Adapun DBMS yang digunakan adalah MySQL.

I.3 *Definition, Acronyms, and Abbreviations*

Bagian ini berisi definisi semua istilah, akronim, dan singkatan yang digunakan dalam dokumen ini.

1.3.1 *Definition*

No	Istilah	Deskripsi
1.	*Approval*	*Approval* adalah persetujuan yang akan diberikan oleh pengurus kepada mahasiswa terhadap pengajuan izin yang dilakukan.
2.	*Geolocation*	*Geolocation* adalah identifikasi lokasi geografis suatu objek menggunakan perangkat yang tersambung ke internet.
3.	*Notifikasi*	Notifikasi adalah pemberitahuan atau kabar tentang pengajuan izin yang masuk dan hasil *approval* dari pengajuan izin tersebut.
4.	*Query*	Bahasa yang digunakan untuk melakukan manipulasi data pada *database.*
5.	SAD	SAD adalah dokumen yang menjelaskan tentang arsitektur dan perancangan yang digunakan dalam pengembangan aplikasi.

1.3.2 *Acronyms*

No	Akronim	Deskripsi
1.	Polban	Politeknik Negeri Bandung
2.	Wadir	Wakil Direktur

1.3.3 *Abbreviations*

No	Singkatan	Deskripsi
1.	API	*Application Programming Language*
2.	GPS	*Global Positioning System*
3.	MVC	*Model, View, Controller*
4.	PC	*Personal Computer*
5.	RDBMS	*Rational Database Management System*
6.	SRS	*Software Requirements Specification*
7.	SQL	*Structured Query Language*
8.	UI	*User Interface*
9.	UML	*Unified Modeling Language*
10.	URL	*Uniform Resource Locator*

I.4 *References*

Referensi yang digunakan untuk penulisan dokumen ini adalah sebagai berikut:

1. Template: Software Architecture Document, Rational Software Corporation 1987-2001.
2. Larman, Craig. (2004): *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition. Addison Wesley Professional.*

I.5 *Overview*

Bagian penjelasan yang disajikan dalam dokumen ini disusun dengan struktur sebagai berikut:

1. *Introduction*

   Bagian ini menjelaskan gambaran umum mengenai tujuan penulisan dokumen, cakupan aplikasi, definisi istilah, akronim, dan singkatan yang digunakan dalam dokumen, referensi yang digunakan untuk menulis dokumen, dan deskripsi setiap bagian pada dokumen.

2. *Architectural Representation*

   Bagian ini menjelaskan mengenai arsitektur yang digunakan pada aplikasi yaitu arsitektur aplikasi dalam lima pandangan yang berbeda (*logical view, process view, deployment view, data view,* dan *use case view*).

3. *Architectural Goals and Constraints*

   Bagian ini menjelaskan tentang batasan dan tujuan aplikasi yang memiliki berdampak signifikan terhadap perancangan arsitektur aplikasi yang dibuat.

4. *Logical View*

   Bagian ini menjelaskan tentang pandangan logis dari arsitektur. Menjelaskan bagian arsitektur yang signifikan dari model desain, seperti dekomposisi menjadi subsistem atau menggunakan organisasi paket.

5. *Process View*

   Bagian ini menjelaskan tentang arsitektur aplikasi dari sudut pandang proses. Hal ini mencakup dekomposisi aplikasi menjadi proses yang terurut.

6. *Deployment View*

   Bagian ini menjelaskan tentang arsitektur aplikasi dari sudut pandang penyebaran. Hal ini mencakup spesifikasi, dan konfigurasi perangkat keras tempat aplikasi tersebut digunakan dan dijalankan.

7. *Data View*

   Bagian ini menjelaskan arsitektur aplikasi dari sudut pandang kebutuhan data dan skema pengelompokkan data ke dalam kelas.

8. *Implementation View*

   Bagian ini menjelaskan tentang arsitektur aplikasi dari sudut pandang pengembangan. Hal ini mencakup struktur keseluruhan dari model implementasi dan arsitektur dari setiap komponen aplikasi.

9. *Size and Performance*

   Bagian ini menjelaskan mengenai ukuran atau volume dan performa dari aplikasi.

10. *Quality*

    Bagian ini menjelaskan tentang kualitas dari aplikasi yang dibuat.

II. *Architectural Representation*

Arsitektur yang digunakan dalam merepresentasikan aplikasi yang dibangun adalah dengan menggunakan pendekatan *view model* 4 + 1. Pendekatan ini digunakan untuk menggambarkan arsitektur aplikasi dari lima sudut pandang yang berbeda, antara lain adalah *logical view, process view, deployment view, implementation view,* dan *use case view* sebagai *view model* tambahan. (Larman, 2004). Pada setiap sudut pandang arsitektur sistem dilakukan pemodelan UML yang bertujuan untuk memudahkan dalam melakukan penggambaran desain dari aplikasi. *Use case* atau *scenario view* tidak akan dijelaskan pada dokumen ini, karena sudah dijelaskan pada dokumen SRS.

II.1 *Logical View*

a.	*Concerns*	:	Pengorganisasian secara konseptual pada aplikasi berdasarkan *package, class,* dan *interface.* (Larman, 2004). Sudut pandang ini menggambarkan fungsionalitas elemen utama pada aplikasi.
b.	*Stakeholders*	:	Pengembang (*developer*)
c.	*UML Modeling*	:	*Package Diagram*

II.2 *Process View*

a.	*Concerns*	:	Menggambarkan arsitektur aplikasi dari sudut pandang proses aplikasi berjalan dan perilaku objek berkomunikasi dengan aplikasi. Berhubungan dengan aspek dinamis dari perancangan aplikasi. (Larman, 2004).
b.	*Stakeholders*	:	Pengembang (*developer*)
c.	*UML Modeling*	:	*Sequence Diagram*

II.3 *Deployment View*

a.	*Concerns*	:	Penggambaran arsitektur dari sisi *deployment dan* komponen aplikasi ke *node* pemrosesan dan konfirgurasinya. (Larman, 2004).
b.	*Stakeholders*	:	Pengembang (*developer*)
c.	*UML Modeling*	:	*Deployment Diagram*

II.4 *Data View*

a.	*Concerns*	:	Penggambaran pengelompokan skema data yang digunakan. (Larman, 2004).
b.	*Stakeholders*	:	Pengembang (*developer*)
c.	*UML Modeling*	:	*Class Diagram*

III. *Architectural Goals and Constraint*

Bagian ini berisi penjelasan terkait persyaratan dan tujuan aplikasi yang memiliki dampak signifikan terhadap arsitektur. Persyaratan tersebut dari sudut pandang *server side, client side, security, persistence, reliability,* dan *performance.*

III.1 *Server Side*

Aplikasi akan di *host* secara *online* dan menggunakan RDBMS MySQL. Aplikasi menggunakan REST API untuk berkomunikasi dengan *client side.*

III.2 *Client Side*

Pengguna dapat mengakses aplikasi menggunakan *device* manapun yang terhubung ke jaringan internet. Aplikasi yang digunakan berbasis *web* yang dapat diakses melalui *browser* pengguna.

III.3 *Security*

Aplikasi harus dapat memberikan keamanan sehingga data pengguna akan terjaga dengan aman. Oleh karena itu aplikasi menerapkan keamanan dasar, yaitu :

1. Aplikasi menerapkan *authentication* dan enkripsi token pada pengelolaan akses pengguna atau otorisasi.
2. Aplikasi memanfaatkan validasi pengubahan *password* pengguna melalui pengiriman *e-mail* konfirmasi saat mengubah *password.*

III.4 *Persistance*

Kesinambungan data yang digunakan pada aplikasi menggunakan basis data relasional.

III.5 *Reliability*

Aplikasi akan melewati tahap pengujian *black box testing* dengan pendekatan *unit testing.* Hal tersebut dilakukan untuk memastikan bahwa aplikasi sudah berjalan sesuai fungsi utama melalui pengecekkan per modul.

III.6 *Performance*

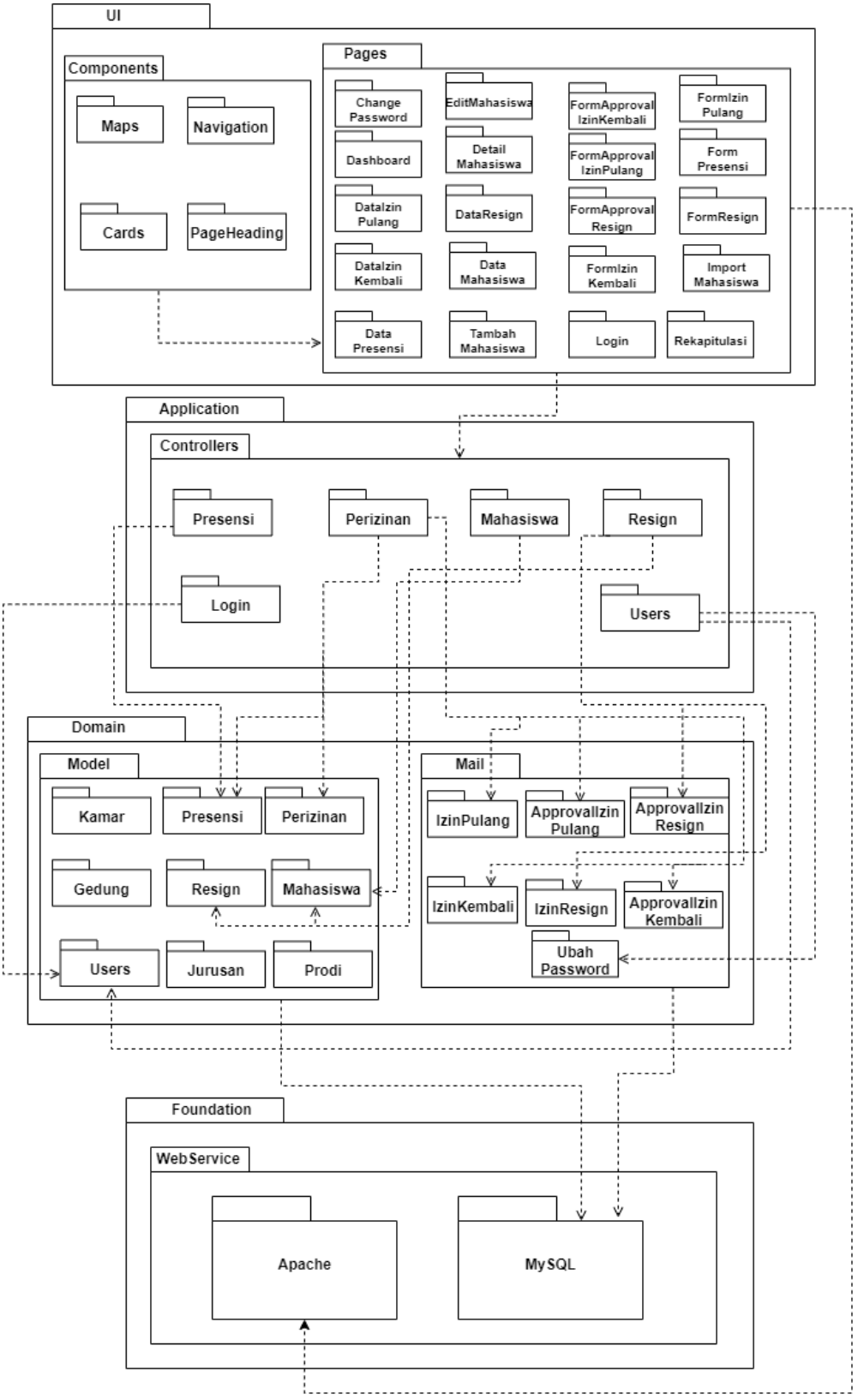
Kinerja aplikasi dapat bergantung pada perangkat keras dan koneksi internet yang digunakan pengguna.

IV. *Logical View*

Bagian ini berisi penjelasan mengenai arsitektur logis (*logical view*). Hal ini mencakup gambaran umum arsitektur aplikasi dari model desain yang didekomposisi menjadi paket-paket, dan paket desain yang signifikan pada arsitektur yang akan didekomposisi kembali menjadi kelas.

IV.1 *Overview*

Package diagram yaitu kumpulan kelas yang menyusun keseluruhan arsitektur pada aplikasi yang dikelompokkan ke dalam *package-package* berdasarkan fungsi kelas. UML *package diagram* biasanya digunakan untuk mengilustrasikan arsitektur dari sisi logika *layer system, subsystem,* dan *packages*. (Larman, 2004). Pandangan logis dari aplikasi pengelolaan presensi dan perizinan terdiri atas empat paket utama yaitu packet *UI, application, domain,* dan *foundation*. Berikut merupakan gambaran dari setiap paket yang terdapat pada Gambar IV.1.



Gambar IV.1 *Package diagram*

IV.4.1 *User Interface* (UI)

Paket ini berfungsi untuk menyimpan kelas atau *subpackage* yang berkaitan dengan menampilkan antarmuka pengguna.

IV.4.2 *Application*

Paket ini berfungsi untuk menyimpan kelas atau *subpackage* yang mengakses model dan mengirim *response* disaat *view* mengirimkan *request.* Paket ini berkaitan dengan *layer application.*

IV.4.3 *Domain*

Paket ini berfungsi untuk menyimpan kelas atau *subpackage* yang merepresentasikan konsep domain, berkaitan dengan logika bisnis dan pengelolaan data.

IV.4.4 *Foundation*

Paket ini berfungsi untuk menyimpan kelas atau *subpackage* pengelolaan *web service* dari aplikasi.

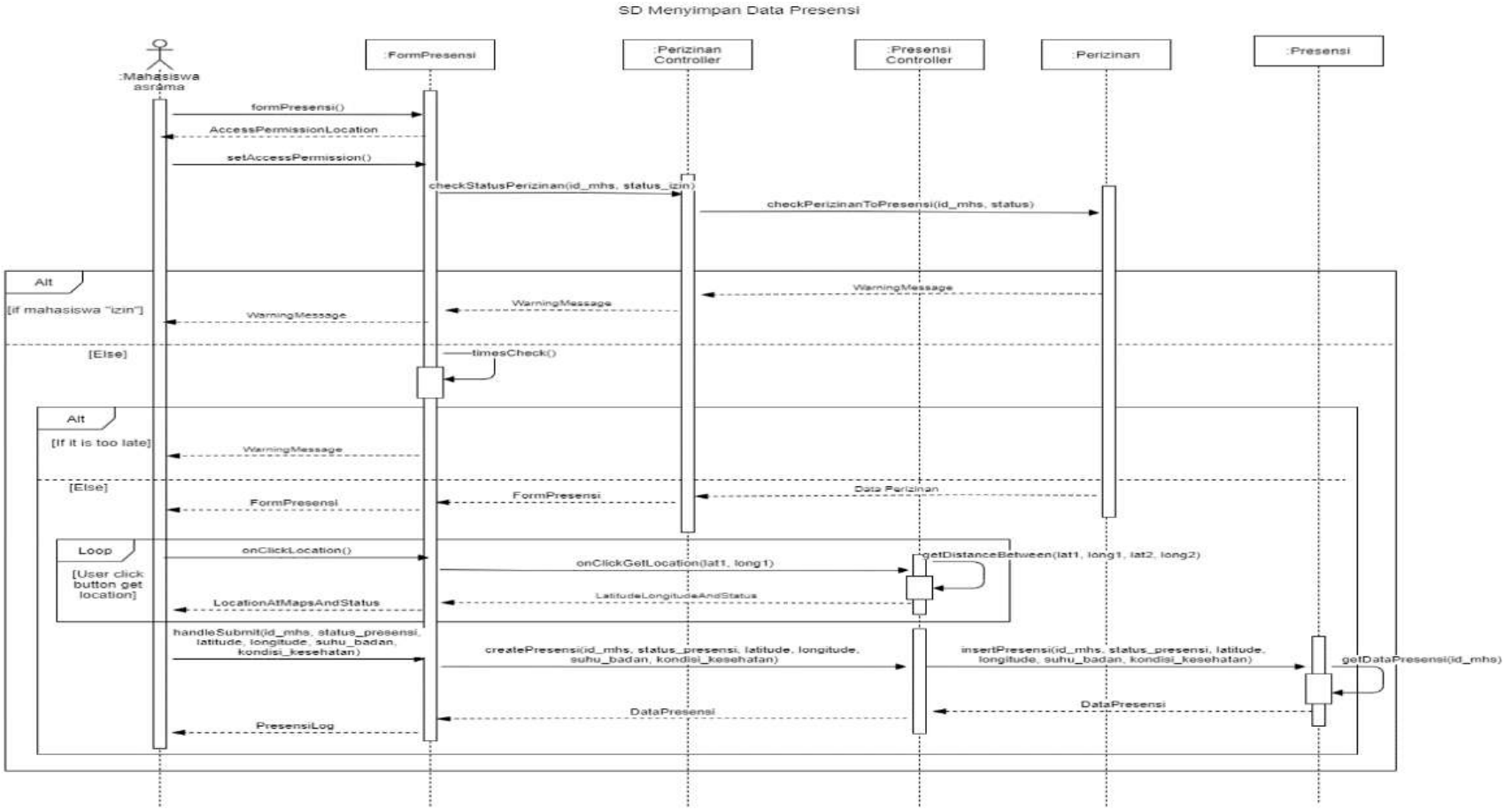
IV.2 *Architectural Significant Design Package*

Paket desain yang signifikan pada arsitektur, didekomposisi kembali menjadi kelas-kelas. Untuk merepresentasikan kelas-kelas yang dibutuhkan maka digunakan pemodelan UML yaitu *class diagram*. Pemodelan *class* dijelaskan pada subbab *data view.*

V. *Process View*

Bagian ini menjelaskan mengenai arsitektur aplikasi dari sudut pandang proses aplikasi. Bagian ini mencakup aspek dinamis aplikasi, proses aplikasi berjalan, dan bagaimana berkomunikasi antar keduanya. Pemodelan UML yang digunakan yaitu *sequence diagram* untuk merepresentasikan *process view*. Selain itu, *sequence diagram* merupakan pemodelan yang menggambarkan interaksi antar objek. *Sequence diagram* ini dibuat berdasarkan *use case* yang telah dibuat pada dokumen SRS. Berikut *sequence diagram* yang telah dibuat beserta penjelasannya.

