

**RANCANG BANGUN APLIKASI DAN WEB SERVICE  
PENGKAJIAN LUCA KRONIS  
KHUSUSNYA MODUL PENGOLAHAN CITRA  
BERBASIS ANDROID**



**PROGRAM STUDI ILMU KOMPUTER  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN  
ALAM UNIVERSITAS NEGERI JAKARTA  
2023**

# LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI

## RANCANG BANGUN APLIKASI DAN WEB SERVICE

### PENGKAJIAN LUKA KRONIS

#### KHUSUSNYA MODUL PENGOLAHAN CITRA

#### BERBASIS ANDROID

Nama : Salsa Rahmadati

No. Registrasi : 1313617010

	Nama	Tanda Tangan	Tanggal
Penanggung Jawab			
Dekan	: Prof. Dr. Muktiningsih N, M.Si..		28/2/2023
	NIP. 196405111989032001		
Wakil Penanggung Jawab			
Wakil Dekan I	: Dr. Esmar Budi, S.Si., MT.		1/3/2023
	NIP. 197207281999031002		
Ketua	: Drs. Mulyono, M.Kom.		17/2/2023
	NIP. 196605171994031003		
Sekretaris	: Ari Hendarno S.Pd., M.Kom.		16/02/2023
	NIP. 198811022022031002		
Pengaji	: Med Irzal, M.Kom.		15/02/2023
	NIP. 197706152003121001		
Pembimbing I	: Muhammad Eka Suryana, M.Kom.		20/02/2023
	NIP. 19851223 2012121002		
Pembimbing II	: Dr. Ria Arafiyah, M.Si.		20/02/2023
	NIP. 197511212005012004		

Dinyatakan lulus ujian skripsi tanggal: 07 Februari 2023

## LEMBAR PERNYATAAN

Saya menyatakan dengan sungguh-sungguh bahwa skripsi dengan judul **“Rancang Bangun Aplikasi dan Web Service Pengkajian Luka Kronis Khususnya Modul Pengolahan Citra Berbasis Android”** yang telah saya susun sebagai syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Ilmu Komputer Universitas Negeri Jakarta adalah karya ilmiah saya dengan arahan dari dosen pembimbing.

Seluruh bahan dan data yang didapatkan dari penulis terdahulu yang sudah terpublikasikan yang tercantum dalam teks skripsi ini, telah tercantum di dalam Daftar Pustaka sesuai dengan etika, norma, dan kaidah penulisan ilmiah.

Apabila dikemudian hari diketemukan sebagian isi skripsi ini bukan hasil karya saya sendiri dalam beberapa bagian tertentu, saya bersedia mendapatkan sanksi pencabutan gelar akademik yang saya sanding dan sanksi-sanksi lainnya sesuai dengan peraturan perundang-undangan yang diberlakukan.

Jakarta, 30 Januari 2023



Salsa Rahmadati



KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN  
UNIVERSITAS NEGERI JAKARTA  
UPT PERPUSTAKAAN

Jalan Rawamangun Muka Jakarta 13220  
Telepon/Faksimili: 021-4894221  
Laman: [lib.unj.ac.id](http://lib.unj.ac.id)

**LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI  
KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademika Universitas Negeri Jakarta, yang bertanda tangan di bawah ini, saya:

Nama : Salsa Rahmadati  
NIM : 1313617010  
Fakultas/Prodi : Matematika dan IPA/Ilu Komputer  
Alamat email : [salsarahmadati@gmail.com](mailto:salsarahmadati@gmail.com)

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada UPT Perpustakaan Universitas Negeri Jakarta, Hak Bebas Royalti Non-Ekslusif atas karya ilmiah:

Skripsi       Tesis       Disertasi       Lain-lain (.....)

yang berjudul :

Rancang Bangun Aplikasi Pengkajian Luka Kronis Khususnya Modul Pengolahan Citra

Berbasis Android

Dengan Hak Bebas Royalti Non-Ekslusif ini UPT Perpustakaan Universitas Negeri Jakarta berhak menyimpan, mengalihmediakan, mengelolanya dalam bentuk pangkalan data (*database*), mendistribusikannya, dan menampilkan/mempublikasikannya di internet atau media lain secara *fulltext* untuk kepentingan akademis tanpa perlu meminta ijin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan atau penerbit yang bersangkutan.

Saya bersedia untuk menanggung secara pribadi, tanpa melibatkan pihak Perpustakaan Universitas Negeri Jakarta, segala bentuk tuntutan hukum yang timbul atas pelanggaran Hak Cipta dalam karya ilmiah saya ini.

Demikian pernyataan ini saya buat dengan sebenarnya.

Jakarta, 1 Maret 2023

Penulis

  
( Salsa Rahmadati )  
*nama dan tanda tangan*

## LEMBAR PERSEMPAHAN



## ABSTRAK

**SALSA RAHMADATI.** Rancang Bangun Aplikasi dan Web Service Pengkajian Luka Kronis Khususnya Modul Pengolahan Citra Berbasis Android. Skripsi, Program Studi Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta, Januari 2023.

Pada umumnya, pengkajian luka masih dilakukan secara konvensional dengan cara pengarsipan atau catatan kertas. Selain itu, penelitian sebelumnya memiliki kekurangan *data ground truth* anotasi luka kronis untuk diuji. Sehingga, diperlukannya aplikasi yang dapat mengarsipkan data pengkajian luka secara digital serta dapat menganotasi luka kronis yang akan digunakan sebagai *ground truth*. Adapun Penelitian ini bertujuan untuk merancang aplikasi pengkajian luka kronis dengan modul pengolahan citra berbasis *Android*. Jenis Penelitian ini adalah Pengembangan/*Research and Development*. Data diambil dari hasil wawancara dengan dosen Politeknik Kesehatan Jakarta dan studi literatur dengan membaca jurnal-jurnal yang terkait dengan topik penelitian. Data kajian yang digunakan dalam penelitian ini terfokus pada kategori kajian luka yang memiliki data gambar seperti ukuran luka, tepi luka, dan epitalisasi luka. Hasil Penelitian menunjukkan bahwa pada *User Acceptance Test* dengan metode *Black Box* yang dilakukan terhadap *internal developer* dan perawat, didapatkan hasil penelitian: (1) Terciptanya prototipe aplikasi pengkajian luka kronis versi pertama berbasis *Android* yang sudah mengintegrasikan fitur-fitur pada Product Backlog. Adapun perancangannya dilakukan dengan metode Scrum dengan tahapan penyusunan Product Backlog, Sprint Backlog, dan dikerjakan dalam sembilan Sprint; (2) Terimplementasikannya Web Service yang berfungsi sebagai Back-End aplikasi pengkajian luka kronis berbasis *Android*; (3) Berdasarkan hasil pengujian yang dilakukan terhadap satu anggota tim penelitian, didapatkan bahwa fitur-fitur yang terdapat pada aplikasi berjalan dengan baik dan lulus uji fungsional. Sedangkan berdasarkan hasil *User Acceptance Test* terhadap satu perawat, didapatkan bahwa masih adanya fitur yang belum sesuai dengan kebutuhan perawat dan belum siap untuk tahap pengujian lebih lanjut terhadap lebih dari satu perawat.

**Kata Kunci:** *Aplikasi, Web-service, Luka Kronis, Pengolahan Citra*

## ABSTRACT

**SALSA RAHMADATI.** Design of Chronic Wound Assessment Applications and Web Services, Especially Image Processing Modules Based on Android. Mini Thesis, Computer Science Department, Faculty of Mathematics and Natural Sciences, Universitas begeri Jakarta, January 2023.