5.1 *Sequence Diagram* : Menyimpan Data Presensi



Gambar V.1 SD-01 menyimpan data presensi

Tabel V.1 SD-01 menyimpan data presensi

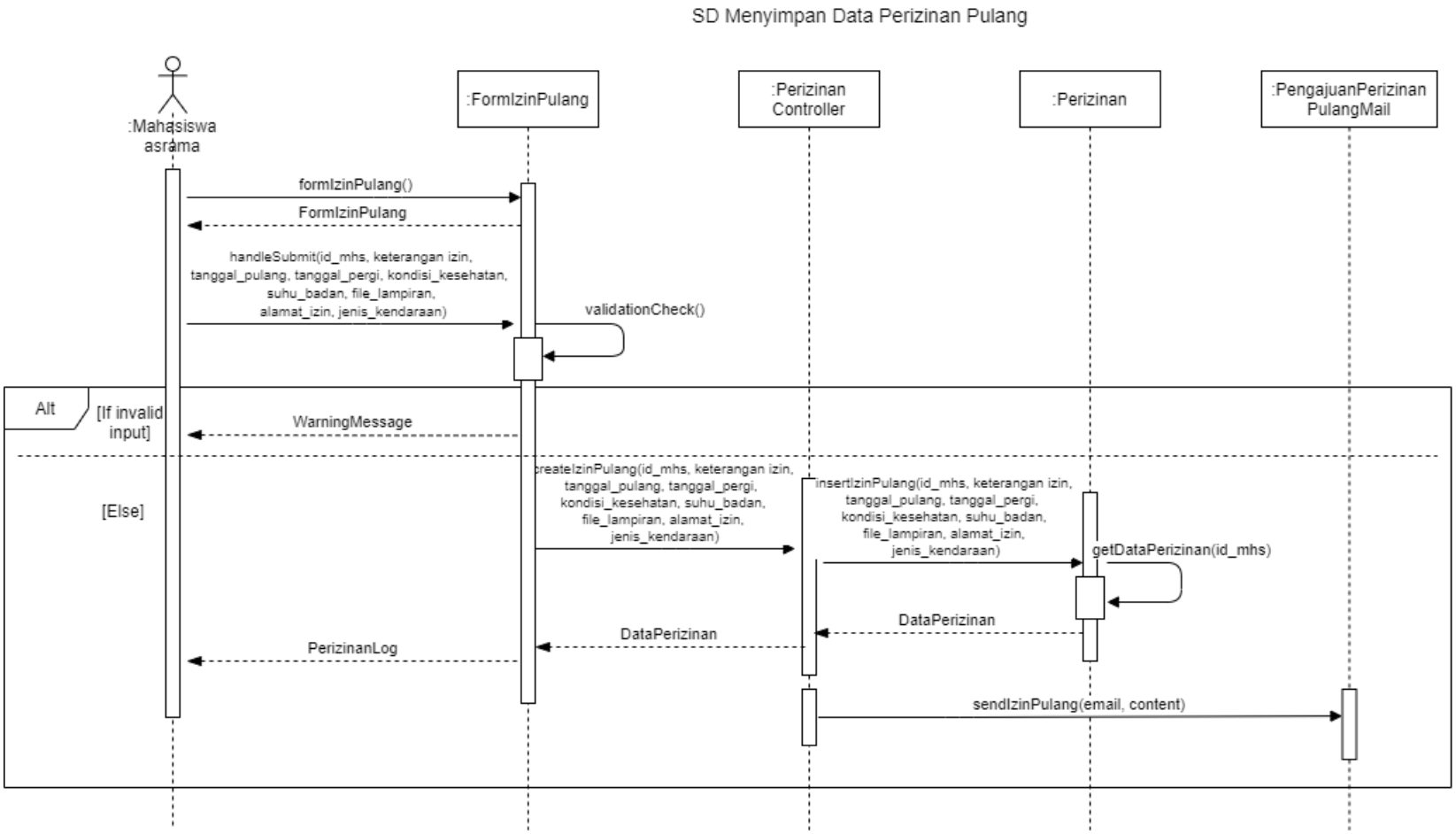
ID Sequence Diagram	SD-01
Nama Sequence Diagram	Menyimpan Data Presensi
ID Use Case	UC-01
ID Requirement	- REQ-F-03 - REQ-F-04 - REQ-F-05 - REQ-F-06 - REQ-F-07 - REQ-F-08 - REQ-F-09 - REQ-F-10 - REQ-F-11
Nama Class Terkait	PresensiController, PerizinanController, Presensi, dan Perizinan
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas pengisian presensi kehadiran. Mahasiswa mengisi kehadiran setiap hari sesuai dengan waktu yang telah ditentukan pada aplikasi.
Method yang Terkait	- formPresensi() - getDistanceBetween(lat1, long1, lat2, long2) - checkStatusPerizinan(id_mhs, status) - timesCheck() - onClickLocation() - onClickGetLocation(lat1, long1) - handleSubmit(id_mhs, status_presensi, latitude, longitude, suhu_badan, kondisi_kesehatan) - createPresensi(id_mhs, status_presensi, latitude, longitude, suhu_badan, kondisi_kesehatan) - getDistanceBetween(lat1, long1, lat2, long2) - checkPerizinanToPresensi(id_perizinan, status) - insertPresensi(id_mhs, status_presensi, latitude, longitude, suhu_badan, kondisi_kesehatan) - getDataPresensi(id_mhs)
Algoritma dan Script	
Kamus Data: id_mhs : Variabel tunggal bertipe integer status_presensi : Variabel tunggal bertipe integer latitude : Variabel tunggal bertipe float longitude : Variabel tunggal bertipe float lat1 : Variabel tunggal bertipe float long1 : Variabel tunggal bertipe float lat2 : Variabel tunggal bertipe float long2 : Variabel tunggal bertipe float dlat : Variabel tunggal bertipe float dlong : Variabel tunggal bertipe float	

```
a : Variabel tunggal bertipe float
c : Variabel tunggal bertipe float
rad : Variabel tunggal bertipe float
hasil : Variabel tunggal bertipe float
status_izin : Variabel tunggal bertipe array of character
validasi : Variabel komposit bertipe array
request : Variabel superglobal berhubungan dengan objek
permintaan HTTP
response : Variabel komposit bertipe JSON
Presensi : Variabel komposit bertipe array
Perizinan : Variabel komposit bertipe array
sql : Variabel tunggal bertipe array of character
send : Fungsi untuk mengirim request ke suatu api
getCurrentPosition : Fungsi untuk mendapatkan koordinat lokasi
terkini
suhu_badan : Variabel tunggal bertipe float
kondisi_kesehatan : Variabel tunggal bertipe array of character
JSON : fungsi untuk menyimpan dan bertukar data

Function formPresebsi()
     window.location.assign('formPresensi)
End Function

Function getDistanceBetween(lat1, long1, lat2, long2)
     dlat ← (lat2 - lat1) * 3.14/180
     dlong ← (long2 - long1) * 3.14/180
     a ← sin(dlat/2)² + sin(dlong/2)² * cos(lat1) * cos(lat2)
     rad ← 6371
     c ← 2 * asin(√a)
     hasil ← rad * c
     return hasil
End Function

Function checkStatusPerizinan(id_mhs, status)
Begin
     request.body ← JSON('id_mhs' ← id_mhs, 'status_izin' ←
status_izin)
     request.url ←
"base_url/api/presensi/checkPerizinanToPresensi"
     request.method ← GET
     response ← send (request) to request.url using
request.method
return response

if (response.status_izin == "izin")then
{
    alert('Anda sedang izin, tidak dapat melakukan presensi')
}else{
     timesCheck()
}

End Function

Function checkPerizinanToPresensi(id_mhs)
Begin
```

```
     perizinan ← SELECT * FROM perizinan Where id_mhs == id_mhs
     return  perizinan
End Function

Function timesCheck()
Begin
     if(Date().toLocaleTimeString() < '16.00.00' OR
Date().toLocaleTimeString() > '20.00.00')
     then
        alert('Sedang Tidak Dalam Waktu Presensi')
End Function


Procedure insertPresensi(id_mhs, status_presensi, latitude,
longitude, suhu_badan, kondisi_kesehatan)
Begin
         insert ← INSERT INTO presensi (status_presensi,
latitude, longitude, id_mhs, suhu_badan, kondisi_kesehatan)

         presensi ← getDataPresensi(id_mhs)
         return presensi
End Procedure

Function onClickLocation()
Begin
      latitude ← getCurrentPosition().latitude
      longitude ← getCurrentPosition().longitude
      request.body ← JSON('lat1' ← latitude, 'long1' ←
longitude)
     request.url ← "base_url/api/presensi/onClickGetLocation"
     request.method ← POST
     response ← send (request) to request.url using
request.method
     return response
End Function

Function onClickGetLocation(lat1, long1)
Begin
      hasil ← getDistanceBetween(lat1, long1, -
6.8719714,107.5711026)
      if(hasil > 0.05)
        then
           status ← "Alfa"
        else
           status ← "Hadir"
      endIf
      return  response('status' ← status, 'latitude' ← lat1,
'longitude' ← long1)
End Function

Function handleSubmit(id_mhs, id_mhs, status_presensi, latitude,
longitude, suhu_badan, kondisi_kesehatan)
Begin
```

```
     request.body ← JSON('id_mhs' ← id_mhs, status_presensi ←
status_presensi, latitude ←  latitude, longitude ←  longitude, ,
suhu_badan ← suhu_badan, kondisi_kesehatan ←  kondisi_kesehatan)
     request.url ← "base_url/api/presensi/createPresensi"
     request.method ← POST
     response ← send (request) to request.url using
request.method
     return response
End Function

Function createPresensi(id_mhs, status_presensi, latitude,
longitude, suhu_badan, kondisi_kesehatan)
Begin
      presensi = insertPresensi(id_mhs, status_presensi,
latitude, longitude, suhu_badan, kondisi_kesehatan)
      return response('presensi' ← presensi)
End Function

Function getDataPresensi(id_mhs)
Begin
      Presensi ← Select * from presensi where id_mhs = id_mhs
      return  Presensi
End Function
```

5.2 *Sequence Diagram* : Menyimpan Data Perizinan Pulang



Gambar V.2 SD-02 menyimpan data perizinan pulang

Tabel V.2 SD-02 menyimpan data perizinan pulang

ID Sequence Diagram	SD-02
Nama Sequence Diagram	Menyimpan Data Perizinan Pulang
ID Use Case	UC-02
ID Requirement	- REQ-F-12 - REQ-F-19 - REQ-F-20 - REQ-F-35
Nama Class Terkait	PerizinanController, Perizinan, dan PengajuanPerizinanPulangMail
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas pengisian pengajuan perizinan pulang. Mahasiswa mengisi formulir pengajuan (izin pulang) melalui aplikasi dan dikirim melalui *e-mail* kepada pengelola asrama.
Method yang Terkait	- formIzinPulang() - handleSubmit(id_mhs, keterangan izin, tanggal_pulang, tanggal_pergi, kondisi_kesehatan, suhu_badan, file_lampiran, alamat_izin, jenis_kendaraan) - validationCheck() - createIzinPulang(id_mhs, keterangan izin, tanggal_pulang, tanggal_pergi, kondisi_kesehatan, suhu_badan, file_lampiran, alamat_izin, jenis_kendaraan) - sendIzinPulang(email, content) - insertIzinPerizinan(id_mhs, keterangan izin, tanggal_pulang, tanggal_pergi, kondisi_kesehatan, suhu_badan, file_lampiran, alamat_izin, jenis_kendaraan) - getDataPerizinan (id_mhs)
Algoritma	

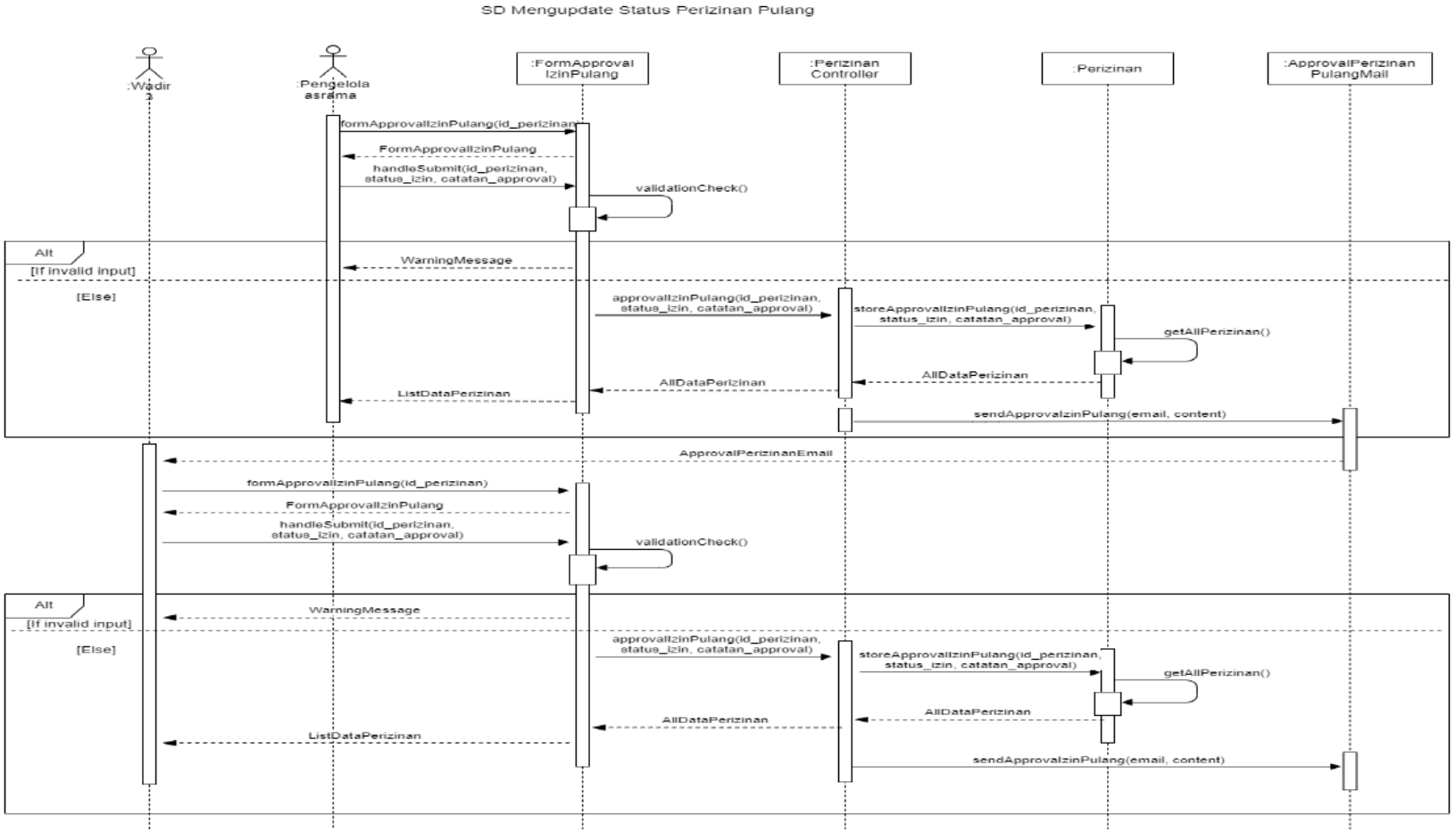
Kamus Data:

```
id_mhs : Variabel tunggal bertipe integer
tanggal_pergi : Variabel tunggal bertipe date
tanggal_pulang : Variabel tunggal bertipe date
keterangan_izin : Variabel tunggal bertipe  array of character
alamat_izin : Variabel tunggal bertipe  array of character
suhu_badan : Variabel tunggal bertipe float
file_lampiran : Variabel tunggal bertipe array od character
kondisi_kesehatan : Variabel tunggal bertipe array of character
jenis_kendaraan : Variabel tunggal bertipe array of character
validasi : Variabel komposit bertipe array
request : Variabel superglobal berhubungan dengan objek
permintaan HTTP
response : Variabel komposit bertipe JSON
```

```
Perizinan : Variabel komposit bertipe array
sql : Variabel tunggal bertipe array of character
JSON : fungsi untuk menyimpan dan bertukar data

Procedure formIzinPulang()
Begin
      window.location.assign('formIzinPulang')
End Procedure

Procedure handleSubmit(id_mhs)
Begin
      request.body ← JSON('id_mhs' ← id_mhs, 'tanggal_pergi' ←
tanggal_pergi, 'tanggal_pulang' ← tanggal_pulang,
'keterangan_izin' ← keterangan_izin, 'alamat_izin' ←
alamat_izin, 'suhu_badan' ← suhu_badan, 'kondisi_kesehatan' ←
kondisi_kesehatan, 'jenis_kendaraan' ← jenis_kendaraan,
'file_lampiran' ← file_lampiran)
      validasi ← validationCheck()
      if(validasi == true)
         then
           request.url ←
'base_url/api/perizinan/createIzinPulang'
           request.method ← 'POST'
           response ← send (request) to request.url using
request.method
         else
           alert('form tidak valid')

End Procedure

Function validationCheck()
Begin
 validasi ← request.body().check('id_mhs' ← 'required|number',
'tanggal_pergi' ← 'required', 'tanggal_pulang' ← 'required',
'keterangan_izin' ← 'required|max:125', 'alamat_izin' ←
'required|max:125', 'suhu_badan' ←
'required|numeric|between:30,50', 'kondisi_kesehatan' =>
'required|max:50', 'jenis_kendaraan' => 'required',
'file_lampiran' => .pdf || .jpg || .png | max:2 MB)
     return validasi
End Function

Function createIzinPulang(id_mhs, keterangan izin,
tanggal_pulang, tanggal_pergi, kondisi_kesehatan, suhu_badan,
file_lampiran, alamat_izin, jenis_kendaraan)
Begin
      perizinan ← insertIzinPulang (id_mhs, keterangan izin,
tanggal_pulang, tanggal_pergi, kondisi_kesehatan, suhu_badan,
file_lampiran, alamat_izin, jenis_kendaraan)

      sendIzinPulang(email, content)
      return response('perizinan' ← perizinan)
End Function

Procedure sendIzinPulang(email, content)
```

```
Begin
      send mail to('email' ← email) with('content' ← content)
End Procedure

Funtion insertIzinPulang(id_mhs, keterangan izin,
tanggal_pulang, tanggal_pergi, kondisi_kesehatan, suhu_badan,
file_lampiran, alamat_izin, jenis_kendaraan)
Begin
      insert ← INSERT INTO perizinan(id_mhs, keterangan izin,
tanggal_pulang, tanggal_pergi, kondisi_kesehatan, suhu_badan,
file_lampiran, alamat_izin, jenis_kendaraan)
      perizinan ← getDataPerizinan(id_mhs)
      return perizinan
End Function

Function getDataPerizinan(id_mhs)
Begin
      Perizinan ← Select * from perizinan where id_mhs = id_mhs
      return  Perizinan
End Function
```

5.3 *Sequence Diagram* : Mengupdate Status Perizinan Pulang



Gambar V.3 SD-03 mengupdate status perizinan pulang

Tabel V.3 SD-03 mengupdate status perizinan pulang

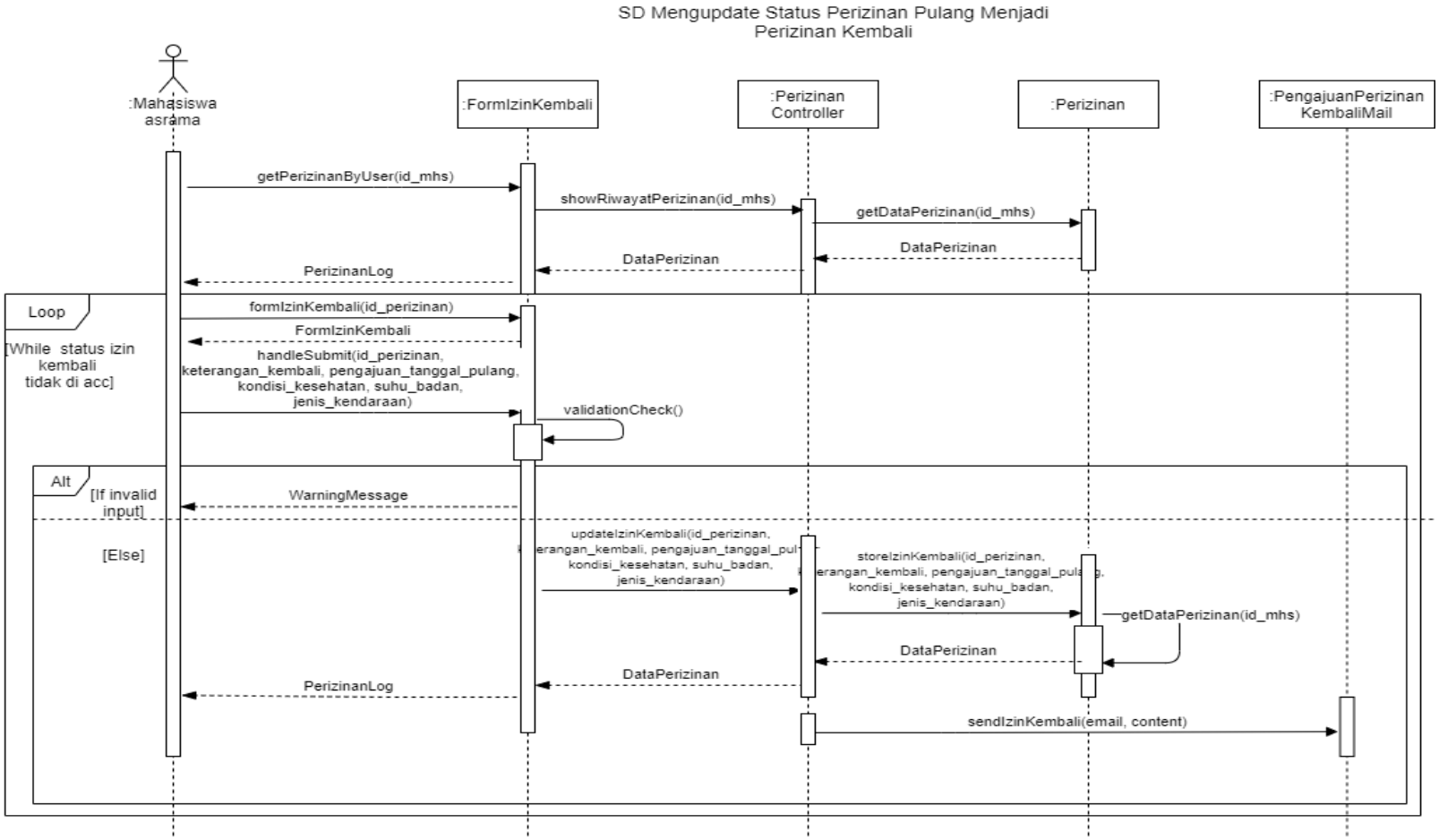
ID Sequence Diagram	SD-03
Nama Sequence Diagram	Mengupdate Status Perizinan Pulang
ID Use Case	UC-03
ID Requirement	- REQ-F-15 - REQ-F-20
Nama Class Terkait	PerizinanController, Perizinan, ApprovalPerizinanPulangMail
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas pengisian *approval* perizinan pulang oleh pengelola asrama dan Wadir 3. Hasil *approval* dikirim kepada mahasiswa melalui *e-mail*.
Method yang Terkait	- formApprovalIzinPulang(id_perizinan) - handleSubmit(id_perizinan, status_izin, catatan_approval) - validationCheck() - approvalIzinPulang(id_perizinan, status_izin, catatan_approval) - storeApprovalIzinPulang(id_perizinan, status_izin, catatan_approval)sendApprovalIzinPulang(email, content) - getAllPerizinan()

Algoritma

```
Kamus Data:
id_perizinan : Variabel tunggal bertipe integer
status_izin : Variabel tunggal bertipe integer
catatan_approval : Variabel tunggal bertipe array of character
validasi : Variabel komposit bertipe array
request : Variabel superglobal berhubungan dengan objek
permintaan HTTP
response : Variabel komposit bertipe JSON
Perizinan : Variabel komposit bertipe array
sql : Variabel tunggal bertipe array of character
JSON : fungsi untuk menyimpan dan bertukar data

Procedure formApprovalIzinPulang(id_perizinan)
Begin
      window.location.assign('formApprovalIzinPulang')
End Procedure

Procedure handleSubmit(id_perizinan, status_izin,
catatan_approval)
Begin
```

```
      request.body ← JSON('id_perizinan' ← id_perizinan,
'status_izin' ← status_izin, 'catatan_approval' ←
catatan_approval)
      validasi ← validationCheck()
      if(validasi == true)
         then
           request.url ←
'base_url/api/perizinan/approvalIzinPulang
           request.method ← 'POST'
           response ← send (request) to request.url using
request.method
         else
           alert('form tidak valid')
End Procedure

Function validationCheck()
Begin
  validasi ← request.body().check('id_mhs' ← 'required|number',
'status_izin' ← 'required')
     return validasi
End Function

Function approvalIzinPulang(id_perizinan, status_izin,
catatan_approval)
Begin
      perizinan ← storeApprovalIzinPulang(id_perizinan,
status_izin, catatan_approval)
      return response('perizinan' ← perizinan)
End Function

Function  storeApprovalIzinPulang(id_perizinan, status_izin,
catatan_approval)
      update ← UPDATE perizinan set  status_izin ← status_izin,
catatan_approval ← catatan_approval where id_perizinan ←
id_perizinan
      perizinan ← getAllPerizinan()
      sendApprovalIzinPulang(email, content)
      return perizinan
End Function

Procedure sendApprovalIzinPulang(email, content)
Begin
      send mail to('email' ← email) with('content' ← content)
End Procedure

Function getAllPerizinan()
Begin
      Perizinan ← Select * from perizinan
      return  Perizinan
End Function
```

5.4 *Sequence Diagram* : Mengupdate Status Perizinan Pulang Menjadi Perizinan Kembali



Gambar V.4 SD-04 mengupdate status perizinan pulang menjadi perizinan kembali

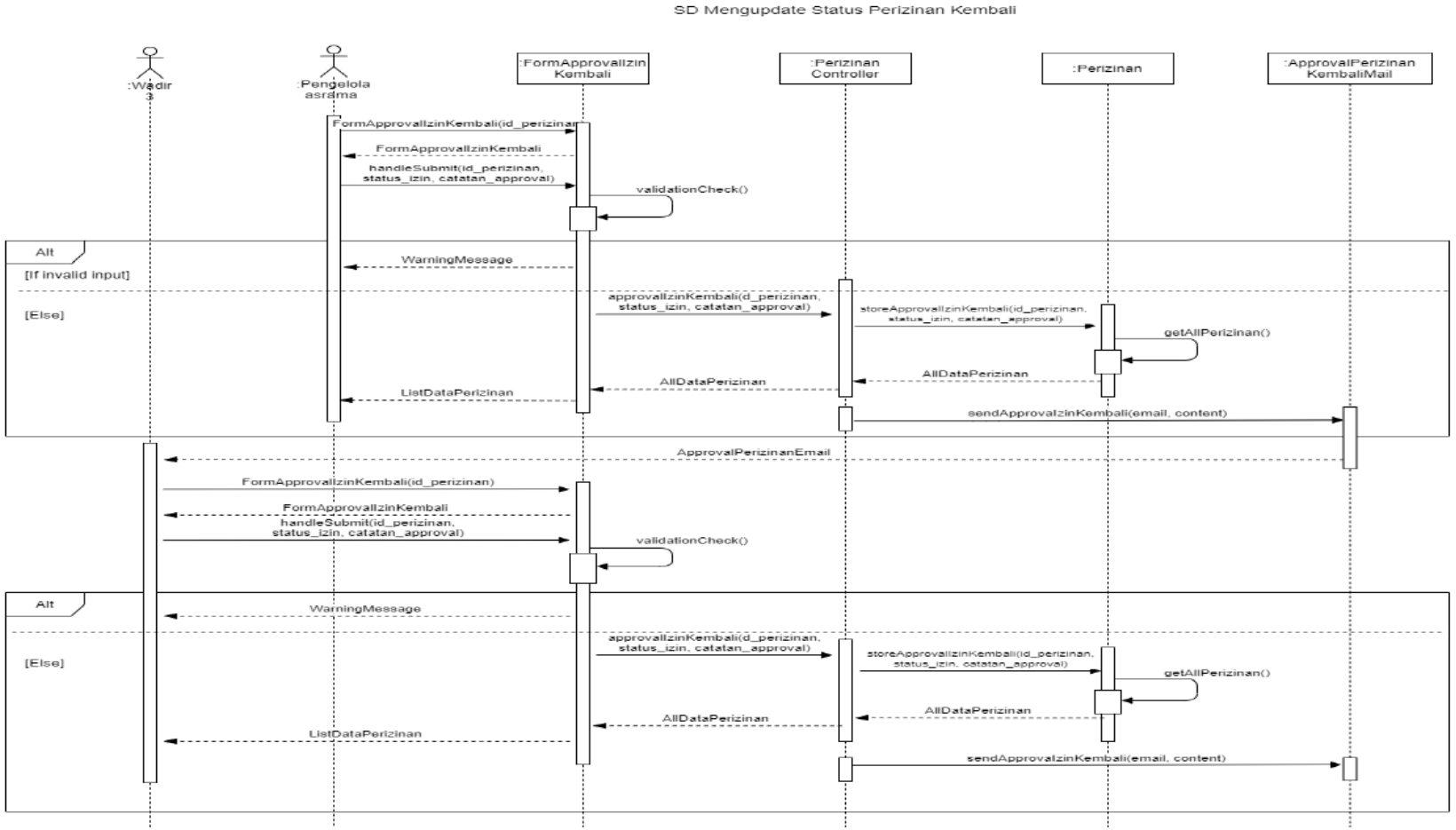
Tabel V.4 SD-04 mengupdate status perizinan pulang menjadi perizinan kembali

ID Sequence Diagram	SD-04
Nama Sequence Diagram	Mengupdate Status Perizinan Pulang Menjadi Kembali
ID Use Case	UC-04
ID Requirement	- REQ-F-13 - REQ-F-19 - REQ-F-20 - REQ-F-34
Nama Class Terkait	PerizinanController, Perizinan, dan PengajuanPerizinanPulangMail
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas mahasiswa mengajukan perizinan kembali ke asrama kepada. Mahasiswa harus mengisi form pengajuan izin kembali ke asrama melalui aplikasi dan akan dikirim kepada pengelola asrama melalui *e-mail*.
Method yang Terkait	- getPerizinanByUser(id_mhs) - showRiwayatPerizinan(id_mhs) - getDataPerizinan(id_mhs) - formIzinKembali(id_perizinan) - handleSubmit(id_perizinan, keterangan_kembali, pengajuan_tanggal_pulang, kondisi_kesehatan, suhu_badan, jenis_kendaraan) - validationCheck() - sendIzinKembali(email, content) - updateIzinKembali(id_perizinan, keterangan_kembali, pengajuan_tanggal_pulang, kondisi_kesehatan, suhu_badan, jenis_kendaraan) - storeIzinKembali(id_perizinan, keterangan_kembali, pengajuan_tanggal_pulang, kondisi_kesehatan, suhu_badan, jenis_kendaraan)
Algoritma	
Kamus Data: id_perizinan : Variabel tunggal bertipe integer id_mhs : Variabel tunggal bertipe integer pengajuan_tanggal_pulang : Variabel tunggal bertipe date keterangan_kembali : Variabel tunggal bertipe array of character validasi : Variabel komposit bertipe array request : Variabel superglobal berhubungan dengan objek permintaan HTTP response : Variabel komposit bertipe JSON Perizinan : Variabel komposit bertipe array sql : Variabel tunggal bertipe array of character JSON : fungsi untuk menyimpan dan bertukar data kondisi_kesehatan : Variabel tunggal bertipe array of character jenis_kendaraan : Variabel tunggal bertipe array of character	

```
suhu_badan : Variabel tunggal bertipe float


Procedure getPerizinanByUser(id_mhs)
Begin
     window.location.assign('getPerizinanByUser')
     request.url ← 'base_url/api/perizinan/showRiwayatPerizinan
     request.method ← 'GET'
     response ← send (request) to request.url using
request.method
     write(response.Perizinan)
End Procedure

Function showRiwayatPerizinan(id_mhs)
Begin
     Perizinan ← getDataPerizinan(id_mhs)
     return response('Perizinan' ← Perizinan
End Function

Function getDataPerizinan(id_mhs)
Begin
      Perizinan ← Select * from perizinan where id_mhs = id_mhs
      return  Perizinan
End Function

Procedure formIzinKembali(id_perizinan)
Begin
      window.location.assign('formIzinKembali')
End Procedure

Procedure handleSubmit(id_perizinan, keterangan_kembali,
pengajuan_tanggal_pulang, kondisi_kesehatan, suhu_badan,
jenis_kendaraan)
Begin
      request.body ← JSON('pengajuan_tanggal_pulang'
 ← pengajuan_tanggal_pulang, 'keterangan_kembali' ←
keterangan_kembali, 'suhu_badan' ← suhu_badan,
'kondisi_kesehatan' ← kondisi_kesehatan, 'jenis_kendaraan' ←
jenis_kendaraan)
      validasi ← validationCheck()
      if(validasi == true)
         then
           request.url ←
'base_url/api/perizinan/updateIzinKembali
           request.method ← 'POST'
           response ← send (request) to request.url using
request.method
         else
           alert('form tidak valid')
End Procedure

Function validationCheck()
Begin
 validasi ← request.body().check('id_perizinan' ← 'required',
'keterangan_kembali' ← 'required|max:125',
```

```
'pengajuan_tanggal_pulang' ← 'required', 'suhu_badan' ←
'required|numeric|between:30,50', 'kondisi_kesehatan' ←
'required|max:50', 'jenis_kendaraan' ← 'required',
)
     return validasi
End Function

Function updateIzinKembali (id_perizinan, keterangan_kembali,
pengajuan_tanggal_pulang, kondisi_kesehatan, suhu_badan,
jenis_kendaraan)
Begin
      perizinan ← storeIzinKembali (id_perizinan,
keterangan_kembali, pengajuan_tanggal_pulang, kondisi_kesehatan,
suhu_badan, jenis_kendaraan)
      sendIzinKembali(email, content)
      return response('perizinan' ← perizinan)
End Function

Procedure sendIzinKembali(email, content)
Begin
      send mail to('email' ← email) with('content' ← content)
End Procedure

Function storeIzinKembali (id_perizinan, keterangan_kembali,
pengajuan_tanggal_pulang, kondisi_kesehatan, suhu_badan,
jenis_kendaraan)
Begin
      update ← UPDATE perizinan set pengajuan_tanggal_pulang
 = pengajuan_tanggal_pulang, keterangan_kembali=
keterangan_kembali, pengajuan_tanggal_pulang =
pengajuan_tanggal_pulang, kondisi_kesehatan = kondisi_kesehatan,
suhu_badan = suhu_badan, jenis_kendaraan = jenis_kendaraan
where  id_mhs =  id_mhs
      perizinan ← getDataPerizinan(id_mhs)
      return perizinan
End Function
```

5.5 *Sequence Diagram* : Mengupdate Status Perizinan Kembali



Gambar V.5 SD-05 mengupdate status perizinan kembali

Tabel V.5 SD-05 mengupdate status perizinan kembali

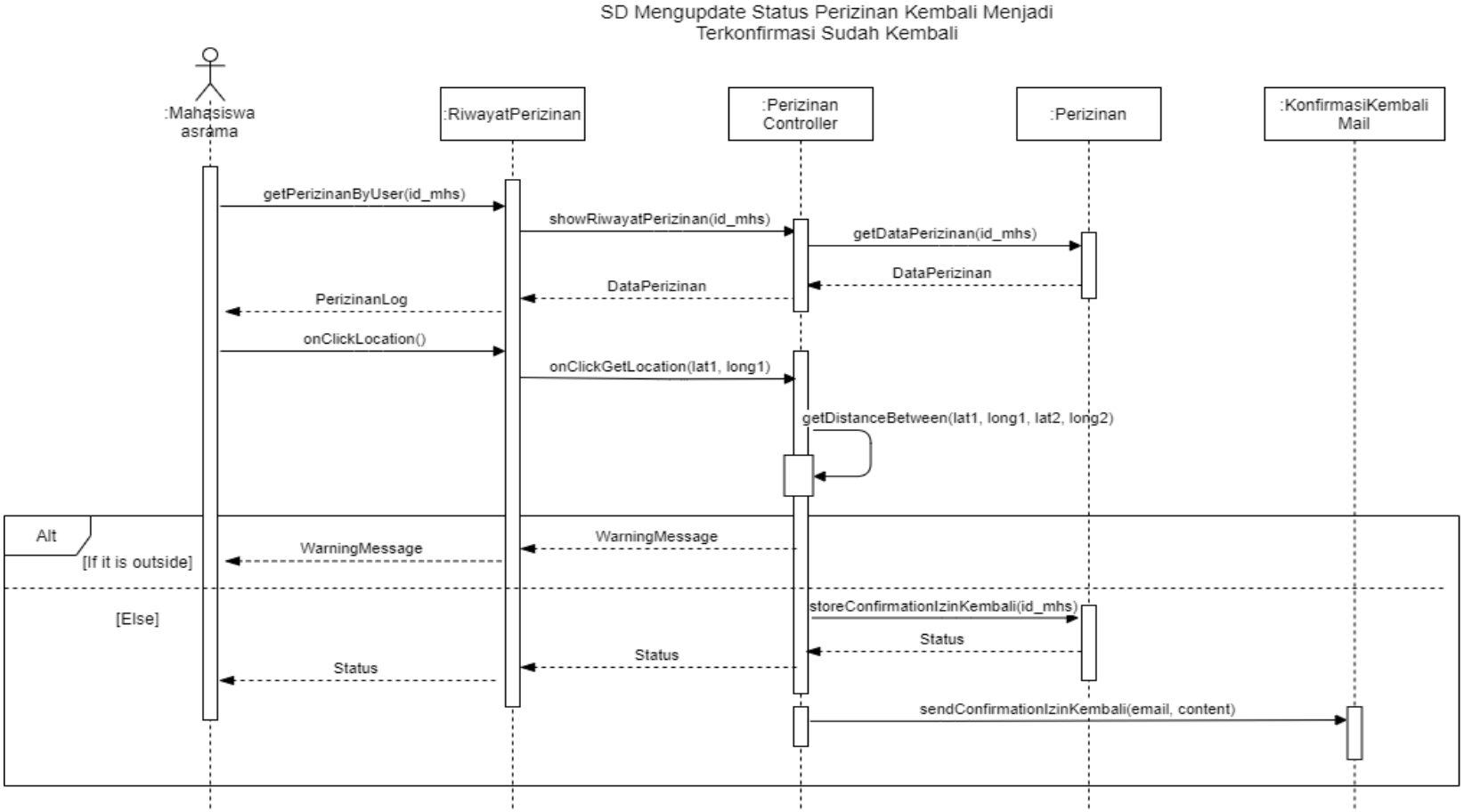
ID Sequence Diagram	SD-05
Nama Sequence Diagram	Mengupdate Status Perizinan Kembali
ID Use Case	UC-05
ID Requirement	- REQ-F-16 - REQ-F-20
Nama Class Terkait	PerizinanController, Perizinan, dan ApprovalPerizinanKembaliMail
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas pengisian *approval* perizinan kembali oleh pengelola asrama dan Wadir 3. Hasil *approval* dikirim kepada mahasiswa melalui *e-mail*.
Method yang Terkait	- formApprovalIzinKembali() - handleSubmit(id_perizinan, status_izin, catatan_approval) - validationCheck() - approvalIzinKembali(id_perizinan, status_izin, catatan_approval) - storeApprovalIzinKembali (id_perizinan, status_izin, catatan_approval) - getAllPerizinan()

Algoritma

```
Kamus Data:
id_perizinan : Variabel tunggal bertipe integer
status_izin : Variabel tunggal bertipe integer
catatan_approval : Variabel tunggal bertipe array of character
validasi : Variabel komposit bertipe array
request : Variabel superglobal berhubungan dengan objek
permintaan HTTP
response : Variabel komposit bertipe JSON
Perizinan : Variabel komposit bertipe array
sql : Variabel tunggal bertipe array of character

Procedure formApprovalIzinKembali()
Begin
      window.location.assign('formApprovalIzinKembali')
End Procedure

Procedure handleSubmit(id_perizinan, status_izin,
catatan_approval)
Begin
      request.body ← JSON('id_perizinan' ←
id_perizinan,'status_izin' ← status_izin, 'catatan_approval' ←
catatan_approval)
      validasi ← validationCheck()
      if(validasi == true)
```

```
        then
          request.url ←
'base_url/api/perizinan/approvalIzinKembali'
          request.method ← 'POST'
          response ← send (request) to request.url using
request.method
        else
          alert('form tidak valid')
End Procedure

Function validationCheck()
Begin
 validasi ← request.body().check('id_perizinan' ←
'required|number', 'status' ← 'required|number')
     return validasi
End Function

Function approvalIzinKembali(id_perizinan, status_izin,
catatan_approval)
Begin
      perizinan ←storeApprovalIzinKembali(id_perizinan,
status_izin, catatan_approval)
      return response('perizinan' ← perizinan)
End Function

Function storeApprovalIzinKembali(id_perizinan, status_izin,
catatan_approval)
Begin
      update ← Update Perizinan set  status_izin =  status_izin,
catatan_approval = catatan_approval where id_perizinan =
id_perizinan
      perizinan ← getAllPerizinan()
      return perizinan
End Function

Function getAllPerizinan()
Begin
      Perizinan ← Select * from perizinan
      return  Perizinan
End Function
```

5.6 *Sequence Diagram* : Mengupdate Status Perizinan Kembali Menjadi Terkonfirmasi Sudah Kembali



Gambar V.6 SD-06 mengupdate status perizinan kembali menjadi terkonfirmasi

Tabel V.6 SD-06 mengupdate status perizinan kembali menjadi terkonfirmasi

ID Sequence Diagram	SD-06
Nama Sequence Diagram	Mengupdate Status Perizinan Kembali Menjadi Terkonfirmasi Sudah Kembali
ID Use Case	UC-06
ID Requirement	- REQ-F-06 - REQ-F-18 - REQ-F-20
Nama Class Terkait	PerizinanController, Perizinan, dan KonfirmasiKembaliMail
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas mahasiswa melakukan konfirmasi keberadaan saat kembali ke asrama dari masa izinnya. Aplikasi melakukan validasi lokasi mahasiswa saat mahasiswa melakukan konfirmasi. Validasi dilakukan dengan menghitung koordinat mahasiswa dengan radius asrama.
Method yang Terkait	- getPerizinanByUser(id_mhs) - showRiwayatPerizinan(id_mhs) - onClickLocation() - onClickGetLocation(lat1, long1) - sendToConfirmationIzinKembali(email, content) - getDistanceBetween(lat1, long1, lat1, long2) - getDataPerizinan(id_mhs) - storeConfirmationIzinKembali(id_mhs)