*In most cases, wound assessment is still conducted using traditional filing or paper records. In addition, research that has been done previously has a lack of ground truth data on chronic wound annotations to be tested. Consequently, an application that can digitally archive wound assessment data and can annotate chronic wounds that will be used as ground truth is needed. This research aims to develop an Android for diagnosing chronic wounds using image processing components. This research and development (R&D), data were taken from interviews with lecturers at the Jakarta Healt Polytechnic and journal articles pertinent to the research issue. The assessment data used in this study focused on the category of wound studies that had image data such as wound size, wound edges, and wound epithelium. The result of the research indicated that User Acceptance Test, conducted on internal developers and nurse using the Black Box method, yielded the following result: (1) Creation of the first version of the Android-based chronic wound assessment application prototype that has integrated features in the Product Backlog. The design is carried out using the Scrum method with the stages of compiling a Product Backlog, Sprint Backlog, and is done in nine Sprints; (2) Implementation of a Web Service that functions as the Back-End of an Android-based chronic wound assessment application; (3) Based on the results of tests conducted on one member of the research team, it was found that the features contained in the application ran well and passed the functional test. Meanwhile, based on the results of the User Acceptance Test on one nurse, it was found that there were still features that were not in accordance with the needs of nurses and were not ready for further testing on more than one nurse.*

**Keywords:** Application, Web-services, Chronic Wounds, Image Processing

## KATA PENGANTAR

Penulis memanjatkan puji dan syukur kepada Allah SWT atas seluruh rahmat dan hidayah-Nya sehingga penulis mampu menyelesaikan proposal yang berjudul. **“Rancang Bangun Aplikasi dan Web Service Pengkajian Luka Kronis Khususnya Modul Pengol Pengolahan Citra Berbasis Android”**. Keberhasilan penyusunan proposal ini tidak lepas dari bantuan berbagai pihak yang dengan ikhlas dan tulus memberikan saran dan masukan yang bermanfaat dalam proses penyusunan proposal ini. Oleh karena itu, pada kesempatan ini dengan kerendahan hati, penulis mengucapkan terima kasih kepada:

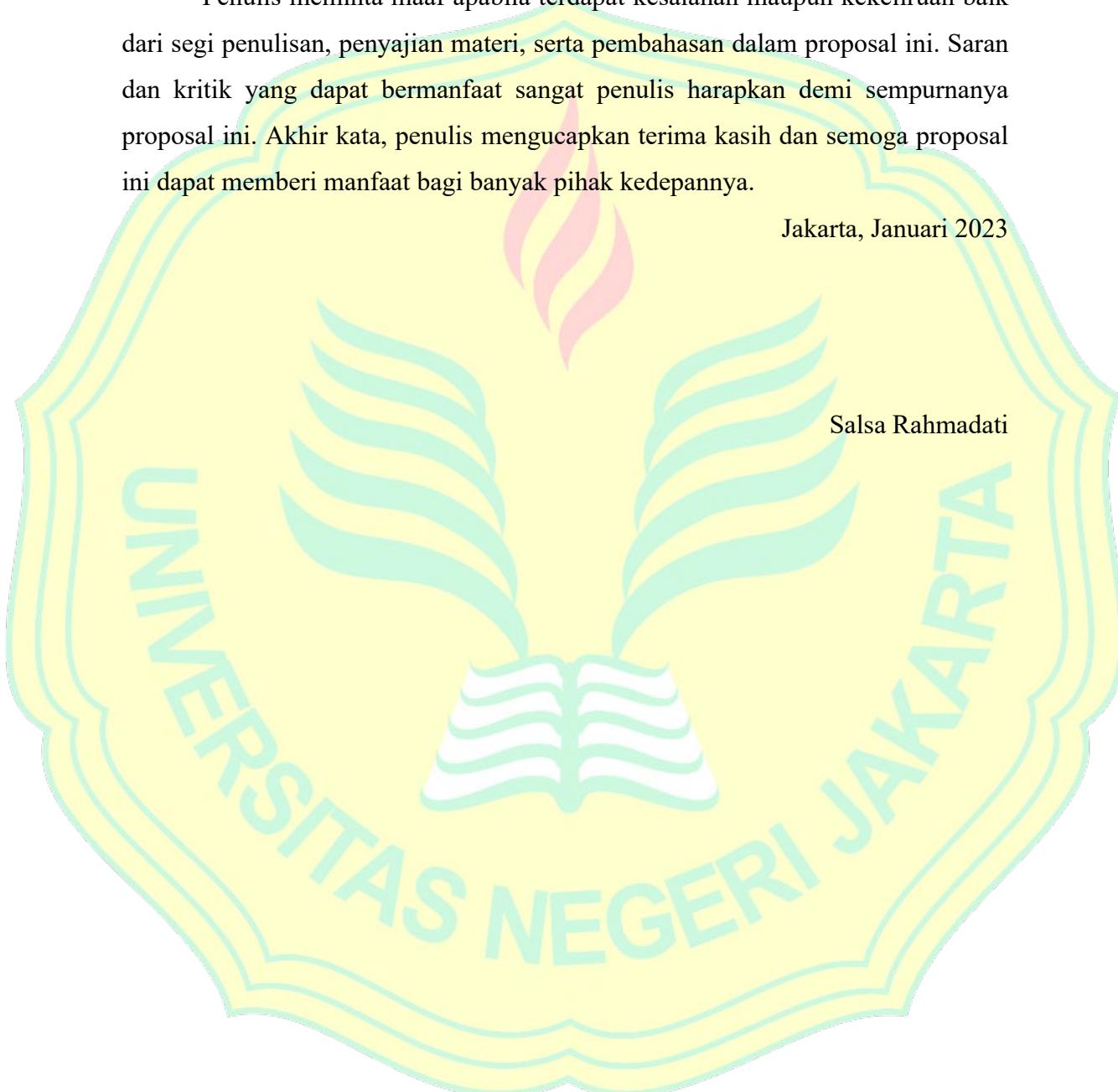
1. Ibu Ir. Fariani Hermin Indiyah, M.T., selaku Koordinator Program Studi Ilmu Komputer FMIPA Universitas Negeri Jakarta,
2. Bapak Muhammad Eka Suryana, M. Kom., selaku dosen pembimbing I yang telah memberikan bimbingan, masukan, dan saran baik secara konten maupun penulisan,
3. Ibu Ria Arafiyah, M.Si., selaku dosen pembimbing II yang telah memberikan bimbingan, masukan, dan saran baik secara konten maupun penulisan,
4. Kedua orang tua penulis dan keluarga yang selama ini telah senantiasa sabar mendorong, memberikan semangat, dan mendoakan penulis,
5. Teman-teman Ilmu Komputer angkatan 2017 sebagai keluarga kedua di kampus yang senantiasa menemani dan memberikan semangat semenjak awal dunia perkuliahan.
6. Saya sendiri yang telah berjuang keras, melalui banyak suka duka, tegar dalam menghadapi rintangan yang ada serta menyelesaikan penyusunan skripsi ini.
7. Teman-teman dalam grup mabar AOV yang senantiasa hadir untuk membantu penulis dalam proses penggeraan skripsi ini.
8. Stella Aurelia, Thalia Shifa Susanto, Daniel, Indri Febriani Hartono serta Ayu Wiwaha yang menjadi tempat cerita, berbagi tawa dan kasih, dan memberikan kebahagiaan bagi penulis dalam penyusunan skripsi ini.

9. Victoria Vania Blanca Widjaya, Deslita Savitri Ramadhiani, serta Salsabila Nadhifa Hartanto selaku teman-teman alumni SMP 252 Jakarta yang senantiasa menjadi penyemangat, teman untuk berbagi tawa, dan memotivasi saya untuk menyelesaikan penyusunan skripsi ini.

Penulis meminta maaf apabila terdapat kesalahan maupun kekeliruan baik dari segi penulisan, penyajian materi, serta pembahasan dalam proposal ini. Saran dan kritik yang dapat bermanfaat sangat penulis harapkan demi sempurnanya proposal ini. Akhir kata, penulis mengucapkan terima kasih dan semoga proposal ini dapat memberi manfaat bagi banyak pihak kedepannya.

Jakarta, Januari 2023

Salsa Rahmadati



## DAFTAR ISI

Halaman

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI .....	i
LEMBAR PERNYATAAN .....	ii
LEMBAR PERSEMBERAHAN .....	iii
ABSTRAK .....	iv
ABSTRACT .....	v
KATA PENGANTAR.....	vi
DAFTAR ISI.....	viii
DAFTAR TABEL.....	x
DAFTAR GAMBAR .....	xiii
DAFTAR LAMPIRAN .....	xv
BAB I .....	1
PENDAHULUAN.....	1
A. Latar Belakang .....	1
B. Identifikasi Masalah .....	3
C. Perumusan Masalah .....	3
D. Pembatasan Masalah .....	3
E. Tujuan Penelitian .....	3
F. Manfaat Penelitian .....	4
BAB II .....	5
KAJIAN TEORI.....	5
A. Pengkajian Luka.....	5
B. Pengantar Android .....	13
C. Pengantar Flask .....	21
D. MongoDB.....	23
E. Scrum .....	28
F. Unified Modelling Language ( <i>UML</i> ) .....	31
G. <i>Unit Testing</i> .....	34
H. <i>User Acceptance Test (UAT)</i> .....	35
BAB III .....	36
METODOLOGI PENELITIAN .....	36

A.	Pengumpulan Data .....	36
B.	Analisa Kebutuhan .....	36
C.	Perancangan Sistem Menggunakan <i>Scrum</i> .....	36
D.	Pengujian.....	39
<b>BAB IV</b>	.....	<b>45</b>
<b>HASIL DAN PEMBAHASAN</b>	.....	<b>45</b>
A.	Pembahasan.....	45
B.	Uji Aplikasi .....	184
<b>BAB V</b>	.....	<b>195</b>
<b>KESIMPULAN DAN SARAN</b>	.....	<b>195</b>
A.	Kesimpulan .....	195
B.	Saran.....	195
<b>DAFTAR PUSTAKA</b>	.....	<b>197</b>
<b>LAMPIRAN</b>	.....	<b>200</b>
<b>DAFTAR RIWAYAT HIDUP</b>	.....	<b>215</b>

## DAFTAR TABEL

Tabel 2.1 Metode Class Canvas dan Paint.....	15
Tabel 2.2 Fungsi-fungsi Metode HTTP .....	21
Tabel 2.3 Notasi Use Case Diagram .....	32
Tabel 2.4 Notasi Activity Diagram.....	33
Tabel 2.5 Notasi Entity Relationship Diagram.....	34
Tabel 3.1 Product Backlog.....	37
Tabel 3.2 Skenario Unit Testing .....	39
Tabel 3.3 Skenario Pengujian User Acceptance Test .....	44
Tabel 4.1 Sprint-1 Backlog.....	45
Tabel 4.2 Sprint-2 Backlog .....	75
Tabel 4.3 Routing Table Pasien .....	79
Tabel 4.4 Sprint-3 Backlog .....	86
Tabel 4.5 Routing Table Image .....	92
Tabel 4.6 Sprint-4 Backlog .....	97
Tabel 4.7 <i>Routing Table Data Kajian</i> .....	105
Tabel 4.8 Sprint-5 Backlog .....	109
Tabel 4.9 Sprint-6 Backlog .....	116
Tabel 4.10 Sprint-7 Backlog .....	118
Tabel 4.11 Sprint-9 Backlog .....	124
Tabel 4.12 Routing Table Log Activity .....	151
Tabel 4.13 Product Backlog Tested .....	154
Tabel 4.14 Deskripsi Field GET All Perawat .....	158
Tabel 4.15 Deskripsi Parameter URL Create Perawat.....	159
Tabel 4.16 Deskripsi Field Create Perawat.....	159
Tabel 4.17 Deskripsi Parameter URL Validasi Login .....	160
Tabel 4.18 Deskripsi Field Validasi Login .....	160
Tabel 4.19 Deskripsi Request Body Update Perawat .....	161
Tabel 4.20 Deskripsi Field Update Perawat.....	161
Tabel 4.21 Deskripsi Parameter URL Delete Perawat.....	161
Tabel 4.22 Deskripsi Field Hapus Perawat.....	162

Tabel 4.23 Deskripsi Field Pasien.....	163
Tabel 4.24 Deskripsi Parameter URL Get 1 Pasien.....	163
Tabel 4.25 Deskripsi Field Pasien.....	164
Tabel 4.26 Request Body Create Pasien .....	165
Tabel 4.27 Deskripsi Field Create Perawat.....	165
Tabel 4.28 Deskripsi Parameter URL Delete Perawat.....	166
Tabel 4.29 Deskripsi Field Get All Image .....	167
Tabel 4.30 Deskripsi Parameter URL Get 1 Image .....	168
Tabel 4.31 Deskripsi Field Get 1 Image .....	168
Tabel 4.32 Request Body Upload Image .....	169
Tabel 4.33 Deskripsi Field Upload Image .....	169
Tabel 4.34 Deskripsi Parameter URL Get 1 Image .....	170
Tabel 4.35 Deskripsi Field Get 1 Image .....	170
Tabel 4.36 Deskripsi Parameter URL Delete Image.....	171
Tabel 4.37 Deskripsi Field Get All Image .....	172
Tabel 4.38 Deskripsi Parameter URL Delete Image.....	173
Tabel 4.39 Deskripsi Field Get All Image .....	174
Tabel 4.40 Request Body Membuat Kajian Baru .....	175
Tabel 4.41 Deskripsi Field Membuat Kajian Baru .....	176
Tabel 4.42 Deskripsi Parameter URL Hapus Data Kajian.....	177
Tabel 4.43 Deskripsi Field Membuat Kajian Baru .....	177
Tabel 4.44 Deskripsi Parameter URL Hapus Data Kajian.....	178
Tabel 4.45 Deskripsi Field Hapus Data Kajian.....	178
Tabel 4.46 Deskripsi Field Seluruh Log Activity .....	179
Tabel 4.47 Deskripsi Parameter URL Log Activity Perawat.....	180
Tabel 4.48 Deskripsi Field Log Activity Perawat.....	180
Tabel 4.49 Request Body Membuat Log Activity Baru .....	181
Tabel 4.50 Deskripsi Field Membuat Log Activity Baru .....	181
Tabel 4.51 Deskripsi Parameter URL Hapus Log Activity .....	182
Tabel 4.52 Deskripsi Field Hapus Log Activity .....	182
Tabel 4.53 Pengujian Halaman Awal .....	184
Tabel 4.54 Pengujian Registrasi User/Perawat.....	184

Tabel 4.55 Pengujian Login User .....	185
Tabel 4.56 Pengujian Halaman Beranda.....	185
Tabel 4.57 Pengujian Halaman Daftar Pasien .....	185
Tabel 4.58 Pengujian Tambah Pasien Baru .....	186
Tabel 4.59 Pengujian Halaman Detail Pasien.....	186
Tabel 4.60 Pengujian Halaman Edit Detail Pasien .....	187
Tabel 4.61 Pengujian Tambah Hasil Kajian .....	188
Tabel 4.62 Pengujian Halaman Detail Kajian.....	189
Tabel 4.63 Pengujian Edit Skoring Kajian.....	189
Tabel 4.64 Pengujian Halaman Detail Foto .....	190
Tabel 4.65 Pengujian Arsir Warna Luka .....	190
Tabel 4.66 Pengujian Halaman Detail Perawat .....	191
Tabel 4.67 Pengujian Fitur Edit Perawat .....	191
Tabel 4.68 Pengujian Fitur Log Activity User.....	192
Tabel 4.69 Pengujian Fitur Unduh Dataset Luka.....	192
Tabel 4.70 Pengujian Fitur Aplikasi pada Perawat.....	193