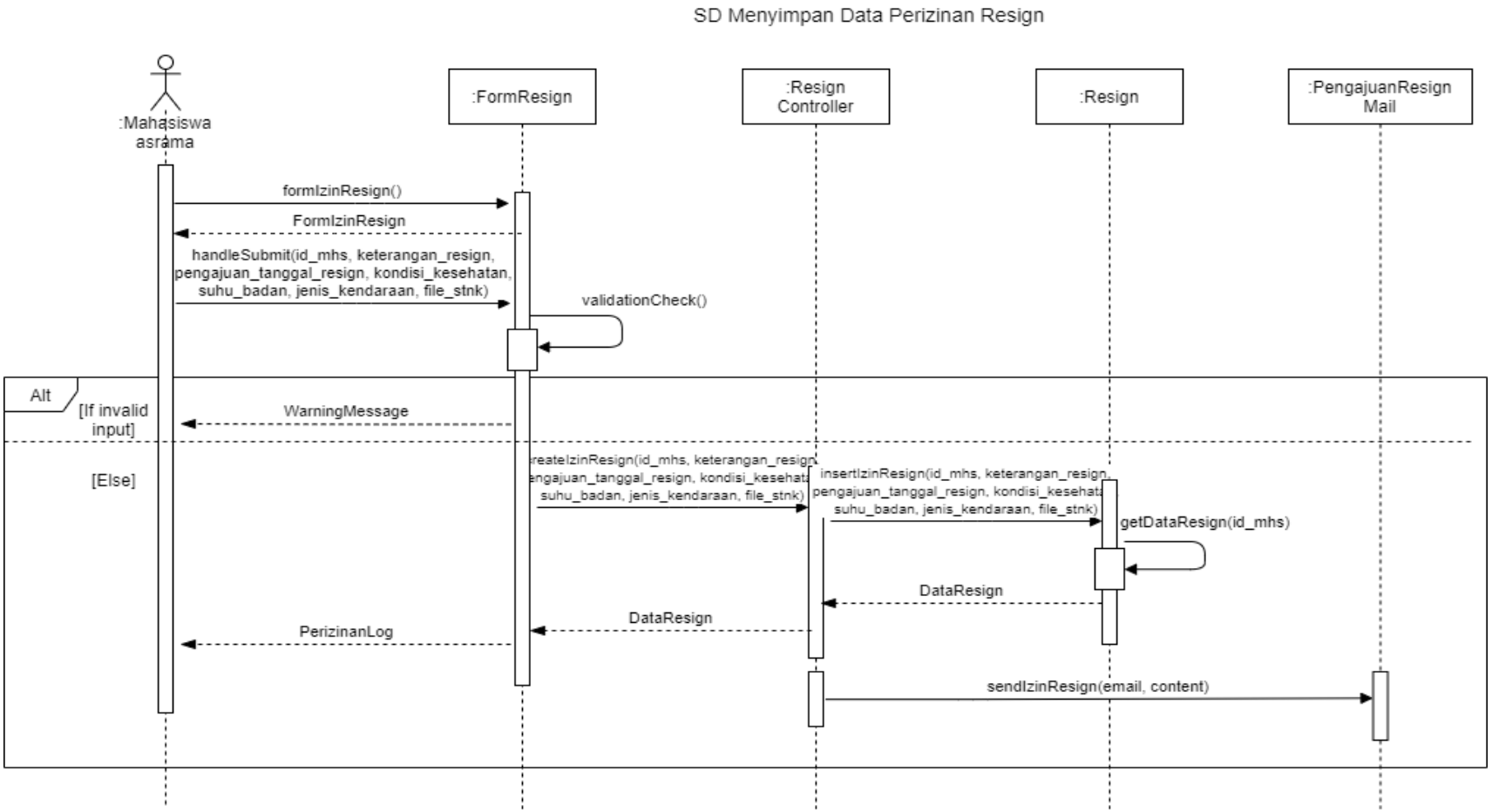
Algoritma

```
Kamus Data:
id_mhs : Variabel tunggal bertipe integer
status : Variabel tunggal bertipe integer
latitude : Variabel tunggal bertipe float
longitude : Variabel tunggal bertipe float
lat1 : Variabel tunggal bertipe float
long1 : Variabel tunggal bertipe float
lat2 : Variabel tunggal bertipe float
long2 : Variabel tunggal bertipe float
dlat : Variabel tunggal bertipe float
dlong : Variabel tunggal bertipe float
a : Variabel tunggal bertipe float
c: Variabel tunggal bertipe float
hasil : Variabel tunggal bertipe float
request : Variabel superglobal berhubungan dengan objek
permintaan HTTP
response : Variabel komposit bertipe JSON
Perizinan : Variabel komposit bertipe array
sql : Variabel tunggal bertipe array of character
```

```
Procedure getPerizinanByUser(id_mhs)
Begin
     window.location.assign('getPerizinanByUser')
     request.url ← 'base_url/api/perizinan/showRiwayatPerizinan
     request.method ← 'GET'
     response ← send (request) to request.url using
request.method
     write(response.Perizinan)
End Function

Function showRiwayatPerizinan(id_mhs)
Begin
     Perizinan ← getDataPerizinan(id_mhs)
     return response('Perizinan' ← Perizinan
End Function

Function onClickLocation()
Begin
      latitude ← getCurrentPosition().latitude
      longitude ← getCurrentPosition().longitude
      request.body ← JSON('lat1' ← latitude, 'long1' ←
longitude)
     request.url ← "base_url/api/presensi/onClickGetLocation"
     request.method ← POST
     response ← send (request) to request.url using
request.method
     return response
End Function

Function onClickGetLocation(lat1, long1)
Begin
      hasil ← getDistanceBetween(lat1, long1, -
6.8719714,107.5711026)
      if(hasil > 0.05)
        then
           alert('Sedang Tidak Dalam Asrama')
        else
           storeConfirmationIzinKembali(id_mhs)
           sendConfirmationIzinKembali(email, content)
      endIf
      return  response('success' ← true)
End Function

Procedure sendConfirmationIzinKembali(email, content)
Begin
      send mail to('email' ← email) with('content' ← content)
End Procedure

Function getDistanceBetween(lat1, long1, lat2, long2)
Begin
     dlat ← (lat2 - lat1) * 3.14/180
     dlong ← (long2 - long1) * 3.14/180
     a ← sin(dlat/2)² + sin(dlong/2)² * cos(lat1) * cos(lat2)
     rad ← 6371
     c ← 2 * asin(√a)
```

```
     hasil ← rad * c
     return hasil
End Function

Function getDataPerizinan(id mhs)
Begin
      Perizinan ← Select * from perizinan where id_mhs = id_mhs
      return  Perizinan
End Function


Procedure storeConfirmationIzinKembali(id_mhs)
Begin
      update ← UPDATE perizinan set status = status
where  id_mhs =  id_mhs
End Procedure

Function getDistanceBetween(lat1, long1, lat2, long2)
Begin
     dlat ← (lat2 - lat1) * 3.14/180
     dlong ← (long2 - long1) * 3.14/180
     a ← sin(dlat/2)² + sin(dlong/2)² * cos(lat1) * cos(lat2)
     rad ← 6371
     c ← 2 * asin(√a)
     hasil ← rad * c
     return hasil
End Function
```

5.7 *Sequence Diagram* : Menyimpan Data Perizinan *Resign*



Gambar V.7 SD-07 menyimpan data perizinan *resign*

Tabel V.7 SD-07 menyimpan data perizinan *resign*

ID Sequence Diagram	SD-07
Nama Sequence Diagram	Menyimpan Data Perizinan *Resign*
ID Use Case	UC-07
ID Requirement	- REQ-F-19 - REQ-F-20
Nama Class Terkait	ResignController, Resign, dan PengajuanResignMail
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas mahasiswa mengajukan izin *resign* melalui aplikasi. Mahasiswa harus mengisi *form* pengajuan *resign* untuk diteruskan kepada pengelola asrama melalui pengiriman *e-mail*.
Method yang Terkait	- formIzinResign() - handeSubmit(id_mhs, keterangan_resign, pengajuan_tanggal_resign, kondisi_kesehatan, suhu_badan, jenis_kendaraan, file_stnk) - validationCheck() - createIzinResign(id_mhs, keterangan_resign, pengajuan_tanggal_resign, kondisi_kesehatan, suhu_badan, jenis_kendaraan, file_stnk) - sendIzinResign(email, content) - insertIzinResign(id_mhs, keterangan_resign, pengajuan_tanggal_resign, kondisi_kesehatan, suhu_badan, jenis_kendaraan, file_stnk) - getDataResign(id_mhs)

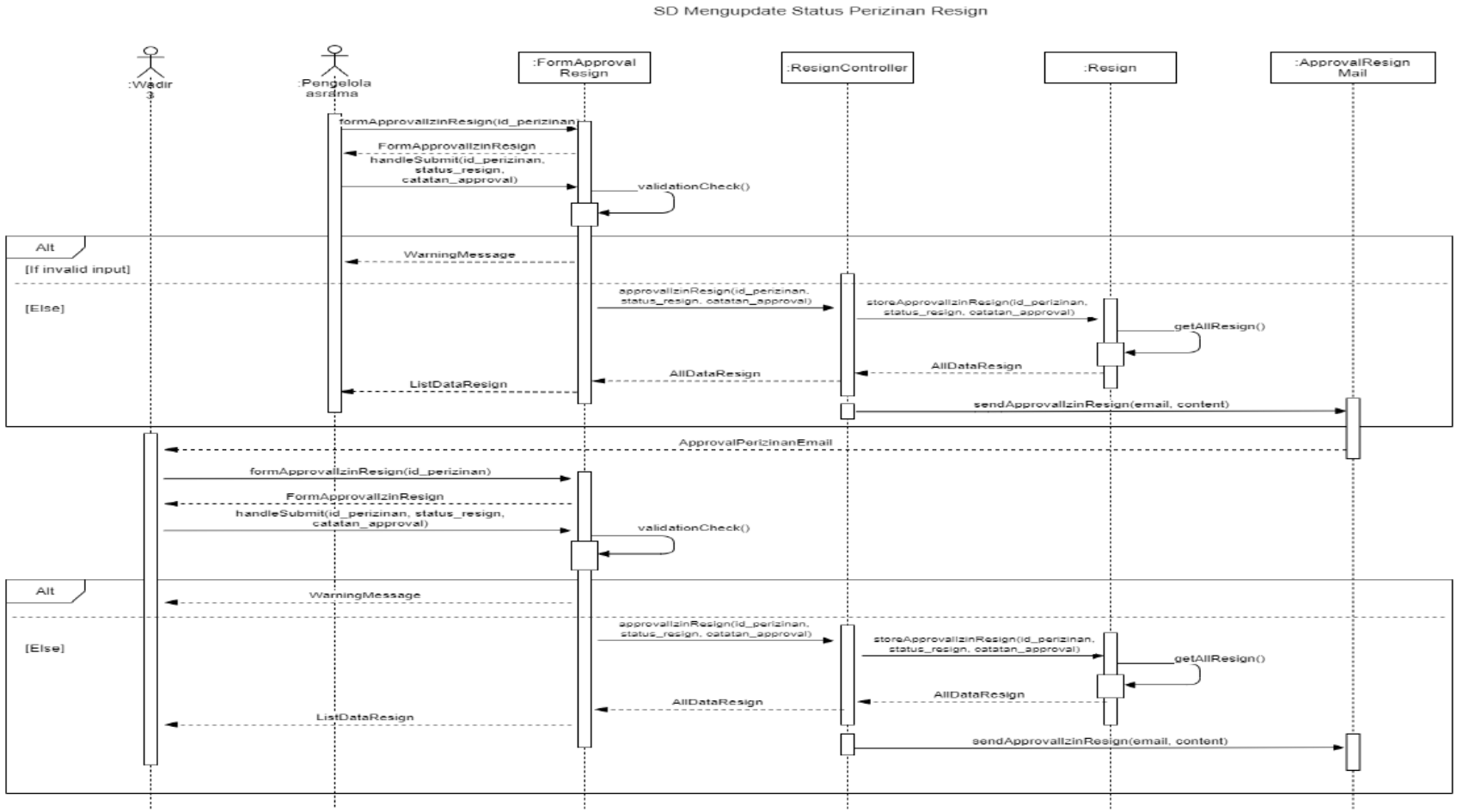
Algoritma

```
Kamus Data:
id_mhs : Variabel tunggal bertipe integer
pengajuan_tanggal_resign: Variabel tunggal bertipe date
keterangan_resign: Variabel tunggal bertipe array of char
kondisi_kesehatan: Variabel tunggal bertipe array of char
jenis_kendaraan: Variabel tunggal bertipe array of char
file_stnk : Variabel tunggal bertipe array of char
suhu_badan: Variabel tunggal bertipe float
kondisi_kesehatan: Variabel tunggal bertipe array of char
validasi : Variabel komposit bertipe array
request : Variabel superglobal berhubungan dengan objek
permintaan HTTP
response : variabel komposit bertipe JSON
sql : Variabel tunggal bertipe array of integer
JSON : fungsi untuk menyimpan dan bertukar data

Procedure formIzinResign()
Begin
      window.location.assign('formIzinResign')
End Procedure
```

```
Procedure handleSubmit(id_mhs, keterangan_resign,
pengajuan_tanggal_resign, kondisi_kesehatan, suhu_badan,
jenis_kendaraan, file_stnk)
Begin
      request.body ← JSON('id_mhs' ← id_mhs,
'pengajuan_tanggal_resign' ← pengajuan_tanggal_resign,
'keterangan_resign' ← keterangan_resign, 'kondisi_kesehatan'←
kondisi_kesehatan, 'suhu_badan' ← suhu_badan, 'jenis_kendaraan'
← jenis_kendaraan, 'file_stnk' ← file_stnk)
      validasi ← validationCheck()
      if(validasi == true)
         then
           request.url ←
'base_url/api/perizinan/createIzinResign'
           request.method ← 'POST'
           response ← send (request) to request.url using
request.method
         else
           alert('form tidak valid')
End Procedure

Function validationCheck()
Begin
      validasi ← request.body().check('id_mhs' ←
'required|number', 'tanggal_resign' ← 'required',
'keterangan_resign' ← 'required', 'jenis_kendaraan' ←
'required', 'suhu_badan' ← 'required|numeric|between:30,50',
'kondisi_kesehatan' ← 'required|max:50', 'file_stnk← '.pdf ||
.jpg || .png'|max: 2MB)
     return validasi
End Function

Function createIzinResign(id_mhs, keterangan_resign,
pengajuan_tanggal_resign, kondisi_kesehatan, suhu_badan,
jenis_kendaraan, file_stnk)
Begin
      Resign ← insertIzinResign(id_mhs, keterangan_resign,
pengajuan_tanggal_resign, kondisi_kesehatan, suhu_badan,
jenis_kendaraan, file_stnk)
      sendIzinResign(email, content)
      return response('perizinan' ← perizinan)
End Function

Procedure sendIzinResign(email, content)
Begin
      send mail to('email' ← email) with('content' ← content)
End Procedure

Function getDataResign(id_mhs)
Begin
      Resign ← Select * from resign where id_mhs = id_mhs
      return  Resign
End Function
```

```
Function insertIzinResign(id_mhs, keterangan_resign,
pengajuan_tanggal_resign, kondisi_kesehatan, suhu_badan,
jenis_kendaraan, file_stnk)
Begin
      insert ← INSERT INTO resign(id_mhs, keterangan_resign,
pengajuan_tanggal_resign, kondisi_kesehatan, suhu_badan,
jenis_kendaraan, file_stnk)
      Resign ← getDataResign(id_mhs)
      return Resign
End Function
```

5.8 *Sequence Diagram* : Mengupdate Status Perizinan *Resign*



Gambar V.8 SD-08 menyimpan data perizinan *resign*

Tabel V.8 SD-08 menyimpan data perizinan *resign*

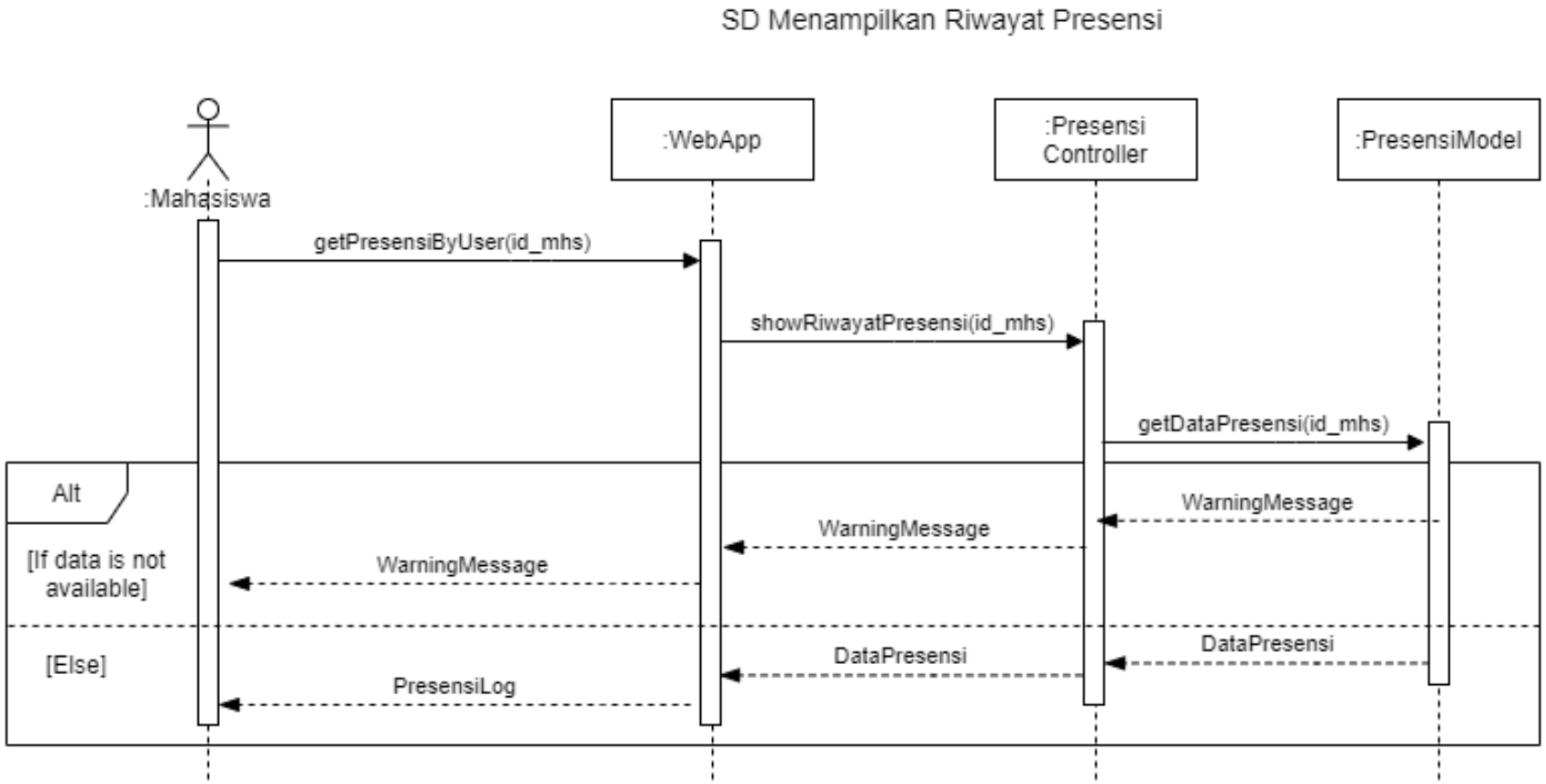
ID Sequence Diagram	SD-08
Nama Sequence Diagram	Mengupdate Status Perizinan *Resign*
ID Use Case	UC-08
ID Requirement	- REQ-F-17 - REQ-F-20
Nama Class Terkait	ResignController, Resign, dan ApprovalResignMail
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas pengisian *approval* perizinan *resign* oleh pengelola asrama dan Wadir 3. Hasil *approval* dikirim kepada mahasiswa melalui *e-mail*.
Method yang Terkait	- formApprovalIzinResign(id_perizinan) - handleSubmit(id_perizinan, status_resign, catatan_approval) - validationCheck() - approvalIzinPulang(id_perizinan, status_resign, catatan_approval) - sendApprovalIzinResign(email, content) - storeApprovalIzinResign(id_perizinan, status_resign, catatan_approval) - getAllResign()