## DAFTAR GAMBAR

Gambar 2.1 Contoh pengukuran ukuran luka .....	6
Gambar 2.2 Contoh pengukuran goa luka dengan pendekatan tiga dimensi .....	8
Gambar 2.3 Data statistik pengguna Android.....	13
Gambar 2.4 Arsitektur Android .....	14
Gambar 2.5 Contoh registrasi Blueprint pada dokumen “auth.py” .....	22
Gambar 2.6 Dokumen init .py.....	23
Gambar 2.7 Contoh pembuatan routing dengan Blueprint .....	23
Gambar 2.8 Pemodelan RDBMS .....	24
Gambar 2.9 Pemodelan NoSQL .....	24
Gambar 2.10 Menambahkan Ekstensi PyMongo pada Flask .....	25
Gambar 2.11 Kerangka kerja metode pengembangan Scrum.....	28
Gambar 3.1 Tahapan penelitian .....	36
Gambar 3.2 Tahapan metode pengembangan menggunakan Scrum .....	37
Gambar 4.1 Pemanggilan REST API buat akun .....	46
Gambar 4.2 Tabel User .....	47
Gambar 4.3 Tampilan Awal.....	49
Gambar 4.4 Tampilan Buat Akun .....	50
<i>Gambar 4.5 Pemanggilan REST API validasi akun.....</i>	66
Gambar 4.6 Tampilan laman Sign In.....	67
Gambar 4.7 Tampilan Laman Beranda.....	76
Gambar 4.8 Mockup Tampilan Penambahan Pasien .....	77
Gambar 4.9 Mockup Tampilan Daftar Pasien .....	77
Gambar 4.10 Mockup Tampilan Detail Pasien Bagian Informasi Umum .....	78
Gambar 4.11 Desain Database Pasien .....	79
Gambar 4.12 REST API Semua Pasien .....	80
Gambar 4.13 REST API Pasien Berdasarkan NRM .....	80
Gambar 4.14 REST API Menambahkan Pasien Baru .....	80
Gambar 4.15 REST API Menghapus Data Pasien.....	81
Gambar 4.16 Implementasi Tampilan Beranda pada XML .....	81

Gambar 4.17 Implementasi Tampilan Tambah Pasien pada XML.....	82
Gambar 4.18 Implementasi Tampilan Detail Pasien pada XML .....	82
Gambar 4.19 Tampilan Tombol Kamera .....	88
Gambar 4.20 Desain Database Image.....	91
Gambar 4.21 REST API Unggah Image.....	93
Gambar 4.22 Mockup Tampilan Detail Pasien Bagian Galeri Luka .....	95
Gambar 4.23 Implementasi Tampilan Galeri Luka Pasien .....	98
Gambar 4.24 Implementasi Tampilan Tambah Kajian.....	103
Gambar 4.25 Desain Database Data Kajian Luka.....	103
Gambar 4.26 Mockup Anotasi Diameter Luka.....	116
Gambar 4.27 Implementasi Layout Anotasi Tepi Luka.....	117
Gambar 4.28 Implementasi Layout Anotasi Tepi Luka.....	118
Gambar 4.29 Flowchart Anotasi Tepi Luka & Diameter Luka .....	119
Gambar 4.30 Implementasi Mockup Arsir Warna Luka.....	130
Gambar 4.31 Layout Galeri Luka Pada XML.....	135
Gambar 4.32 REST API Get All Image.....	135
Gambar 4.33 Mockup Tampilan Detail Pasien Bagian Histori Kajian.....	140
Gambar 4.34 Tampilan Detail Kajian.....	140
Gambar 4.35 Tampilan Histori Kajian pada XML .....	141
Gambar 4.36 Tampilan Detail Kajian pada XML.....	141
Gambar 4.37 Tampilan Profil Perawat pada XML .....	147
Gambar 4.38 Tampilan Profil Perawat pada XML .....	148
Gambar 4.39 REST API Log Activity .....	152
Gambar 4.40 REST API Log Activity Berdasarkan ID User .....	153
Gambar 4.41 Menghapus Log Activity Berdasarkan ID Log .....	153
Gambar 4.42 Desain Database Sistem .....	156
Gambar 4.43 Use Case Diagram.....	157
Gambar 4.44 Flowchart Aplikasi.....	183

## **DAFTAR LAMPIRAN**

Lampiran 1 Transkrip Wawancara Pertama.....	200
Lampiran 2 Transkrip Wawancara Kedua .....	202
Lampiran 3 User Acceptance Test Internal Developer.....	205
Lampiran 4 User Acceptance Test Perawat.....	213



## **BAB I**

### **PENDAHULUAN**

#### **A. Latar Belakang**

Luka kronis ialah klasifikasi luka berdasarkan lama penyembuhan, yaitu jenis luka yang belum memiliki tanda-tanda akan sembuh dengan rentang waktu lebih dari 4-6 minggu (Kartika *et al.*, 2015). Jika perawatan yang digunakan tidak efektif, luka kronis bisa saja memerlukan waktu yang lebih lama untuk sembuh, oleh sebab itu pengkajian luka yang komprehensif harus dilakukan. Pengkajian luka memiliki beberapa manfaat yaitu untuk mengawasi perbaikan pada luka, menetapkan *goal setting* dan menjadi dasar pemberian balutan luka yang tepat (Greatrex-White & Moxey, 2015).

Berdasarkan hasil wawancara dengan Ns. Ratna Aryani, M.Kep., Dosen Politeknik Negeri Jakarta I, pengkajian luka dilakukan pada saat awal pengecekan kondisi luka dan saat penggantian balutan luka. Adapun tahapan pengkajian luka dimulai dari membuka balutan luka, pencucian luka, dan dilanjutkan dengan proses pengkajian. Salah satu instrumen yang digunakan untuk melakukan pengkajian luka adalah Bates-Jensen Wound Assesment Tools (BWAT). Pada instrumen pengkajian luka, BWAT terdapat tiga belas kategori penilaian beberapa diantaranya adalah ukuran luka, tepi luka, jumlah eksudat dan epitalisasi. Saat ini pencatatan data pengkajian luka masih dilakukan dan disimpan secara konvensional pada arsip atau catatan kertas. Oleh karena itu, peneliti menyarankan untuk mengembangkan digitalisasi dalam pencatatan data pengkajian luka.

Menurut penelitian yang dilakukan oleh Dienillah & Dewi (2018) tentang upaya penyelamatan informasi melalui proses digitalisasi arsip akta kelahiran di dinas kependudukan dan pencatatan sipil kota, menjelaskan bahwa digitalisasi arsip atau catatan kertas dilakukan untuk mengamankan, melestarikan, dan mencegah terjadinya kerusakan data, sehingga dapat digunakan di masa mendatang. Selain itu, digitalisasi arsip juga memberikan kemudahan akses, data, kontrol dokumen, dan penyimpanan yang terorganisir (Siregar, 2019).