Algoritma

```
Kamus Data:
id_perizinan : Variabel tunggal bertipe array of integer
status_resign : Variabel tunggal bertipe integer
catatan_approval : Variabel tunggal bertipe array of char
request : Variabel superglobal berhubungan dengan objek
permintaan HTTP
response : Variabel komposit bertipe JSON
Resign : Variabel komposit bertipe array
validasi : Variabel komposit bertipe array
sql : Variabel tunggal bertipe array of integer
JSON : fungsi untuk menyimpan dan bertukar data
Procedure formApprovalIzinResign(id_perizinan)
Begin
      window.location.assign('formApprovalIzinResign')
End Procedure

Procedure handleSubmit(id_perizinan, status_resign,
catatan_approval)
Begin
      request.body ← JSON('id_perizinan' ← id_perizinan,
'status_resign' ← status_resign, 'catatan_approval' ←
catatan_approval)
      validasi ← validationCheck()
      if(validasi == true)
```

```
         then
           request.url ←
'base_url/api/perizinan/approvalIzinResign
           request.method ← 'POST'
           response ← send (request) to request.url using
request.method
         else
           alert('form tidak valid')
End Procedure

Function validationCheck()
Begin
      validasi ← request.body().check('id_mhs' ←
'required|number')
     return validasi
End Function

Function approvalIzinPulang(id_perizinan, status_resign,
catatan_approval)
Begin
      Resign ← storeApprovalIzinResign(id_perizinan,
status_resign, catatan_approval)
      sendApprovalIzinResign(email, content)
      return response('perizinan' ← perizinan)
End Function

Procedure sendApprovalIzinResign(email, content)
Begin
      send mail to('email' ← email) with('content' ← content)
End Procedure


Function getAllResign()
Begin
      Resign ← Select * from resign where status_resign == 0,
      return  Resign
End Function


Procedure  storeApprovalIzinResign(id_perizinan, status_resign,
catatan_approval)
Begin
      update ← UPDATE perizinan set  status_resign
=  status_resign, catatan_approval = catatan_approval where
id_perizinan = id_perizinan
      Resign ← getAllResign()
      return Resign
End Procedure
```

5.9 *Sequence Diagram* : Menampilkan Riwayat Presensi



Gambar V.9 SD-09 menampilkan riwayat presensi

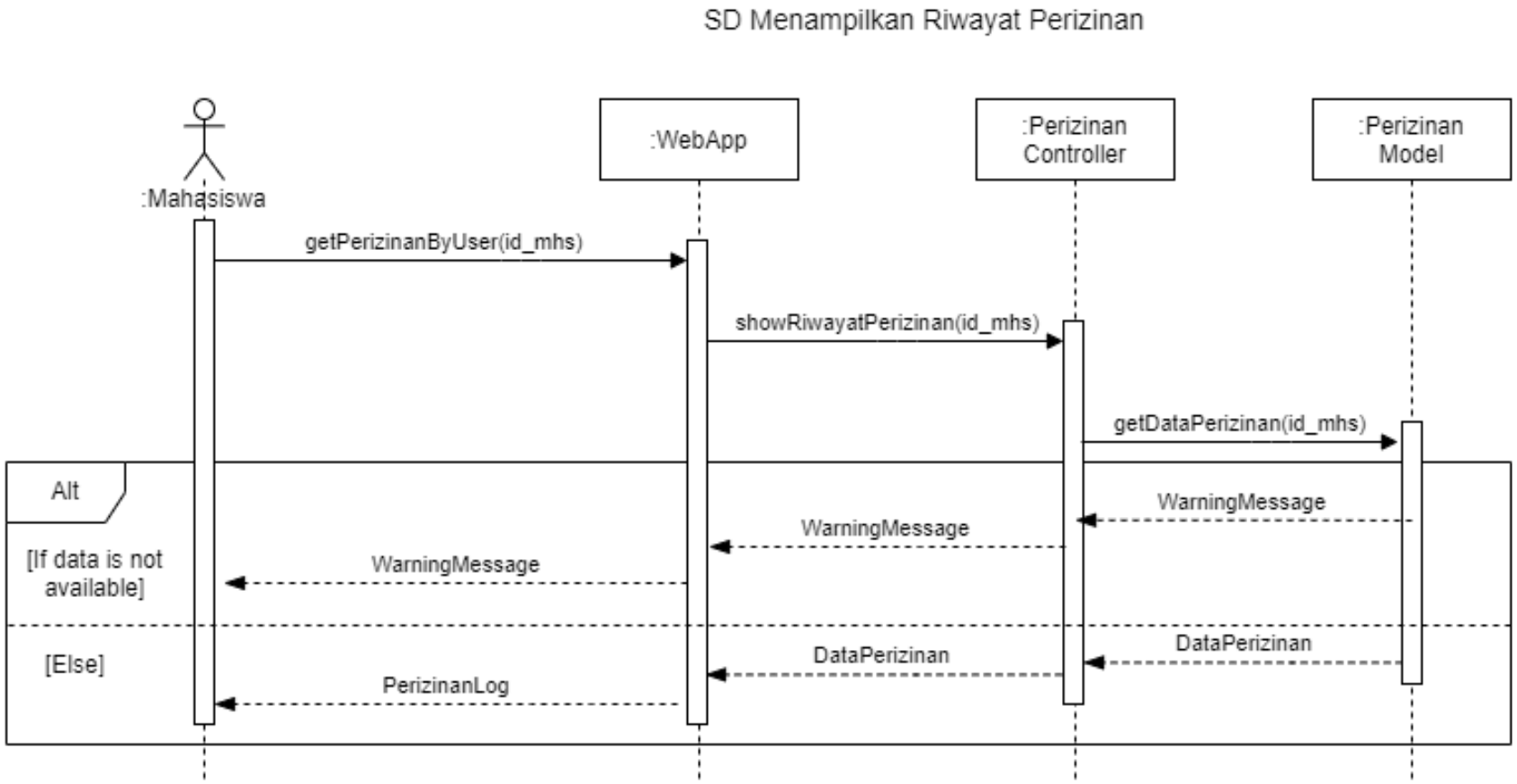
Tabel V.9 SD-09 menampilkan riwayat presensi

ID Sequence Diagram	SD-09
Nama Sequence Diagram	Menampilkan Riwayat Presensi
ID Use Case	UC-09
ID Requirement	- REQ-F-09
Nama Class Terkait	PresensiController dan Presensi
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas mahasiswa dalam mengakses riwayat presensi pribadi melalui aplikasi.
Method yang Terkait	- getPresensiByUser(id_mhs) - showRiwayatPerizinan(id_mhs) - getDataPresensi(id_mhs)

Algoritma

```
Kamus Data:
id_mhs : Variabel tunggal bertipe integer
request : Variabel superglobal berhubungan dengan objek
permintaan HTTP
response : Variabel komposit bertipe JSON
Presensi : Variabel komposit bertipe array

Procedure getPresensiByUser(id_mhs)
Begin
     window.location.assign('getPresensiByUser')
     request.url ← 'base_url/api/presensi/showRiwayatPresensi
     request.method ← 'GET'
     response ← send (request) to request.url using
request.method
     write(response.Presensi)
End Procedure

Function showRiwayatPerizinan(id_mhs)
Begin
     Presensi ← getDataPresensi(id_mhs)
     return response('Presensi' ← Presensi
End Function


Function getDataPresensi(id_mhs)
Begin
      Presensi ← Select * from presensi where id_mhs = id_mhs
      return  Presensi
End Function
```

5.10 *Sequence Diagram* : Menampilkan Riwayat Perizinan



Gambar V.10 SD-10 menampilkan riwayat perizinan

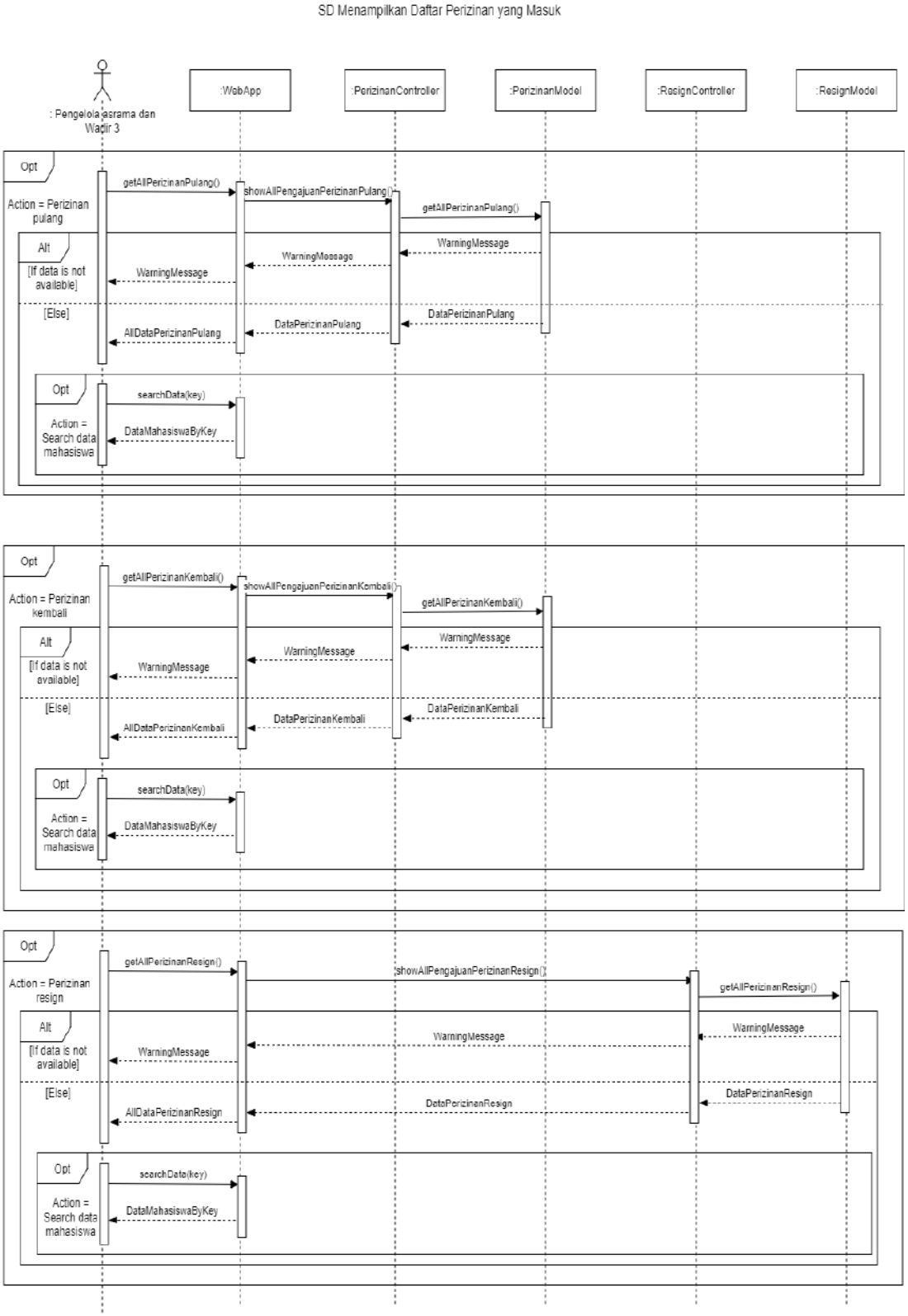
Tabel V.10 SD-10 menampilkan riwayat perizinan

ID Sequence Diagram	SD-10
Nama Sequence Diagram	Menampilkan Riwayat Perizinan
ID Use Case	UC-10
ID Requirement	- REQ-F-19
Nama Class Terkait	PerizinanController dan Perizinan
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas mahasiswa dalam mengakses riwayat perizinan pribadi melalui aplikasi.
Method yang Terkait	- getPerizinanByUser(id_mhs) - showRiwayatPerizinan(id_mhs) - getDataPerizinan(id_mhs)

Algoritma

```
Kamus Data:
id_mhs : Variabel tunggal bertipe integer
request : Variabel superglobal berhubungan dengan objek
permintaan HTTP
response : Variabel komposit bertipe JSON
Perizinan : Variabel komposit bertipe array

Procedure getPerizinanByUser(id_mhs)
Begin
     window.location.assign('getPerizinanByUser')
     request.url ← 'base_url/api/perizinan/showRiwayatPerizinan
     request.method ← 'GET'
     response ← send (request) to request.url using
request.method
     write(response.Perizinan)
End Procedure

Function showRiwayatPerizinan(id_mhs)
Begin
     Perizinan ← getDataPerizinan(id_mhs)
     return response('Perizinan' ← Perizinan
End Function

Function getDataPerizinan(id_mhs)
Begin
      Perizinan ← Select * from perizinan where id_mhs = id_mhs
      return  Perizinan
End Function
```

5.11 *Sequence Diagram* : Menampilkan Daftar Perizinan yang Masuk



Gambar V.11 SD-11 menampilkan daftar perizinan yang masuk

Tabel V.11 SD-11 menampilkan daftar perizinan yang masuk

ID Sequence Diagram	SD-11
Nama Sequence Diagram	Menampilkan Daftar Perizinan yang Masuk
ID Use Case	UC-11
ID Requirement	- REQ-F-21 - REQ-F-22
Nama Class Terkait	PerizinanController, ResignController, Perizinan, dan Resign
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas pengelola asrama dan Wadir 3 dalam mengakses daftar perizinan yang masuk.
Method yang Terkait	- getAllPerizinanPulang() - showAllPengajuanPerizinanPulang() - getAllPerizinanPulang() - getAllPerizinanKembali() - showAllPengajuanPerizinanKembali() - getAllPerizinanKembali() - getAllPerizinanResign() - showAllPengajuanPerizinanResign() - getAllPerizinanResign() - searchData(key)

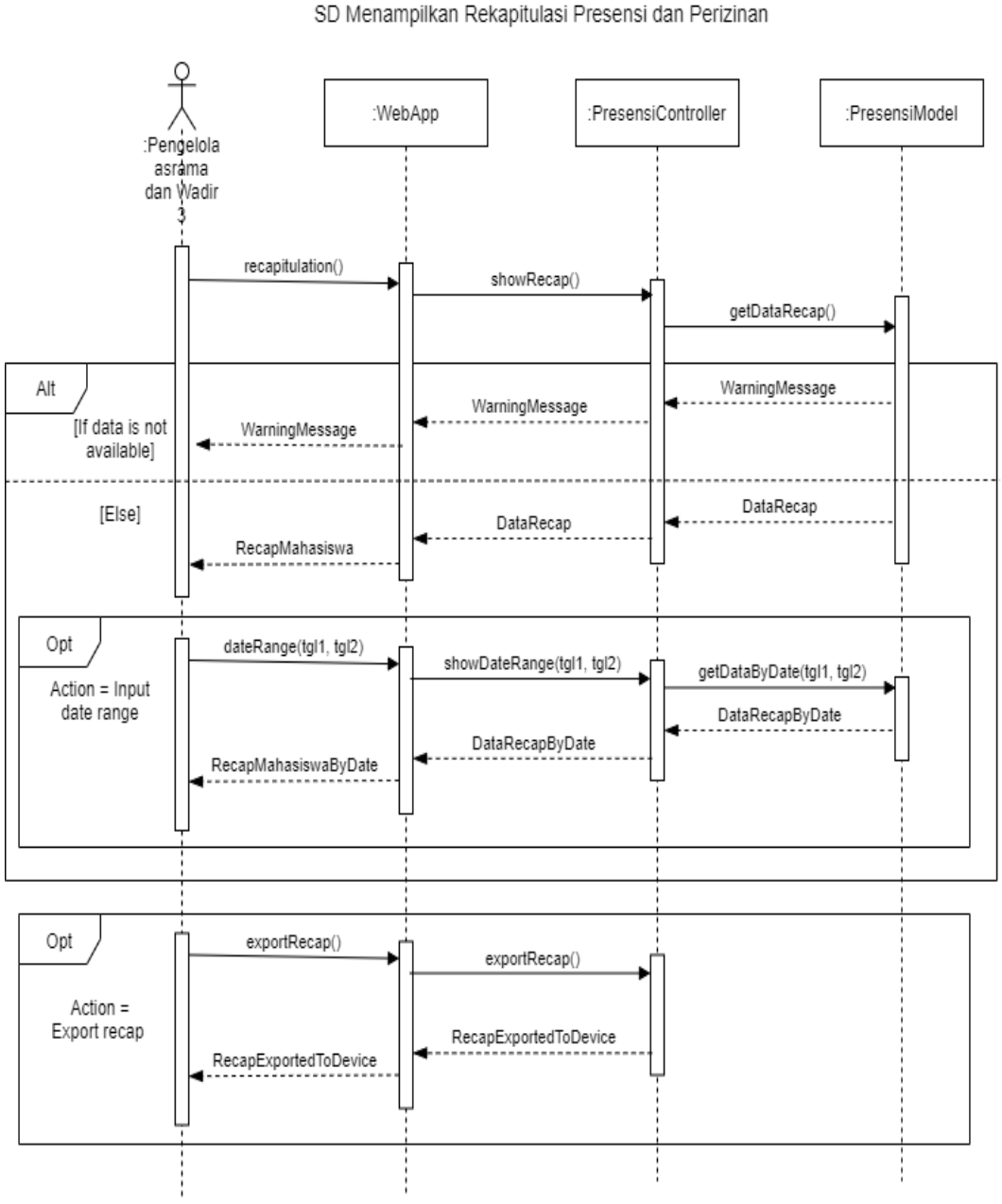
Algoritma

```
Kamus Data:
request : Variabel superglobal berhubungan dengan objek
permintaan HTTP
response : Variabel komposit bertipe JSON
Perizinan : Variabel komposit bertipe array
PerizinanPulang : Variabel komposit bertipe array
PerizinanKembali : Variabel komposit bertipe array
status_izin : Variabel tunggal bertipe integer
status_resign : Variabel tunggal bertipe integer
key : Variabel tunggal bertipe array of character

Procedure getAllPerizinanPulang()
Begin
     request.url ←
'base_url/api/perizinan/showAllPengajuanPerizinanPulang
     request.method ← 'GET'
     response ← send (request) to request.url using
request.method
     write(response.Perizinan)
End Procedure

Function showAllPengajuanPerizinanPulang()
Begin
     Perizinan ← getAllPerizinanPulang()
```

```
     return response('Perizinan' ← Perizinan
End Function

Function getAllPerizinanPulang()
Begin
      PerizinanPulang ← Select * from perizinan where
status_izin == 0 OR status_izin == 1
      return  PerizinanPulang
End Function

Procedure getAllPerizinanKembali()
Begin
     request.url ←
'base_url/api/perizinan/showAllPengajuanPerizinanKembali
     request.method ← 'GET'
     response ← send (request) to request.url using
request.method
     write(response.Perizinan)
End Procedure

Function showAllPengajuanPerizinanKembali()
Begin
     Perizinan ← getAllPerizinanKembali()
     return response('Perizinan' ← Perizinan
End Function

Function getAllPerizinanKembali()
Begin
      PerizinanKembali ← Select * from perizinan status_izin ==
5 OR status_izin == 6
      return  PerizinanKembali
End Function

Procedure getAllPerizinanResign()
Begin
     request.url ←
'base_url/api/perizinan/showAllPengajuanPerizinanResign
     request.method ← 'GET'
     response ← send (request) to request.url using
request.method
     write(response.Perizinan)
End Procedure

Function showAllPengajuanPerizinanResign()
Begin
     Perizinan ← getAllPerizinanResign()
     return response('Perizinan' ← Perizinan
End Function

Function getAllPerizinanResign()
Begin
      PerizinanResign ← Select * from resign where status_resign
== 0
      return  PerizinanResign
End Function
```

```
Procedure searchData(key)
Begin
      For each data Perizinan
          if(Perizinan == key)
              then write(Perizinan)
          end If
      End For
End Procedure
```