Penelitian serupa telah dilakukan oleh Wang, S. et al. (2018), dimana dalam penelitiannya menghasilkan aplikasi pengkajian luka kronis dengan fitur pengukuran manual dan otomatis. Kategori pengkajian luka kronis yang diukur secara manual, diantaranya yaitu: ukuran luka, komposisi jaringan, tipe eksudat, bau, derajat kesakitan dan status kulit sekitar luka. Sedangkan kategori pengkajian luka yang diukur secara otomatis, yaitu: lebar luka, panjang luka, luas luka dan komposisi jaringan. Terdapat perbaikan teknis yang dapat dilakukan dengan cara peningkatan akurasi dari deteksi luka secara otomatis, dimana dalam beberapa kasus tepi luka cenderung tidak dapat dilihat dengan jelas dan mirip dengan warna kulit keliling luka, sehingga mengakibatkan deteksi luka yang tidak tepat. Hal tersebut dapat dicapai jika terdapat *big data* dari gambar luka kronis serta gambar luka yang teranotasi.

Pada payung penelitian yang sama, *medical imaging*, telah dilakukan penelitian tentang Pengaruh Penggunaan Color Model LAB dalam Kalibrasi Warna Luka Menggunakan Metode Segmentasi K- Means dan Mean Shift, oleh Khairunnisa, Aprilia (2021) dan penelitian tentang Deteksi Keliling Luka Menggunakan Active Contour yang dilakukan oleh Muhammad Rizki. Kedua penelitian tersebut merupakan penelitian berdasarkan dua kategori pengkajian luka yaitu warna luka dan tepi luka, algoritma yang dikembangkan pada penelitian tersebut direncanakan akan terintegrasi dalam satu aplikasi. Adapun hambatan yang terdapat pada penelitian tersebut antara lain: (1) tidak terdapat *big data ground truth* warna luka dan tepi luka oleh perawat, dimana jika terdapat *ground truth* maka tingkat akurasi dan ketahanan algoritma dapat diuji (Huang & Dom, 1995); (2) Berdasarkan Lampiran 2., perawat tidak mungkin untuk memberikan anotasi warna luka sehingga data *ground truth* warna tidak bisa didapatkan, namun anotasi tepi luka memungkinkan untuk dilakukan.

Berdasarkan uraian di atas, dapat dianalisis bahwa dibutuhkannya aplikasi untuk mengarsipkan data pengkajian luka secara digital, serta dibutuhkannya data *ground truth* untuk meningkatkan ketepatan deteksi luka, warna luka ataupun kategori deteksi citra lainnya. Oleh karena itu, penelitian yang dilakukan bertujuan untuk membuat aplikasi pengkajian luka kronis berbasis *Android* yang akan terfokus pada modul pengolahan citra. Aplikasi tersebut diharapkan dapat

membantu pengumpulan data *ground truth*, dan menjaga data pengkajian luka kronis agar tidak mudah rusak serta dapat dimanfaatkan sebagai bahan penelitian selanjutnya.

## B. Identifikasi Masalah

Masalah yang diidentifikasi pada fokus penelitian ini adalah sebagai berikut:

1. Pengkajian luka umumnya masih dilakukan dan disimpan secara konvensional menggunakan arsip atau catatan kertas.
2. Dibutuhkan data *ground truth* untuk meningkatkan ketepatan deteksi luka, warna luka ataupun kategori deteksi citra lainnya.

## C. Perumusan Masalah

Berdasarkan uraian pada latar belakang yang telah dipaparkan, maka perumusan masalah dalam penelitian yang akan dilakukan adalah bagaimana tahapan perancangan aplikasi untuk pengkajian luka kronis berbasis *Android* modul pengolahan citra?

## D. Pembatasan Masalah

Adapun batasan-batasan masalah dalam penelitian yang akan dilakukan, diantaranya sebagai berikut:

1. Aplikasi pengkajian luka dibuat dengan dasar instrumen pengkajian *Bates-Jensen Wound Assessment Tool (BWAT)*.
2. Sasaran pengguna aplikasi adalah perawat.
3. Luka yang dikaji hanya luka kronis.
4. Aplikasi yang dibuat berbasis sistem operasi *Android*.
5. Model pengembangan yang digunakan untuk mengembangkan aplikasi adalah *Scrum*.

## E. Tujuan Penelitian

Mengacu pada latar belakang yang telah disampaikan, penelitian yang akan dilaksanakan bertujuan untuk merancang aplikasi pengkajian luka kronis berbasis *Android* modul pengolahan citra versi pertama.

## F. Manfaat Penelitian

Manfaat yang diharapkan dari hasil penelitian yang akan dilaksanakan adalah sebagai berikut:

1. Bagi Penulis
  - a. Untuk memenuhi syarat kelulusan Strata Satu (S1) Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Jakarta.
  - b. Dapat mengimplementasikan ilmu-ilmu yang didapat di bangku kuliah untuk merancang suatu aplikasi.
2. Bagi Program Studi Ilmu Komputer
  - a. Dapat memanfaatkan data *ground truth* dan pengkajian luka yang telah dikumpulkan sebagai bahan untuk penelitian di bidang pengkajian luka.
  - b. Menjadi pembuka untuk penelitian selanjutnya.
3. Bagi Perawat
  - a. Terciptanya sebuah aplikasi pengkajian luka kronis sebagai pendukung pelaksanaan perawatan luka.
  - b. Memberikan kemudahan akses data, kontrol dokumen dan penyimpanan data pengkajian luka kronis yang terorganisir.

## BAB II

### KAJIAN TEORI

#### A. Pengkajian Luka

Dalam upaya untuk mendukung proses penyembuhan luka, perawat harus mampu memilih balutan yang tepat. Pengkajian luka menjadi dasar yang diterapkan dalam pemilihan balutan luka yang baik dan benar. Oleh karena itu, perlakuan dengan sistematis dan komprehensif diperlukan untuk pengkajian luka. Informasi yang penting mengenai pasien dan luka, penentuan program perawatan luka yang akan diberikan, dan evaluasi keberhasilan perawat juga bisa didapatkan dari pelaksanaan pengkajian luka.

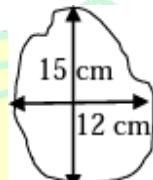
Pada pelaksanaan pengkajian luka, terlebih dahulu perawat harus mencuci luka tersebut dan mengevaluasi setiap pasien terhadap:

- a. Hal yang menyebabkan luka (tekanan, trauma, *insufisiensi vena*, dan *diabetes*).
- b. Riwayat pengkajian luka terakhir.
- c. Usia pasien.
- d. Durasi luka: akut (<12 minggu) atau kronis (>12 minggu).
- e. Kecukupan saturasi oksigen.
- f. Mengidentifikasi faktor-faktor *systemic* yang memberi pengaruh pada proses sembahunya luka. Seperti: obat-obatan dan data laboratorium.
- g. Identifikasi adanya penyakit akut dan kronis, misalnya anemia berat, penyakit vaskuler perifer, penyakit jantung, sepsis, diabetes, gagal ginjal, gangguan pernafasan yang berbahaya, dehidrasi, dan malnutrisi atau *cachexia*.