5.12 *Sequence Diagram* : Menampilkan Rekapitulasi Presensi dan Perizinan



Gambar V.12 SD-12 menampilkan rekapitulasi presensi dan perizinan

Tabel V.12 SD-12 menampikan rekapitulasi presensi dan perizinan

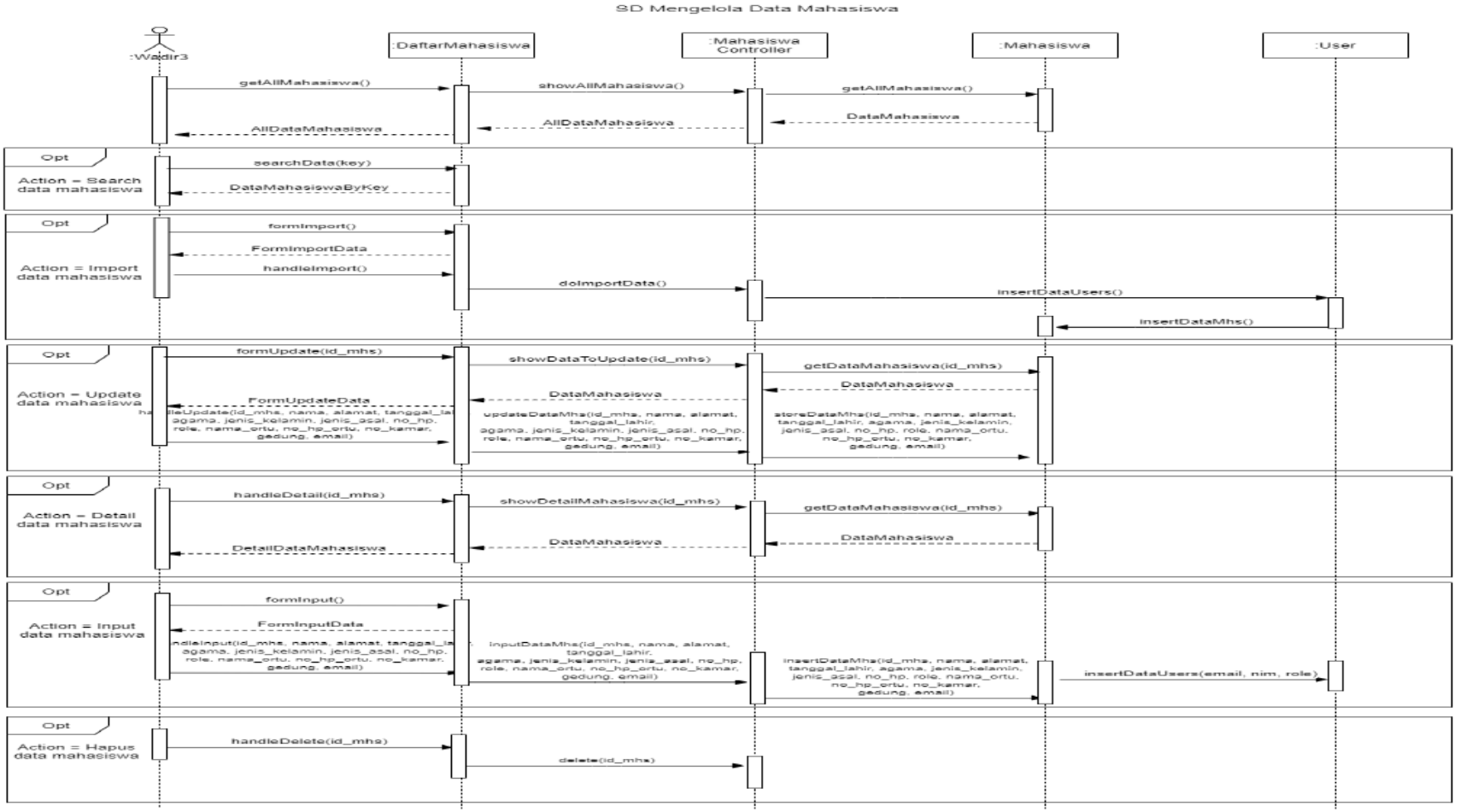
ID Sequence Diagram	SD-12
Nama Sequence Diagram	Menampilkan Rekapitulasi Presensi dan Perizinan
ID Use Case	UC-12
ID Requirement	- REQ-F-29 - REQ-F-30 - REQ-F-31
Nama Class Terkait	PresensiController dan Presensi
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas pengelola asrama dan Wadir 3 dalam mengakses data rekapitulasi presensi dan perizinan.
Method yang Terkait	- recapitulation() - showRecap() - dateRange(tgl1, tgl2) - showDateRange(tgl1, tgl2) - getDataRecap() - getDataByDate(tgl1, tgl2) - exportRecap()

Algoritma

```
Kamus Data:
tgl1 : Variabel tunggal bertipe date
tgl2 : Variabel tunggal bertipe date
Presensi : Variabel komposit bertipe array
Perizinan : Variabel komposit bertipe array
Recap : Variabel komposit bertipe array
Resign : Variabel komposit bertipe array
request : Variabel superglobal berhubungan dengan objek
permintaan HTTP
response : Variabel komposit bertipe JSON

Procedure recapitulation()
Begin
     request.url ← 'base_url/api/recap/showRecap
     request.method ← 'GET'
     response ← send (request) to request.url using
request.method
     write(response.Recap)
End Procedure

Function showRecap()
Begin
     Recap ← getDataRecap
     return Recap
End Function

Procedure dateRange(tgl1, tgl2)
```

```
Begin
     request.url ← 'base_url/api/recap/showDateRange
     request.method ← 'GET'
     response ← send (request) to request.url using
request.method
     write(response.Recap)
End Procedure

Function showDateRange(tgl1, tgl2)
Begin
     Recap ← getDataByDate(tgl1, tgl2)
     return Recap
End Function

Function getDataRecap()
Begin
      Presensi ← Select * from Presensi
      Perizinan ← Select * from Perizinan
      Resign ← Select * from Resign
      return  (Presensi, Perizinan, Resign)
End Function


Function getDataByDate(tgl1, tgl2)
Begin
      Presensi ← Select * from presensi where tgl1 < tanggal and
tgl2 > tanggal
      Perizinan ← Select * from perizinan where tgl1 < tanggal
and tgl2 > tanggal
      Resign ← Select * from resign where tgl1 < tanggal and
tgl2 > tanggal
      return  (Presensi, Perizinan, Resign)
End Function


Procedure exportRecap()
Begin
     download(Recap as .pdf)
End Procedure
```

5.13 *Sequence Diagram* : Mengelola Data Mahasiswa



Gambar V.13 SD-13 mengelola data mahasiswa

Tabel V.13 SD-13 mengelola data mahasiswa

ID Sequence Diagram	SD-13
Nama Sequence Diagram	Mengelola Data Mahasiswa
ID Use Case	UC-13
ID Requirement	- REQ-F-23 - REQ-F-24 - REQ-F-25 - REQ-F-26 - REQ-F-27 - REQ-F-28 - REQ-F-33 - REQ-F-37
Nama Class Terkait	MahasiswaController, Mahasiswa, dan User
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas yang dilakukan Wadir 3 untuk melakukan pengelolaan data mahasiswa. Pengelolaan meliputi tambah, ubah, lihat detail, dan *import* data mahasiswa.
Method yang Terkait	- getAllMahasiswa() - showAllMahasiswa() - getAllMahasiswa() - searchData(key) - form*Import*() - handle*Import*() - do*Import*Data() - formUpdate(id_mhs) - showDataToUpdate(id_mhs) - getDataMahasiswa(id_mhs) - handleUpdate(id_mhs, nama, alamat, tanggal_lahir, agama, jenis_kelamin, jenis_asal, no_hp, role, nama_ortu, no_hp_ortu, no_kamar, gedung, email) handleDetail(id_mhs) - showDetailMahasiswa(id_mhs) - updateDataMhs(Id_mhs) - storeDataMhs(id_mhs) - formInput() - handleInput(id_mhs,nama, alamat, tanggal_lahir, nim, prodi, jurusan, agama, jenis_kelamin, jenis_asal, no_hp, role, nama_ortu, no_hp_ortu, no_kamar, gedung, email) - inputDataMhs(id_mhs) - insertDataMhs(id_mhs) - insertDataUsers(id_users) - handleDelete(id_mhs) - delete(id_mhs)
Algoritma	

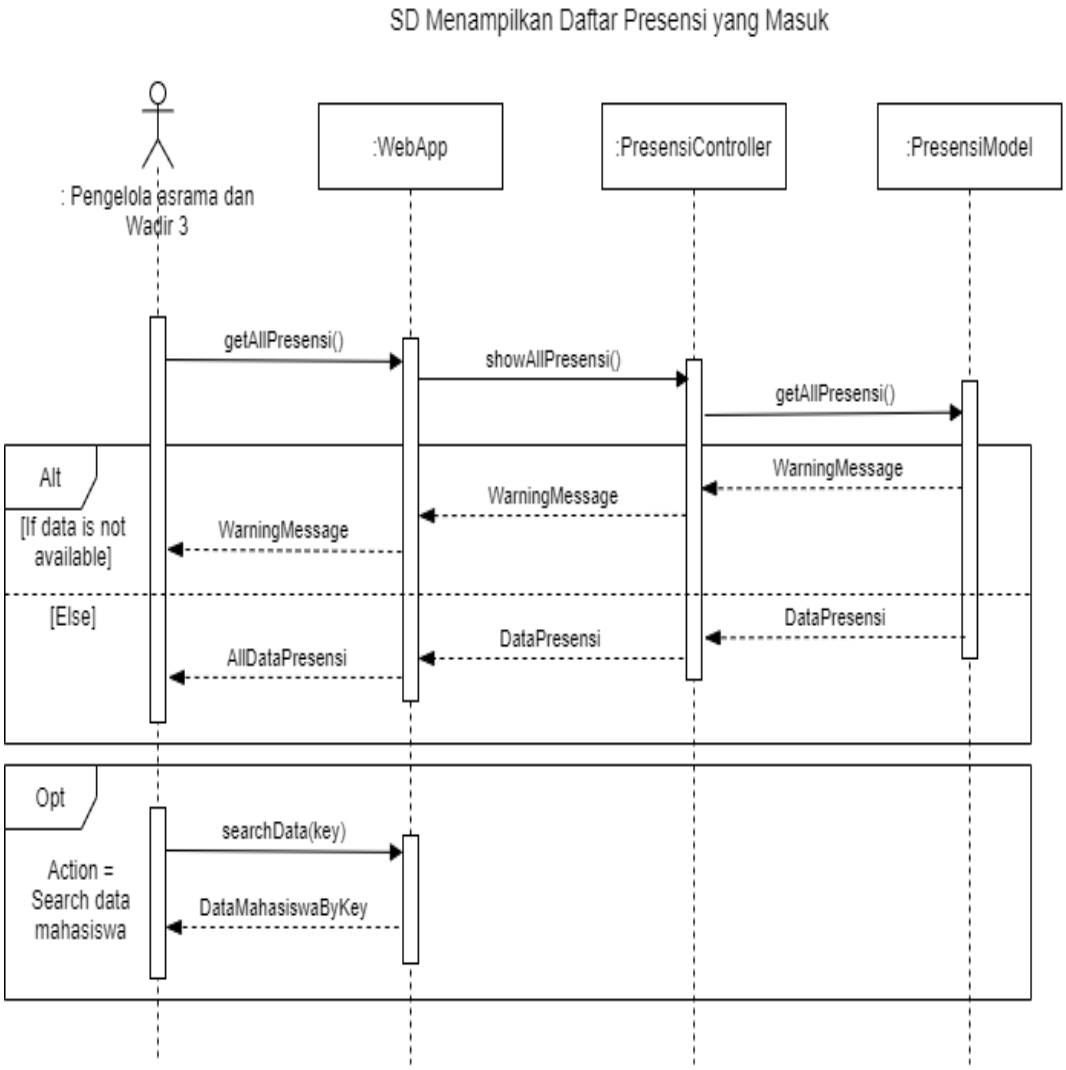
```
Kamus Data:
id_mhs : Variabel tunggal bertipe integer
id_users : Variabel tunggal bertipe integer
nama_mhs : Variabel tunggal bertipe array of character
nim : Variabel tunggal bertipe array of character
alamat : Variabel tunggal bertipe array of character
no_hp_mhs : Variabel tunggal bertipe array of character
nama_ortu : Variabel tunggal bertipe array of character
no_hp_ortu : Variabel tunggal bertipe array of character
jenis_kelamin : Variabel tunggal bertipe integer
tanggal_lahir : Variabel tunggal bertipe date
agama : Variabel tunggal bertipe array of character
keterangan_asal : Variabel tunggal bertipe array of character
request : Variabel superglobal berhubungan dengan objek
permintaan HTTP
response : Variabel komposit bertipe JSON
Mahasiswa : Variabel komposit bertipe array
sql : Variabel tunggal bertipe array of character
role: Variabel tunggal bertipe array of character
email : Variabel tunggal bertipe array of character
gedung : Variabel tunggal bertipe character
password : Variabel tunggal bertipe array of character
listMahasiswa : Variabel komposit bertipe array
count : Method untuk menghitung data yang terdapat pada array
delete : Variabel komposit bertipe array
delete() : Method untuk menghapus data
status : Variabel tunggal bertipe array of character
message : Variabel tunggal bertipe array of character
data : Variabel komposit bertipe array
i : Variabel tunggal bertipe integer

Procedure getAllMahasiswa()
Begin
     window.location.assign('getAllMahasiswa')
     request.url ← 'base_url/api/mahasiswa/showAllMahasiswa
     request.method ← 'GET'
     response ← send (request) to request.url using
request.method
     write(response.Mahasiswa)
End Procedure

Function showAllMahasiswa()
Begin
     Mahasiswa ← getAllMahasiswa()
     return response('Mahasiswa' ← Mahasiswa)
End Function

Function getAllMahasiswa()
Begin
      Mahasiswa ← Select * from mahasiswa
      return  Mahasiswa
End Function

Procedure searchData(key)
Begin
```

```
      For each data Mahasiswa
          if(Mahasiswa == key)
             then write(Mahasiswa)
          end If
      End For
End Procedure

Procedure formImport()
Begin
      window.location.assign('formImport')
End Procedure

Procedure handleImport()
Begin
     request.body('file_Import')
     request.url ← 'base_url/api/mahasiswa/doImportData
     request.method ← 'POST'
     response ← send (request) to request.url using
request.method
End Procedure

Procedure doImportData()
Begin
      For each row file_import
          insertDataMhs()
      End For

End Procedure

Procedure formUpdate(id_mhs)
Begin
      window.location.assig('formUpdate')
      request.url ← 'base_url/api/mahasiswa/showDataToUpdate
      request.method ← 'GET'
      response ← send (request) to request.url using
request.method
End Procedure

Function showDataToUpdate(id_mhs)
Begin
     Mahasiswa ← getDataMahasiswa(id_mhs)
     return response('Mahasiswa' ← Mahasiswa)
End Function

Function getDataMahasiswa(id_mhs)
Begin
      Mahasiswa ← Select * from mahasiswa where id_mhs = id_mhs
      return  Mahasiswa
End Function

Procedure handleUpdate(id_mhs, nama, alamat, tanggal_lahir,
agama, jenis_kelamin, jenis_asal, no_hp, role, nama_ortu,
no_hp_ortu, no_kamar, gedung, email)
Begin
      request.body('id_mhs' ← id_mhs)
```

```
      request.url ← 'base_url/api/mahasiswa/updateDataMhs
      request.method ← 'POST'
      response ← send (request) to request.url using
request.method
End Procedure

Procedure handleDetail(id_mhs)
Begin
      request.url ← 'base_url/api/mahasiswa/showDetailMahasiswa
      request.method ← 'GET'
      response ← send (request) to request.url using
request.method
      write(Mahasiswa)
End Procedure

Function showDetailMahasiswa(id_mhs)
Begin
      Mahasiswa ← getDataMahasiswa(id_mhs)
      return response('Mahasiswa' ← Mahasiswa)
End Function

Procedure updateDataMhs(id_mhs, nama, alamat, tanggal_lahir,
agama, jenis_kelamin, jenis_asal, no_hp, role, nama_ortu,
no_hp_ortu, no_kamar, gedung, email)
Begin
      update ← storeDataMhs(id_mhs)
End Procedure

Procedure  storeDataMhs(id_mhs, nama, alamat, tanggal_lahir,
agama, jenis_kelamin, jenis_asal, no_hp, role, nama_ortu,
no_hp_ortu, no_kamar, gedung, email)
Begin
      insert ← INSERT INTO mahasiswa(nama_mhs, nim, alamat,
no_hp_mhs, nama_ortu, no_hp_ortu, jenis_kelamin, tanggal_lahir,
agama, keterangan_asal)
End Procedure

Procedure formInput()
Begin
      window.location.assign('formInput')
End Procedure

Procedure handleInput(id_mhs, nama, alamat, tanggal_lahir,
agama, jenis_kelamin, jenis_asal, no_hp, role, nama_ortu,
no_hp_ortu, no_kamar, gedung, email)
Begin
      request.body('id_mhs' ← id_mhs)
      request.url ← 'base_url/api/mahasiswa/inputDataMhs
      request.method ← 'POST'
      response ← send (request) to request.url using
request.method
End Procedure
```

```
Procedure inputDataMhs(id_mhs, nama, alamat, tanggal_lahir,
agama, jenis_kelamin, jenis_asal, no_hp, role, nama_ortu,
no_hp_ortu, no_kamar, gedung, email)
Begin
      Mahasiswa ← insertDataMhs(id_mhs, nama, alamat,
tanggal_lahir, agama, jenis_kelamin, jenis_asal, no_hp, role,
nama_ortu, no_hp_ortu, no_kamar, gedung, email)
End Procedure

Procedure insertDataMhs(id_mhs, nama, alamat, tanggal_lahir,
agama, jenis_kelamin, jenis_asal, no_hp, role, nama_ortu,
no_hp_ortu, no_kamar, gedung, email)
Begin
      insert ← INSERT INTO mahasiswa(nama, alamat,
tanggal_lahir, agama, jenis_kelamin, jenis_asal, no_hp, role,
nama_ortu, no_hp_ortu, no_kamar, gedung, email)
      insertDataUsers(email, nim, role)
End Procedure

Procedure  insertDataUsers(email, nim, role)
Begin
     password ← nim
     insert ← INSERT INTO users(email, password, role)
End Procedure

Procedure handleDelete(id_mhs)
Begin
      request.url ← 'base_url/api/mahasiswa/delete
      request.method ← 'DELETE'
      response ← send (request) to request.url using
request.method
End Procedure

Function delete(id_mhs)
Begin
listMahasiswa ← request.listMhs;
        for(i←0; i < count(listMahasiswa); i++){
            delete ← Mahasiswa Where([
                ['id_mhs' ==  listMahasiswa[i]]) delete();
        }
        if(delete){
            return response.json([
                status ← 'success',
                message ← 'Data mahasiswa berhasil dihapus',
                data ← delete()
            ]);
        }
        else{
            return response.json([
                status ← 'error',
                message ← 'Data mahasiswa gagal dihapus',
            ]);
End Procedure
```

5.14 *Sequence Diagram* : Menampilkan Daftar Presensi yang Masuk



Gambar V.14 SD-14 menampilkan daftar presensi yang masuk

Tabel V.14 SD-14 menampilkan daftar presensi yang masuk

ID Sequence Diagram	SD-14
Nama Sequence Diagram	Menampilkan Daftar Presensi yang Masuk
ID Use Case	UC-14
ID Requirement	- REQ-F-22 - REQ-F-32
Nama Class Terkait	PresensiController, Presensi
Deskripsi	*Sequence diagram* ini berfungsi untuk menjelaskan proses aplikasi menerima aktivitas pengelola asrama dan Wadir 3 dalam mengakses daftar presensi yang masuk.
Method yang Terkait	- getAllPresensi() - showAllPresensi() - getAllPresensi() - searchData(key)