Bates-Jensen Wound Assessment Tool (BWAT) merupakan salah satu instrumen skalar pengkajian luka yang dikembangkan dan dipakai untuk mempelajari keadaan luka yang kronis pada luka tekan. Nilai yang diperoleh dari skalar ini menjelaskan status keparahan luka (Bates-Jensen et al., 2019). BWAT memiliki tiga belas kategori pengkajian luka yang mempunyai skala 1-60. Apabila semakin besarnilainya, maka semakin parah kondisi lukanya. Adapun kriteria

pengkajian luka berdasarkan Bates-Jensen Wound Assessment Tool (BWAT) adalah sebagai berikut:

### 1. Ukuran Luka



Gambar 2.1 Contoh pengukuran ukuran luka

Sumber: Modul Perawatan Luka (Aminuddin et al., 2020)

Dimensi ukuran luka terdiri dari panjang, lebar, dan diameter permukaan luka. Penggaris dapat digunakan untuk melakukan pengukuran panjang dan lebar luka dalam *centimeter* yang kemudian dikalikan untuk mendapat luasnya. Direkomendasikan untuk menjiplak keliling luka di atas plastik transparan atau *asetat sheet* dengan menggunakan spidol. Penilaian terhadap ukuran luka terbagi atas lima skor di antaranya:

- 1 = luas  $< 4\text{cm}^2$
- 2 = luas  $4\text{cm}^2 \leq 16\text{cm}^2$
- 3 = luas  $16.1\text{cm}^2 \leq 36\text{cm}^2$
- 4 = luas  $36\text{cm}^2 \leq 80\text{cm}^2$
- 5 = luas  $> 80\text{cm}^2$

### 2. Kedalaman Luka

Penilaian kedalaman luka ditentukan dengan memilih ketebalan dan kedalaman yang paling sesuai dengan luka berdasarkan kriteria penilaian berikut:

- 1 = *Eritema* yang tidak memucat pada kulit yang utuh.
- 2 = Hilangnya beberapa ketebalan kulit yang melibatkan epidermis dan/atau dermis.
- 3 = Hilangnya semua ketebalan kulit yang membawa kerusakan atau nekrosis jaringan subkutan mampu meluas ke bawah, tetapi tidak melewati fasia di bawahnya dan/atau campuran sebagian dan ketebalan

penuh dan/atau lapisan jaringan tertutup oleh jaringan granulasi.

- 4 = Visualisasi lapisan jaringan tidak mungkin karena terkaburkan oleh nekrosis.
- 5 = Hilangnya semua ketebalan kulit dengan area rusak yang luas, nekrosis jaringan atau kerusakan pada otot, tulang atau struktur pendukung.

### 3. Tepi Luka

Indikasi tepi luka dilihat dari jelas atau tidaknya tepi luka tersebut. Adapun jenis-jenis tepi luka di antaranya adalah sebagai berikut:

- *Indistinct, diffuse* : tidak dapat membedakan garis luka dengan jelas.
- *Attached* : rata dengan dasar luka, tidak terdapat sisi atau dinding.
- *Not Attached* : terdapat sisi atau dinding, dasar luka lebih dalam dari tepi.
- *Rolled Under, thickened* : antara lembut, kaku, dan fleksibel untuk disentuh
- Hiperkeratosis : terbentuknya jaringan seperti kapalan di sekitar luka dan di tepinya.
- *Fibrotic, scarred* : keras dan kaku ketika disentuh.

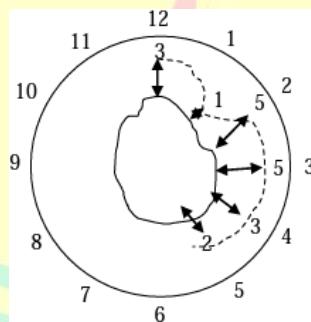
Adapun lima kondisi penilaian berdasarkan BWAT sebagai berikut:

- 1 = *Indictinct, diffuse*, tidak terlalu terlihat.
- 2 = Jelas, terlihat jelas garis besarnya, melekat, bahkan dengan dasar luka.
- 3 = Hilangnya semua ketebalan kulit yang membawa kerusakan atau nekrosis jaringan subkutan dapat menyebar ke bawah. Namun, tidak melewati fasia di bawahnya dan/atau campuran sebagian dan ketebalan penuh dan/atau lapisan jaringan tertutup oleh jaringan granulasi.
- 4 = Visualisasi lapisan jaringan tidak mungkin karena terkaburkan oleh nekrosis.
- 5 = Hilangnya semua ketebalan kulit dengan area rusak yang luas,

nekrosis jaringan atau kerusakan pada otot, tulang atau struktur pendukung.

#### 4. Goa Luka

Pengkajian goa luka atau kedalaman luka memerlukan pendekatan tiga dimensi. Instrumen yakni kateter atau aplikator kapas lembab steril dipergunakan sebagai metode pengukuran.



Gambar 2.2 Contoh pengukuran goa luka dengan pendekatan tiga dimensi

Sumber: Modul Perawatan Luka (Aminuddin et al., 2020)

Saat mengukur goa luka dilakukan dengan memegang aplikator menggunakan ibu jari dan jari telunjuk pada titik yang sesuai dengan tepi luka. Tarik perlahan aplikator sambil menjaga letak ibu jari dan jari telunjuk menahannya, kemudian ukur alat dari ujung pada posisi sejajar dengan penggaris dalam sentimeter (cm) dan luka diukur seperti menghadap ke arah jam, Bagian atas luka (pukul 12) mengarah ke kepala pasien, dan bagian bawah luka (pukul 6) mengarah ke kaki pasien. Panjang goa dapat diukur dari "12 jam hingga 6 jam". Lebar goa dapat diukur dari sisi ke sisi atau dari jam 3 sampai jam 9.

Pada instrumen BWAT, pengkajian goa luka memiliki lima skala penilaian, yaitu:

- 1 = Tidak terdapat goa luka.
- 2 = Goa < 2cm di area mana pun.
- 3 = Goa luka 2-4cm melibatkan < 50% margin luka.
- 4 = Goa luka 2-4cm melibatkan > 50% margin luka.
- 5 = Goa luka > 4cm di area mana pun.

## 5. Tipe Jaringan Nekrotik

Jaringan nekrotik (mati) yang terdapat pada luka kronis dapat menghambat penyembuhan luka, seperti pada instrumen pengkajian luka BWAT, jaringan nekrotik dihitung berdasarkan jenis jaringan nekrotik yang dominan pada luka menurut warna, konsistensi, dan perlekatan menggunakan panduan di bawah ini:

- 1 = Tidak terdapat jaringan nekrotik.
- 2 = Jaringan putih/abu-abu yang tidak dapat hidup dan/atau kulit kuning yang tidak melekat.
- 3 = Cangkang kuning yang melekat longgar.
- 4 = Lengket, lembut, terdapat eschar hitam.
- 5 = Sangat melekat, keras, terdapat eschar hitam.

## 6. Jumlah Jaringan Nekrotik

Jumlah jaringan nekrotik pada luka juga menjadi salah satu kategori untuk mengkaji luka, dimana semakin sedikit jumlah jaringan nekrotik, maka penyembuhan luka semakin baik. Adapun berdasarkan BWAT dinilai dengan lima kondisi, di antaranya:

- 1 = Tidak terdapat jaringan nekrotik.
- 2 = < 25% dari dasar luka tertutup.
- 3 = 25% sampai 50% dasar luka tertutup.
- 4 = > 50% dan < 75% dasar luka tertutup.
- 5 = 75% sampai 100% dasar luka tertutup.

## 7. Tipe Eksudat

Eksudat adalah cairan yang diproduksi oleh luka akut maupun kronis.

Eksudat memiliki beberapa tipe, di antaranya:

- *Bloody* = tipis, berwarna merah terang.
- *Serosanguineous* = tipis, cair berwarna merah hingga merah muda.
- *Serous* = bening, cair, dan tipis.
- *Purulent* = tipis atau tebal, berwarna coklat kekuningan.

- *Foul Purulent* = tebal, kuning buram hingga hijau dengan bau yang menyengat.

Berdasarkan BWAT, penilaian kondisi tipe eksudat dinilai dengan lima skala, di antaranya:

- 1 = Tidak terdapat eksudat.
- 2 = *Bloody*.
- 3 = *Serosanguineous*.
- 4 = *Serous*.
- 5 = *Purulent*.

#### 8. Jumlah Eksudat

Produksi eksudat yang berlebih akan meningkatkan risiko infeksi dan merusak kulit sekitar luka. Oleh karena itu, jumlah eksudat termasuk dalam kategori pengkajian luka. Berikut adalah lima kondisi penilaian berdasarkan jumlah eksudat:

- 1 = Tidak terdapat eksudat, luka kering.
- 2 = Terdapat sedikit, luka lembab namun tidak ada eksudat yang terlihat.
- 3 = Kecil.
- 4 = Sedang.
- 5 = Banyak.

#### 9. Warna Kulit Keliling Luka

Pemeriksaan kulit keliling luka menentukan terdapatnya selulitis, edema, benda asing, eksim, dermatitis kontak atau maserasi. Pada tahap ini dilakukan pengkajian vaskularisasi jaringan sekitar dan pencatatan batas-batas, warna, kehangatan, dan waktu pengisian kapiler jika luka mendapat tekanan atau kompresi.

Terdapat lima kondisi warna kulit keliling luka, diantaranya:

- 1 = Merah muda atau normal untuk grup etnis.
- 2 = Merang terah dan/atau pucat ketika disentuh.

- 3 = Pucat putih atau abu-abu atau hipopigmentasi.
- 4 = Merah tua atau ungu dan/atau tidak pucat.
- 5 = Hitam atau hiperpigmentasi.

#### 10. Peripheral Tissue Edema

Kondisi dimana jaringan membengkak dalam tubuh akibat adanya cairan yang menumpuk disebut dengan Edema. Komponen pengkajian edema di antaranya sebagai berikut:

- 1 = Tidak ada pembengkakan atau edema.
- 2 = Edema non-*pitting* meluas <4 cm di sekitar luka.
- 3 = Edema non-*pitting* meluas >4 cm di sekitar luka.
- 4 = *Pitting* edema meluas <4 cm di sekitar luka.
- 5 = Krepitus dan/atau *pitting* edema meluas >4 cm di sekitar luka.

#### 11. Peripheral Tissue Induration

*Induration* adalah pengerasan lokal jaringan tubuh. Area menjadi kencang, namun tidak sek keras tulang. Penilaian dilakukan dengan mencubit lembut jaringan. Hasil indurasi dalam ketidakmampuan untuk mencubit jaringan. Penetuan seberapa jauh endema atau indurasi melampaui luka dapat dilakukan dengan menggunakan panduan pengukuran metrik transparan.

- 1 = Tidak terdapat jaringan indurasi.
- 2 = Terdapat indurasi < 2cm disekitar luka.
- 3 = Indurasi 2-4 cm memanjang <50% di sekitar luka.
- 4 = Indurasi 2-4 cm memanjang > 50% di sekitar luka.
- 5 = Indurasi > 4 cm di area sekitar luka.

## 12. Jaringan Granulasi

Pembuluh darah kecil yang bertumbuh dan jaringan ikat yang mengisi seluruh ketebalan luka disebut dengan jaringan granulasi. Jaringan dikatakan sehat apabila memiliki warna cerah, merah daging, mengkilap, dan granular dengan penampilan beludru. Pasokan vaskular yang tidak baik muncul sebagai merah muda pucat atau pucat sampai kusam, dan warna merah kehitaman. Adapun *scoring*-nya adalah sebagai berikut:

- 1 = Kulit utuh atau luka ketebalan sebagian.
- 2 = Merah cerah dan merah daging; 75% hingga 100% dari luka yang terisi dan/atau pertumbuhan berlebih jaringan.
- 3 = Merah cerah dan merah daging; < 75% & > 25% luka terisi.
- 4 = Merah muda, dan/atau kusam, merah kehitaman dan/atau mengisi < 25% luka.
- 5 = Tidak ada jaringan granulasi.

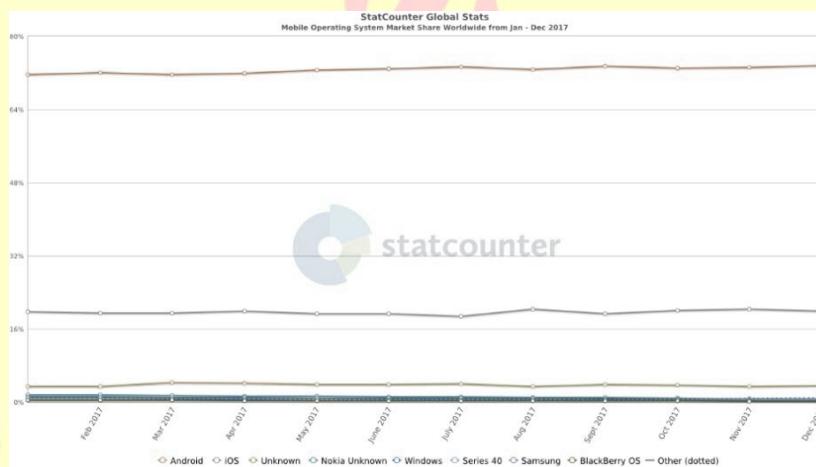
## 13. Epitelisasi

Epidermis yang mengalami proses pelapisan ulang dan muncul sebagai kulit merah muda atau merah disebut dengan epitelisasi. Epitelasi dapat terjadi diseluruh dasar luka serta dari tepi luka, pada luka dengan ketebalan parsial. Sedangkan, pada luka dengan kondisi ketebalan penuh hanya dari bagian tepi saja. Penentuan persentase luka yang terlibat dan pengukuran jarak jaringan epitel yang meluas ke dalam luka dapat menggunakan panduan pengukuran metrik transparan dengan lingkaran konsentris yang terbagi empat (25%) kuadran berbentuk pai.

- 1 = 100% luka tertutup, permukaan utuh
- 2 = 75% hingga <100% luka tertutup &/atau jaringan epitel meluas >0,5cm ke dasar luka
- 3 = 50% hingga <75% luka tertutup &/atau jaringan epitel meluas hingga <0,5 cm ke dasar luka
- 4 = 25% hingga <50% luka tertutup
- 5 = < 25% luka tertutup

## B. Pengantar Android

Open Handset Alliance, anak perusahaan Google, membentuk dan mengembangkan suatu sistem operasi *open source* dengan basis Linux yang dikenal dengan istilah *Android*. Karena sifatnya yang *open source*, pertumbuhan penelitian yang dilakukan dengan sistem operasi *Android* sangat pesat (KOCAKOYUN, 2017). Pada tahun 2017 jumlah aplikasi *Android* pada Google Play Store mencapai 3,7 juta aplikasi (York, 2018), dan menjadikan *Android* sebagai salah satu pemimpin pasar *mobile*. Hal tersebut terbukti dengan adanya data dari StatCounter dimana pengguna *Android* mencapai 71,73% (Gambar 2.3).