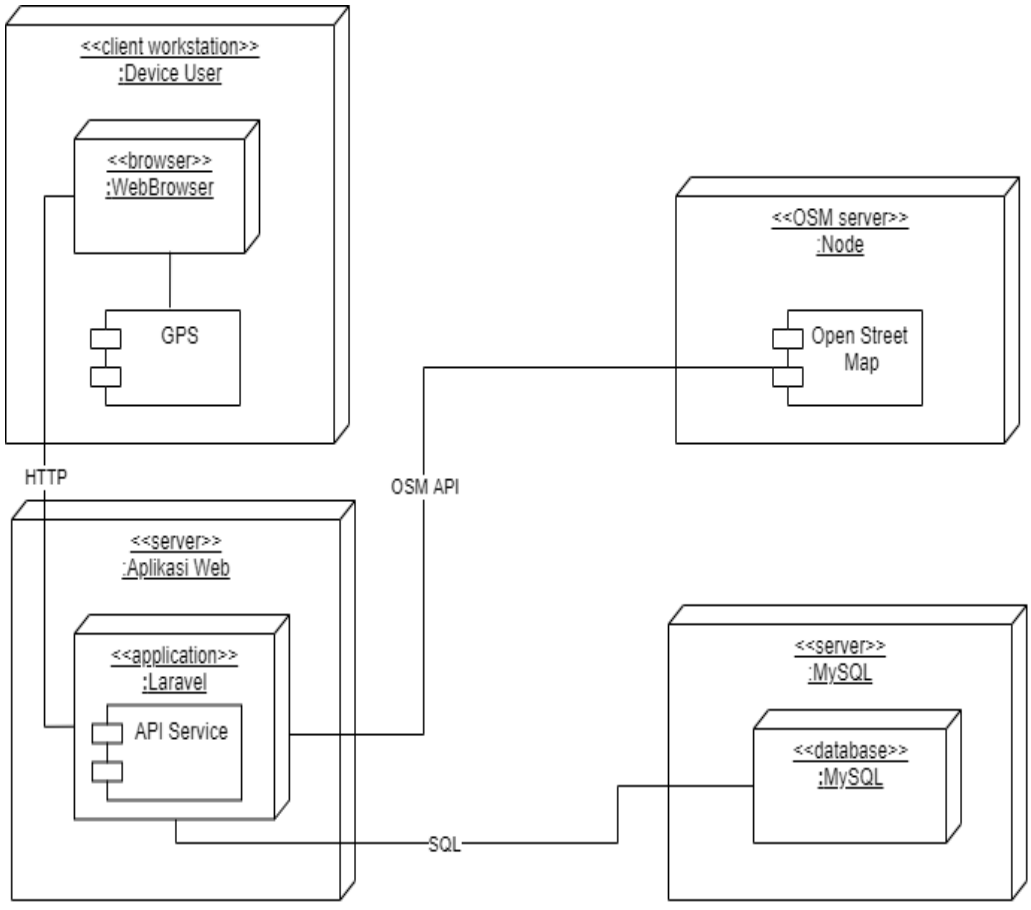
Algoritma

```
Kamus Data:
request : Variabel superglobal berhubungan dengan objek
permintaan HTTP
response : Variabel komposit bertipe JSON
Presensi : Variabel komposit bertipe array

Procedure getAllPresensi()
Begin
      window.location.assign('getAllPresensi')
     request.url ← 'base_url/api/perizinan/showAllPresensi
     request.method ← 'GET'
     response ← send (request) to request.url using
request.method
     write(response.Presensi)
End Procedure

Function showAllPresensi()
Begin
     Presensi ← getAllPresensi()
     return response('Presensi' ← Presensi)
End Function

Function getAllPresensi()
Begin
      Presensi ← Select * from presensi
      return  Presensi
End Function

Procedure searchData(key)
Begin
      For each data Presensi
```

```
            if(Presensi == key)
               then write(Presensi)
            end If
        End For
End Procedure
```

VI. *Deployment View*

Bagian ini berisi penjelasan mengenai arsitektur aplikasi dari sudut pandang *deployment.* Pemodelan UML yang digunakan yaitu *deployment diagram* yang merepresentasikan *deployment view*. *Deployment diagram* digunakan untuk memvisualisasi hubungan antara *software* dan *hardware* yang digunakan untuk menjalankan aplikasi. *Deployment diagram* berguna untuk menggambarkan komunikasi yang terjadi antar *node* secara fisik. (Larman, 2004). Adapun *deployment diagram* digambarkan pada Gambar VI.1.



Gambar VI.1 *Deployment Diagram*

Berdasarkan gambar *deployment diagram* di atas terdapat tiga jenis *node* utama pada aritektur aplikasi yaitu :

1. *Client Workstation*

   Node ini menggambarkan representatif kondisi fisik dari sisi *client* terhadap aplikasi. *Client workstation* memiliki *subnode* yaitu *browser*. *Browser* digunakan *client* untuk mengakses URL aplikasi. *Browser* yang digunakan adalah *web browser*. Selain itu, pada *client workstation* terdapat komponen GPS yang digunakan untuk mendukung aplikasi saat proses pengambilan koordinat pengguna. Komponen mewakili bagian modular dari sistem yang isinya merangkum dan yang manifestasinya dapat diganti dalam lingkungannya. (Larman, 2004). Komunikasi yang terjadi pada *node* ini adalah menggunakan protokol HTTP dengan *node server* aplikasi untuk melakukan *request.*

2. *Server*

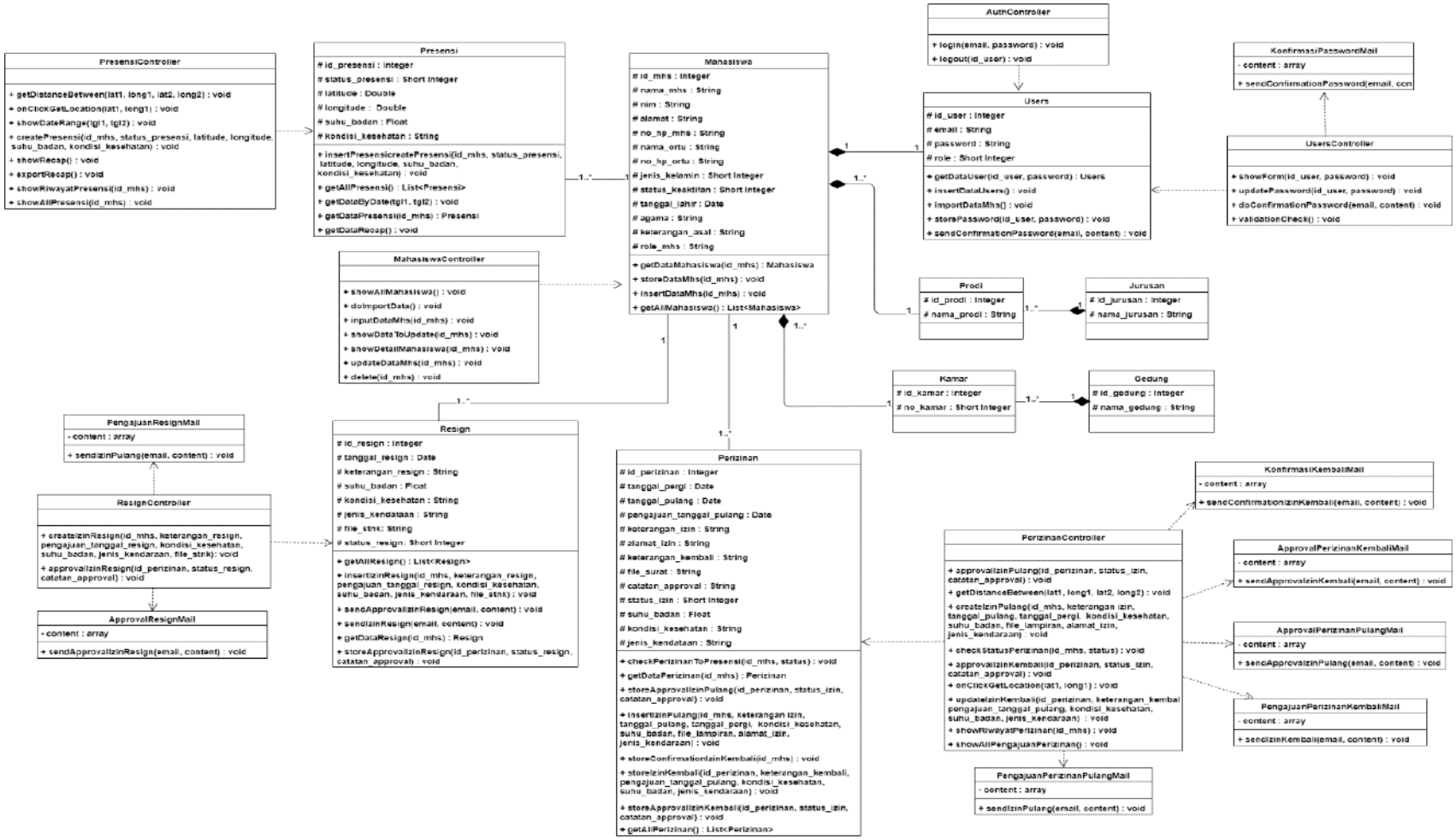
   Node ini menggambarkan representatif kondisi fisik dari sisi *server* aplikasi dan penyimpanan data di *database. Server* aplikasi *web* memiliki *subnode* yaitu *application. Subnode* ini menggambarkan representatif *backend* aplikasi yang digunakan yang di dalamnya terdapat komponen API *service* untuk mendukung komunikasi data. Terdapat komunikasi yang terjalin antara *server web* aplikasi dengan *server database*, dalam hal ini menggunakan MySQL. Komunikasi yang terjadi menggunakan *query* SQL.

3. *OSM Server*

   Node ini menggambarkan representatif kondisi fisik dari sisi *server* OSM yang dimanfaatkan aplikasi. OSM adalah komponen *server* yang digunakan untuk memanfaatkan *maps* yang dimilikinya. OSM bersifat *open source.* Berdasarkan penggambaran *deployment diagram*, OSM *server* berkomunikasi dengan *node server web* aplikasi menggunakan pemanggilan API milik OSM.

VII. *Data View*

Bagian ini menjelaskan arsitektur aplikasi dari sudut pandang kebutuhan data dan skema pengelompokkan data ke dalam kelas yang digambarkan menggunakan *class diagram*. *Class diagram* adalah penggambaran statis yang dikembangkan berdasarkan *domain model*. (Larman, 2004). *Class diagram* terdiri atas pengelompokan kelas, relasi, atribut dan *method* yang digunakan, Pada dokumen SAD ini, penggambarkan *class diagram* merupakan lanjutan dari *subbab architectural significant design package.* Adapun penggambarkannya digambarkan pada Gambar VII.1.



Gambar VII.1 *Class diagram* aplikasi presensi dan perizinan asrama berbasis *web*

Terdapat penjelasan rinci masing-masing *class.* Penjelasan dari masing-masing *class* dijelaskan pada tabel dibawah ini.

Tabel VII.1 Deskripsi *class* Mahasiswa

Nama Class	Mahasiswa		
Deskripsi	*Class* ini berfungsi untuk mengelola data mahasiswa. *Class* ini merupakan *class* model sebagai representasi dari data mahasiswa.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
id_mhs	Protected	Integer	Variabel untuk menampung data id mahasiswa.
nama_mhs	Protected	String	Variabel untuk menampung data nama mahasiswa.
nim	Protected	String	Variabel untuk menampung data nim mahasiswa.
alamat	Protected	String	Variabel untuk menampung data alamat mahasiswa.
no_hp_mhs	Protected	String	Variabel untuk menampung data nomor HP mahasiswa.
nama_ortu	Protected	String	Variabel untuk menampung data nama orang tua dari mahasiswa.
no_hp_ortu	Protected	String	Variabel untuk menampung data nomor HP dari orang tua mahasiswa.
jenis_kelamin	Protected	Short Integer	Variabel ini menampung data jenis kelamin mahasiswa.
status_keaktifan	Protected	Short Integer	Variabel untuk menampung status apakah mahasiswa masih tinggal di asrama atau tidak.
tanggal_lahir	Protected	Date	Variabel untuk menampung data tanggal lahir mahasiswa.
agama	Protected	String	Variabel untuk menampung data agama mahasiswa.

keterangan_asal	Protected	String	Variabel untuk menampung data keterangan asal mahasiswa.
role_mhs	Protected	String	Variabel untuk menampung data role mahasiswa.
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
getAllMahasiswa()	Public	List <Mahasiswa>	Method ini digunakan untuk mendapatkan seluruh data mahasiswa.
getDataMahasiswa(id_mhs)	Public	Mahasiswa	*Method* ini digunakan pengurus untuk mendapatkan data mahasiswa berdasarkan id mahasiswa.
storeDataMhs(id_mhs)	Public	Void	*Method* ini digunakan untuk mengubah data mahasiswa di database.
insertDataMhs(id_mhs)	Public	Void	*Method* ini digunakan untuk menyimpan data mahasiswa ke database.

Tabel VII.2 Deskripsi *class* Users

Nama Class	Users		
Deskripsi	*Class* ini berfungsi untuk mengelola data akun *user*. *Class* ini merupakan *class* model sebagai representasi dari data akun yang digunakan oleh pengguna pada aplikasi.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
id_user	Protected	Integer	Variabel untuk menampung data id *user*.
email	Protected	String	Variabel untuk menampung data *e-mail*.
password	Protected	String	Variabel untuk menampung data *password*.

role	Protected	Short Integer	Variabel untuk menampung data *role* dari *user* tersebut.
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
insertDataUsers()	Public	Void	*Method* ini digunakan untuk mendapatkan seluruh data *user*.
getDataUser(id_user)	Public	Users	*Method* ini digunakan untuk mendapatkan data mahasiswa berdasarkan id *user*.
*import*DataMhs()	Public	Void	*Method* ini digunakan untuk menyimpan data hasil *import.*
storePassword(id_user, password)	Public	Void	*Method* ini digunakan untuk mengupdate data hasil perubahan.
sendConfirmationPassword(email, content)	Public	Void	*Method* ini digunakan untuk mengirim *e-mail* yang ditangani oleh class Mail.

Tabel VII.3 Deskripsi *class* Presensi

Nama Class	Presensi		
Deskripsi	*Class* ini berfungsi untuk mengelola data presensi. *Class* ini merupakan *class* model sebagai representasi dari data presensi yang dilakukan mahasiswa.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
id_presensi	Protected	Integer	Variabel untuk menampung data id presensi.
status_presensi	Protected	Short Integer	Variabel untuk menampung data status presensi berupa hadir, izin, dan alfa.
latitude	Protected	Double	Variabel untuk menampung koordinat latitude mahasiswa ketika melakukan presensi.
longitude	Protected	Double	Variabel untuk menampung koordinat longitude mahasiswa ketika melakukan presensi.

suhu_badan	Protected	Float	Variabel untuk menampung data suhu badan mahasiswa.
kondisi_kesehetan	Protected	String	Variabel untuk menampung data kondisi kesehatan mahasiswa.
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
insertPresensi(id_mhs, status_presensi, latitude, longitude, suhu_badan, kondisi_kesehatan)	Public	Void	Method ini digunakan untuk menyimpan data presensi mahasiswa.
getDataPresensi(id_mhs)	Public	Presensi	Method ini digunakan untuk mendapatkan data presensi berdasarkan id mahasiswa.
getDataRecap()	Public	Void	Method ini digunakan untuk mengambil rekap data presensi mahasiswa.
getDataByDate(tgl1, tgl2)	Public	Void	Method ini digunakan untuk mengambil rekap data presensi mahasiswa berdasarkan rentang tanggal.
getAllPresensi()	Public	List <Presensi>	Method ini digunakan untuk mendapatkan seluruh data presensi mahasiswa.

Tabel VII.4 Deskripsi *class* Perizinan

Nama Class	Perizinan		
Deskripsi	*Class* ini berfungsi untuk mengelola data perizinan. *Class* ini merupakan *class* model sebagai representasi dari data perizinan.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
id_perizinan	Protected	Integer	Variabel untuk menampung data id perizinan.

tanggal_pergi	Protected	Date	Variabel untuk menampung data tanggal awal izin.
tanggal_pulang	Protected	Date	Variabel untuk menampung data tanggal kembali ke asrama.
pengajuan_tanggal_pulang	Protected	Date	Variabel untuk menampung data pengajuan tanggal kembali ke asrama.
keterangan_izin	Protected	String	Variabel untuk menampung data deskripsi atau alasan mahasiswa mengajukan izin.
alamat_izin	Protected	String	Variabel untuk menampung data alamat atau lokasi yang dituju saat izin.
keterangan_kembali	Protected	String	Variabel untuk menampung data alasan saat pengajuan izin kembali ke asrama.
file_surat	Protected	String	Variabel untuk menampung data *path* dan nama file yang diupload.
catatan_approval	Protected	String	Variabel untuk menampung data catatan yang diberikan saat memberikan *approval.*
status_izin	Protected	Short Integer	Variabel untuk menampung status *approval* perizinan.
suhu_badan	Protected	Float	Variabel untuk menampung data suhu badan.

kondisi_kesehatan	Protected	String	Variabel untuk menampung data kondisi kesehatan.
jenis_kendaraan	Protected	String	Variabel untuk menampung data jenis kendaraan yang digunakan untuk izin atau pergi.
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
checkPerizinanToPresensi(id_mhs, status)	Public	Void	*Method* ini digunakan untuk mengambil data perizinan yang dilakukan pengecekkan.
insertIzinPulang(id_mhs, keterangan izin, tanggal_pulang, tanggal_pergi, kondisi_kesehatan, suhu_badan, file_lampiran, alamat_izin, jenis_kendaraan)	Public	Void	*Method* ini digunakan untuk membuat perizinan baru.
getDataPerizinan(id_mhs)	Public	Perizinan	*Method* ini digunakan untuk mengambil data perizinann *by* id
storeApprovalIzinPulang(id_perizinan, status_izin, catatan_approval)	Public	Void	*Method* ini digunakan untuk menyimpan status *approval* izin pulang
getAllPerizinan()	Public	List<Perizinan>	*Method* ini digunakan untuk mendapatkan seluruh data perizinan.
storeIzinKembali(id_perizinan, keterangan_kembali, pengajuan_tanggal_pulang, kondisi_kesehatan, suhu_badan, jenis_kendaraan)	Public	Void	*Method* ini digunakan untuk mendapatkan data perizinan berdasarkan id mahasiswa.
storeApprovalIzinKembali(id_perizinan, status_izin, catatan_approval)	Public	Void	*Method* ini digunakan untuk

			mengambil rekap data perizinan mahasiswa.
storeConfirmationIzinKembali(id_mhs)	Public	Void	*Method* ini digunakan untuk mengambil data perizinan yang sudah di approve.

Tabel VII.5 Deskripsi *class* Resign

Nama Class	Resign		
Deskripsi	*Class* ini berfungsi untuk mengelola data perizinan *resign*. *Class* ini merupakan *class* model sebagai representasi dari data *resign*.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
id_resign	Protected	Integer	Variabel untuk menampung data id *resign*.
tanggal_resign	Protected	Date	Variabel untuk menampung data tanggal *resign*.
keterangan_resign	Protected	String	Variabel untuk menampung alasan mengajukan *resign*.
suhu_badan	Protected	Float	Variabel untuk menampung data suhu badan.
kondisi_kesehatan	Protected	String	Variabel untuk menampung data kondisi kesehatan.
jenis_kendaraan	Protected	String	Variabel untuk menampung data jenis kendaraan yang dimiliki.

file_stnk	Protected	String	Variabel untuk menampung data *path file* foto stnk
status_resign	Protected	Short Integer	Variabel untuk menampung data status *approval resign.*
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
insertIzinResign(id_mhs,keterangan_resign, pengajuan_tanggal_resign, kondisi_kesehatan, suhu_badan, jenis_kendaraan, file_stnk)	Public	Void	*Method* ini digunakan untuk menyimpan data pengajuan perizinan *resign*.
getAllResign()	Public	List<Resign>	*Method* ini digunakan untuk mendapatkan seluruh data perizinan *resign*.
getDataResign(id_mhs)	Public	Resign	*Method* ini digunakan untuk mendapatkan data perizinan resign berdasarkan id mahasiswa.
storeApprovalIzinResign(id_perizinan, status_resign, catatan_approval)	Public	Void	*Method* ini digunakan untuk mengubah data saat *approval resign.*
sendIzinResign(email, content)	Public	Void	*Method* ini digunakan untuk mengirim *e-mail* yang ditangani oleh class Mail.

Tabel VII.6 Deskripsi *class* Prodi

Nama Class	Prodi
Deskripsi	*Class* ini berfungsi untuk mengelola data prodi. *Class* ini merupakan *class* model sebagai representasi dari data prodi.

Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
id_prodi	Protected	Integer	Variabel untuk menampung data id prodi.
nama_prodi	Protected	String	Variabel untuk menampung data nama prodi.
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
-	-	-	-

Tabel VII.7 Deskripsi *class* Jurusan

Nama Class	Jurusan		
Deskripsi	*Class* ini berfungsi untuk mengelola data jurusan. *Class* ini merupakan *class* model sebagai representasi dari data jurusan.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
id_jurusan	Protected	Integer	Variabel untuk menampung data id jurusan.
nama_jurusan	Protected	String	Variabel untuk menampung data nama jurusan.
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
-	-	-	-

Tabel VII.8 Deskripsi *class* kamar

Nama Class	Kamar		
Deskripsi	*Class* ini berfungsi untuk mengelola data kamar asrama. *Class* ini merupakan *class* model sebagai representasi dari data kamar.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
id_kamar	Protected	Integer	Variabel untuk menampung data id kamar.

nomor_kamar	Protected	Short Integer	Variabel untuk menampung data nomor kamar.
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
-	-	-	-

Tabel VII.9 Deskripsi *class* Gedung

Nama Class	Gedung		
Deskripsi	*Class* ini berfungsi untuk mengelola data gedung asrama. *Class* ini merupakan *class* model sebagai representasi dari data gedung.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
id_gedung	Protected	Integer	Variabel untuk menampung data id gedung.
nama_gedung	Protected	String	Variabel untuk menampung data nama gedung.
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
-	-	-	-

Tabel VII.10 Deskripsi *class* MahasiswaController

Nama Class	MahasiswaController		
Deskripsi	*Class* ini berfungsi untuk mengontrol permintaan dari *view* yang berhubungan dengan data mahasiswa yang diteruskan ke model sesuai permintaan.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
-	-	-	-
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**

showAllMahasiswa()	Public	Void	*Method* ini digunakan untuk menampilkan semua data mahasiswa hasil, permintaan dari *view* yang diteruskan ke *model.*
do*Import*Data()	Public	Void	*Method* ini digunakan untuk melakukan *import* data mahasiswa, permintaan dari *view* yang diteruskan ke *model.*
inputDataMhs(id_mhs)	Public	Void	*Method* ini digunakan untuk melakukan *input* data mahasiswa, permintaan dari *view* yang diteruskan ke *model.*
showDataToUpdate(id_mhs)	Public	Void	*Method* ini digunakan untuk menampilkan data yang di*update*, permintaan dari *view* yang diteruskan ke *model.*
showDetailMahasiswa(id_mhs)	Public	Void	*Method* ini digunakan untuk menampilkan *detail* data mahasiswa, permintaan dari *view* yang diteruskan ke *model.*
updateDataMhs(id_mhs)	Public	Void	*Method* ini digunakan untuk melakukan *update* data mahasiswa, permintaan dari *view* yang diteruskan ke *model.*
delete(id_mhs)	Public	Void	*Method* ini digunakan untuk menghapus data mahasiswa dari database.

Tabel VII.11 Deskripsi *class* Users

Nama Class	UsersController		
Deskripsi	Class ini berfungsi untuk mengontrol permintaan dari *view* yang berhubungan dengan data *users* yang diteruskan ke model sesuai permintaan.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
-	-	-	-
Operations			

Nama	Visibility	Return Value	Deskripsi
showForm(id_user, password)	Public	Void	*Method* ini digunakan untuk manampilkan form *reset password*, permintaan dari *view* yang diteruskan ke *model.*
updatePassword(id_user, password)	Public	Void	*Method* ini digunakan untuk mengupdate *password user*, permintaan dari *view* yang diteruskan ke *model.*
doConfirmationPassword(email, content)	Public	Void	*Method* ini digunakan untuk mengirim konfirmasi *password* ke *e-mail*, permintaan dari *view* yang diteruskan ke *model.*
validationCheck()	Public	Void	*Method* ini digunakan untuk melakukan validasi *form*, permintaan dari *view* yang diteruskan ke *model.*

Tabel VII.12 Deskripsi *class* PresensiController

Nama Class	Presensi		
Deskripsi	*Class* ini berfungsi untuk mengontrol permintaan dari view yang berhubungan dengan data presensi yang diteruskan ke model sesuai permintaan.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
-	-	-	-
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
getDistanceBetween(lat1, long1, lat2, long2)	Public	Void	*Method* ini digunakan untuk menghitung jarak dua lokasi, permintaan dari *view* yang diteruskan ke *model.*
onClickGetLocation(lat1, long1)	Public	Void	*Method* ini digunakan untuk mendapatkan data koordinat

			lokasi, permintaan dari *view* yang diteruskan ke *model.*
showDateRange(tgl1, tgl2)	Public	Void	*Method* ini digunakan untuk melihat rekapitulasi berdasarkan tanggal yang diinput permintaan dari *view* yang diteruskan ke *model.*
createPresensi(id_mhs)	Public	Void	*Method* ini digunakan untuk menyimpan data presensi, permintaan dari *view* yang diteruskan ke *model.*
showRecap()	Public	Void	*Method* ini digunakan untuk menampilkan rekapitulasi data, permintaan dari *view* yang diteruskan ke *model.*
exportRecap()	Public	Void	*Method* ini digunakan untuk mengeksport data rekap, permintaan dari *view* yang diteruskan ke *model.*
showRiwayatPresensi(id_mhs)	Public	Void	*Method* ini digunakan untuk menampilkan riwayat presensi, permintaan dari *view* yang diteruskan ke *model.*
showAllPresensi(id_mhs)	Public	Void	*Method* ini digunakan untuk menampilkan seluruh data presensi mahasiswa, permintaan dari *view* yang diteruskan ke *model.*

Tabel VII.13 Deskripsi *class* PerizinanController

Nama Class	PerizinanController		
Deskripsi	*Class* ini berfungsi untuk mengontrol permintaan dari *view* yang berhubungan dengan data perizinan yang diteruskan ke model sesuai permintaan.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
-	-	-	-
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**

approvalIzinPulang(id_perizinan, status_izin, catatan_approval)	Public	Void	*Method* ini digunakan untuk menyimpan data *approval* yang telah dilakukan, permintaan dari *view* yang diteruskan ke *model.*
getDistanceBetween(lat1, long1, lat2, long2)	Public	Void	*Method* ini digunakan untuk menghitung jarak dua lokasi *user*, permintaan dari *view* yang diteruskan ke *model.*
createIzinPulang(id_mhs, keterangan izin, tanggal_pulang, tanggal_pergi, kondisi_kesehatan, suhu_badan, file_lampiran, alamat_izin, jenis_kendaraan)	Public	Perizinan	*Method* ini digunakan untuk menyimpan data izin pulang, permintaan dari *view* yang diteruskan ke *model.*
checkStatusPerizinan(id_mhs, status)	Public	Void	*Method* ini digunakan untuk menyimpan melakukan pengecekkan status perizinan sebelum melakukan presensi, permintaan dari *view* yang diteruskan ke *model.*
approvalIzinKembali(id_perizinan, status_izin, catatan_approval)	Public	List<Perizinan>	*Method* ini digunakan untuk menyimpan data *approval* izin kembali yang telah dilakukan, permintaan dari *view* yang diteruskan ke *model.*
onClickGetLocation(lat1, long1)	Public	Void	*Method* yang digunakan untuk meneruskan koordinat yang telah terambil untuk dilakukan perhitungan, permintaan dari *view* yang diteruskan ke *model.*
updateIzinKembali(id_perizinan, keterangan_kembali, pengajuan_tanggal_pulang, kondisi_kesehatan, suhu_badan, jenis_kendaraan)	Public	Void	*Method* yang digunakan untuk mengupdate status izin kembali, permintaan

			dari *view* yang diteruskan ke *model.*
showRiwayatPerizinan(id_mhs)	Public	Void	*Method* yang digunakan untuk menampilkan riwayat perizinan, permintaan dari *view* yang diteruskan ke *model.*
showAllPengajuanPerizinan()	Public	Void	*Method* yang digunakan untuk menampilkan seluruh data perizinan mahasiswa, permintaan dari *view* yang diteruskan ke *model.*

Tabel VII.14 Deskripsi *class* ResignController

Nama Class	ResignController		
Deskripsi	*Class* ini berfungsi untuk mengontrol permintaan dari *view* yang berhubungan dengan data *resign* yang diteruskan ke model sesuai permintaan.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
-	-	-	-
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
createIzinResign(id_mhs, keterangan_resign, pengajuan_tanggal_resign, kondisi_kesehatan, suhu_badan, jenis_kendaraan, file_stnk)	Public	Void	*Method* yang digunakan untuk menyimpan data izin *resign* yang diajukan, permintaan dari *view* yang diteruskan ke *model.*
approvalIzinResign(id_perizinan, status_resign, catatan_approval)	Public	Void	*Method* yang digunakan untuk menyimpan data hasil *approval* izin *resign*, permintaan dari *view* yang diteruskan ke *model.*

Tabel VII.15 Deskripsi *class* AuthController

Nama Class	AuthController		
Deskripsi	*Class* ini berfungsi untuk mengontrol permintaan dari *view* yang berhubungan dengan *login* dan *logout* akun pengguna dan diteruskan ke model sesuai permintaan.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
-	-	-	-
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
login(email, password)	Public	Void	*Method* yang digunakan untuk melakukan autentikasi akun pengguna, permintaan dari *view* yang diteruskan ke *model.*
logout(id_user)	Public	Void	*Method* yang digunakan untuk menangi *logout* atau akun yang keluar dari aplikasi, permintaan dari *view* yang diteruskan ke *model.*

Tabel VII.16 Deskripsi *class* KonfirmasiPasswordMail

Nama Class	KonfirmasiPasswordMail		
Deskripsi	*Class* ini berfungsi untuk melakukan pengiriman konfirmasi *password* melalui *e-mail* ke *e-mail* pengguna yang dituju terhadap aksi *reset password* yang dilakukan.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
content	Public	Array	Variabel untuk menampung konten *e-mail.*
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**

sendConfirmationPassword(email, content)	Public	Void	*Method* yang digunakan untuk melakukan pengiriman konten *e-mail* konfirmasi *password*.

Tabel VII.17 Deskripsi *class* PengajuanResignMail

Nama Class	PengajuanResignMail		
Deskripsi	*Class* ini berfungsi untuk melakukan pengiriman pengajuan *resign* melalui *e-mail* ke *e-mail* pengguna yang dituju terhadap aksi pengajuan *resign* oleh mahasiswa.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
content	Public	Array	Variabel untuk menampung konten *e-mail.*
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
sendIzinResign(email, content)	Public	Void	*Method* yang digunakan untuk melakukan pengiriman konten *e-mail* izin *resign*.

Tabel VII.18 Deskripsi *class* ApprovalResignMail

Nama Class	ApprovalResignMail		
Deskripsi	*Class* ini berfungsi untuk melakukan pengiriman hasil *approval resign* melalui *e-mail* ke *e-mail* pengguna yang dituju terhadap aksi pemberian *approval* izin *resign.*		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
content	Public	Array	Variabel untuk menampung konten *e-mail.*
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**

sendApprovalResign(email, content)	Public	Void	*Method* yang digunakan untuk melakukan pengiriman konten *e-mail approval resign*.

Tabel VII.19 Deskripsi *class* KonfirmasiKembaliMail

Nama Class	KonfirmasiKembaliMail		
Deskripsi	*Class* ini berfungsi untuk melakukan pengiriman hasil konfirmasi kembali ke asrama melalui *e-mail* ke *e-mail* pengguna yang dituju terhadap aksi mahasiswa melakukan konfirmasi kembali ke asrama.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
content	Public	Array	Variabel untuk menampung konten *e-mail.*
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
sendConfirmationIzinKembali(email, content)	Public	Void	*Method* yang digunakan untuk melakukan pengiriman konten *e-mail* konfirmasi izin kembali ke asrama.

Tabel VII.20 Deskripsi *class* ApprovalPerizinanKembaliMail

Nama Class	ApprovalPerizinanKembaliMail		
Deskripsi	*Class* ini berfungsi untuk melakukan pengiriman hasil *approval* izin kembali ke asrama melalui *e-mail* ke *e-mail* pengguna yang dituju terhadap aksi pemberian *approval* izin kembali ke asrama.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
content	Public	Array	Variabel untuk menampung konten *e-mail.*

Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
sendApprovalIzinKembali(email, content)	Public	Void	*Method* yang digunakan untuk melakukan pengiriman konten *e-mail approval* izin kembali ke asrama.

Tabel VII.21 Deskripsi *class* ApprovalPerizinanPulangMail

Nama Class	ApprovalPerizinanPulangMail		
Deskripsi	*Class* ini berfungsi untuk melakukan pengiriman hasil *approval* izin pulang melalui *e-mail* ke *e-mail* pengguna yang dituju terhadap aksi pemberian *approval* izin pulang.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
content	Public	Array	Variabel untuk menampung konten *e-mail.*
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
sendApprovalIzinPulang(email, content)	Public	Void	*Method* yang digunakan untuk melakukan pengiriman konten *e-mail approval* izin pulang.

Tabel VII.22 Deskripsi *class* PengajuanPerizinanKembaliMail

Nama Class	PengajuanPerizinanKembaliMail		
Deskripsi	*Class* ini berfungsi untuk melakukan pengajuan perizinan kembali ke asrama melalui *e-mail* ke *e-mail* pengguna yang dituju terhadap aksi pengajuan perizinan kembali ke asrama.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**

content	Public	Array	Variabel untuk menampung konten *e-mail.*
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
sendIzinKembali(email, content)	Public	Void	*Method* yang digunakan untuk melakukan pengiriman konten *e-mail* izin kembali ke asrama.

Tabel VII.23 Deskripsi *class* PengajuanPerizinanPulangMail

Nama Class	PengajuanPerizinanPulangMail		
Deskripsi	*Class* ini berfungsi untuk melakukan pengajuan perizinan pulang melalui *e-mail* ke *e-mail* pengguna yang dituju terhadap aksi pengajuan perizinan pulang oleh mahasiswa.		
Attributes			
Nama	**Visibility**	**Data Type**	**Deskripsi**
content	Public	Array	Variabel untuk menampung konten *e-mail.*
Operations			
Nama	**Visibility**	**Return Value**	**Deskripsi**
sendIzinPulang(email, content)	Public	Void	*Method* yang digunakan untuk melakukan pengiriman konten *e-mail* izin pulang.

VIII. *Implementation View*

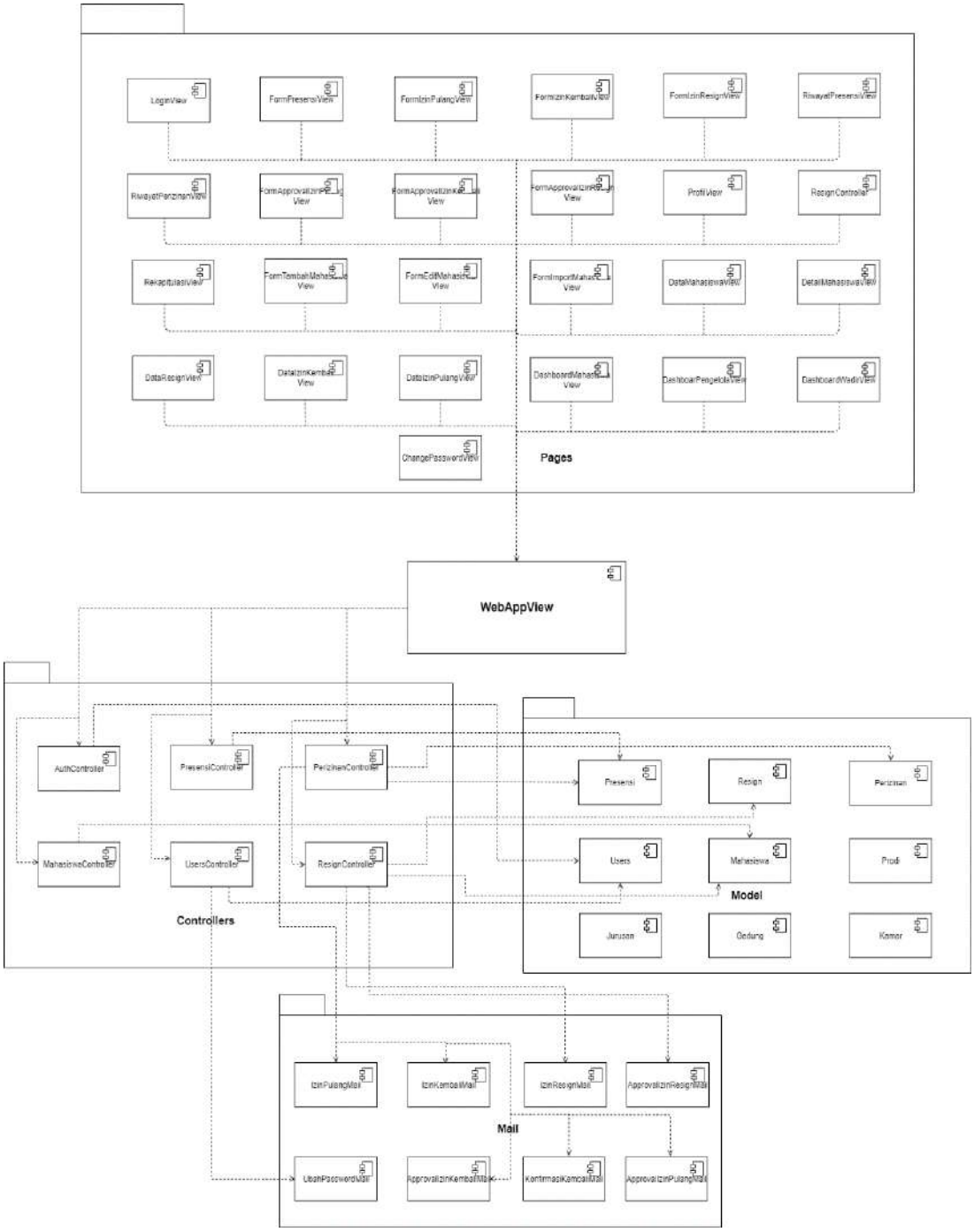
Bagian ini berisi penjelasan mengenai arsitektur aplikasi dari sudut pandang implementasi (*implementation view*). *Implementation view* menjelaskan struktur keseluruhan dari model implementasi, dekomposisi aplikasi yang terdiri atas beberapa lapisan.

VII.1 *Overview*

Aplikasi yang dibangun berbasis aplikasi *web* yang bertujuan agar aplikasi dapat diakses oleh pengguna di *smartphone* atau PC. Aplikasi ini dibangun dengan menggunakan arsitektur MVC (*Model, View, Controller*).

VII.2 *Layers*

Berdasarkan penggambaran *deployment* diagram di atas, aplikasi terdiri atas *presentation layer* dalam konteks ini sebagai *view* atau UI, *application layer* dalam konteks ini sebagai *controllers, business layer* dalam konteks ini sebagai *model* dan *mail.* Setiap *layer* didekomposisi menjadi beberapa komponen di dalamnya. Setiap komponen berinteraksi atau berhubungan dengan komponen lain dalam *layer* yang berbeda sesuai dengan fungsi yang ditangani.



Gambar VIII.1 *Deployment diagram*

Berdasarkan penggambaran *deployment* diagram di atas, aplikasi terdiri atas *presentation layer* dalam konteks ini sebagai *view* atau UI, *application layer* dalam konteks ini sebagai *controllers*, *business layer* dalam konteks ini sebagai *model* dan *mail*. Setiap *layer* didekomposisi menjadi beberapa komponen di dalamnya. Setiap

komponen berinteraksi atau berhubungan dengan komponen lain dalam *layer* yang berbeda sesuai dengan fungsi yang ditangani.

IX. *Size and Performance*

Ukuran aplikasi ini belum dihitung dan belum diperkirakan besarnya, semua fungsionalitas diproses pada RESTful API dan *database* MySQL, serta akan di *hosting* secara *online*.

X. *Quality*

Pada bagian *Architectural Goals* dan *Constraint* sudah mencakup parameter kualitas untuk aplikasi yang akan dibangun.