Gambar 2.3 Data statistik pengguna Android

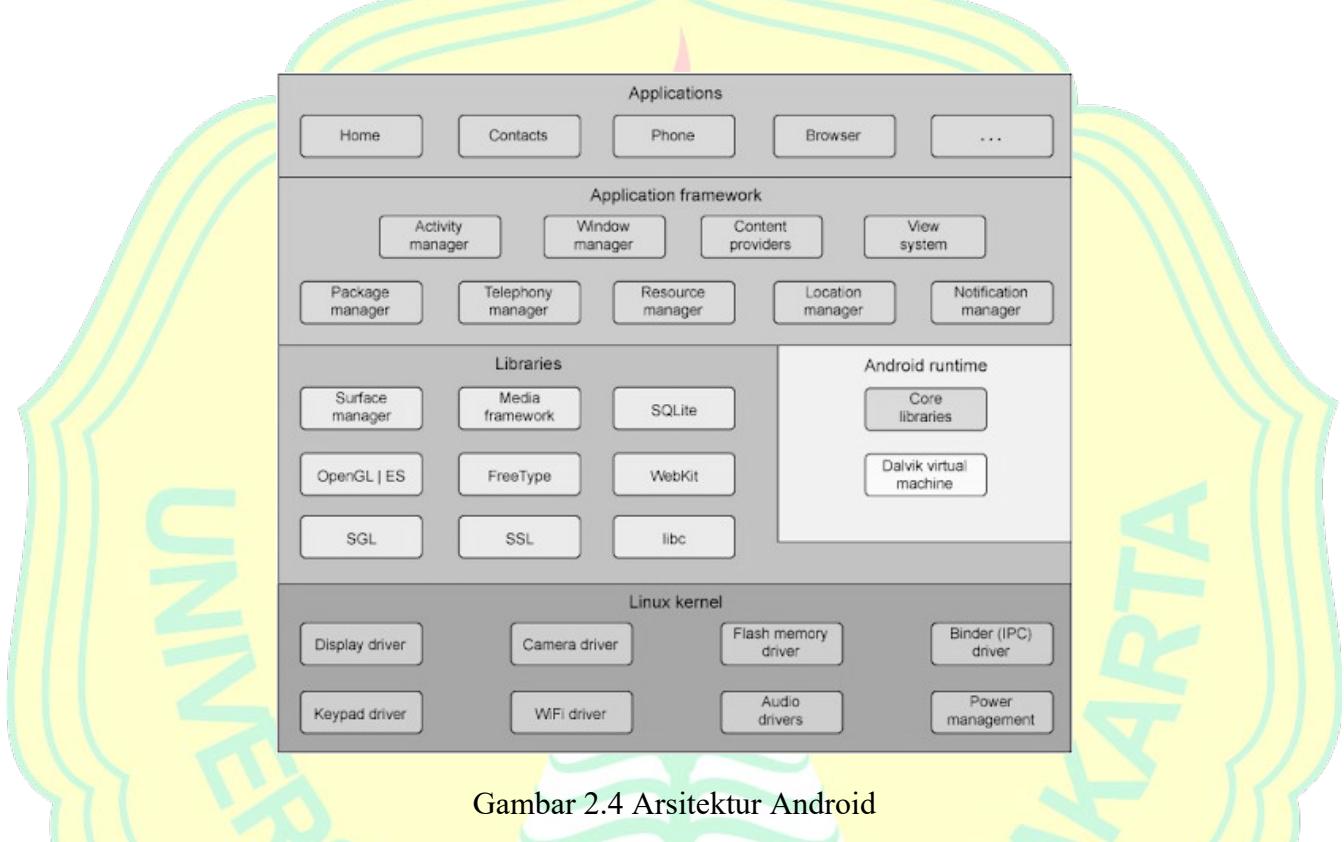
Sumber: <https://gs.statcounter.com/os-market-share/mobile/worldwide>

Sistem operasi *Android* dapat berjalan pada banyak perangkat yang memiliki ukuran layar dan resolusi yang berbeda-beda. Namun, Google hanya mengizinkan aplikasi untuk berjalan pada perangkat yang kompatibel. Agar perangkat *Android* tersertifikasi kompatibel, perangkat tersebut harus memiliki beberapa perangkat keras seperti kamera, kompas, fitur *Global Positioning System* (GPS) dan fitur *Bluetooth*. Selain dapat berjalan di semua perangkat yang memiliki layar dan resolusi berbeda, *Android* juga memiliki banyak *Application Programming Interfaces* (APIs) yang memudahkan para pengembang untuk mengembangkan aplikasi berfitur lengkap dalam waktu relatif singkat. Selain itu, pengembang juga dapat memanfaatkan perangkat keras yang terdapat pada *Android*.

seperti kamera dan kompas sebagai pendukung fitur aplikasi yang akan dibuat.

### 1. Arsitektur *Android*

*Android* merupakan sistem operasi dengan basis kernel Linux dan juga sistem berlapis. Lapisan arsitektur *Android* dapat dilihat pada Gambar 2.4 dimana dibagi atas lapisan *Applications*, lapisan *Application framework*, lapisan *Libraries*, lapisan *Android runtime*, dan kernel Linux.



Gambar 2.4 Arsitektur *Android*

Sumber: [https://www.tutorialspoint.com/android/android\\_architecture.htm](https://www.tutorialspoint.com/android/android_architecture.htm)

- *Lapisan Applications*

Lapisan ini berada pada lapisan teratas dan berhubungan secara langsung dengan aplikasi-aplikasi yang berjalan pada *Android*. Jika pengguna ingin menginstal aplikasi ataupun membuat aplikasi, proses tersebut terjadi pada lapisan ini.

- *Lapisan Application Framework*

Lapisan *Application Framework* merupakan lapisan yang menerjemahkan *framework API Android*. Komponen-komponen API *framework* yang terdapat pada lapisan ini dapat dengan mudah diakses serta dimodifikasi oleh para pengembang aplikasi sesuai dengan

kebutuhan mereka. *Application Framework* memudahkan pembuat aplikasi untuk memahami fungsi tiap komponennya. Adapun *framework* yang dipakai dalam penelitian yang akan dilakukan adalah sebagai berikut:

- *Android Graphics*

*Android Graphics Framework* menyediakan metode yang dapat dipakai untuk menghasilkan aplikasi menggambar. Ada dua *class* yang berperan utama yaitu *Canvas* dan *Paint*. Beberapa metode pada *class* *Canvas* dan *Paint* yang digunakan pada penelitian ini adalah sebagai berikut:

Tabel 2.1 Metode Class Canvas dan Paint

Class	Metode	Deskripsi
Canvas	Canvas()	Membuat kanvas kosong.
	Canvas(Bitmap bitmap)	Membuat kanvas dengan bitmap yang terspesifikasi.
	drawRGB(int r, int g, int b)	Mewarnai seluruh kanvas dengan kode warna RGB.
	drawLine(float startX, float startY, float stopX, float stopY, Paint paint)	Menggambar segmen garis dengan spesifikasi koordinat titik mulai (x,y) dan titik selesai dengan <i>style</i> <i>paint</i> yang telah ditentukan.
	drawPath(Path path, Paint paint)	Menggambar <i>path</i> terspesifikasi dengan <i>style</i> <i>paint</i> yang telah ditentukan.
	enableZ()	Mengaktifkan <i>support</i> koordinat Z yang untuk memungkinkan <i>layering</i> .

Class	Metode	Deskripsi
	save()	Menyimpan <i>state</i> pada Canvas.
	Restore()	Mengembalikan <i>state</i> pada Canvas yang telah disimpan sebelumnya.
Paint	Paint(Paint paint)	Menginisialisasikan atribut Paint yang akan digunakan.
	setColor(int color)	Mendefinisikan warna yang akan dipakai.
Paint	setStrokeWidth(float width)	Mengatur lebar garis yang digunakan untuk menggambar.
	setStyle (Paint.Style style)	Menentukan apakah primitif yang digambar terisi dengan warna, hanya garis, atau keduanya (dalam warna yang sama).

- *Drag and Drop*

*Framework drag/drop* memungkinkan pengguna untuk memindahkan data menggunakan gestur grafis tarik dan lepas. Hal tersebut dapat dilakukan dari satu *view* ke *view* lainnya dalam aplikasi. *View* menerima *event* drag/drop dengan mengimplementasikan *View.OnDragListener* atau dengan metode *callback* *onDragEvent()*. Ketika sistem memanggil *listener* atau menggunakan metode tersebut, maka akan mengembalikan argument *DragEvent*.

- *Touch Event*

*Touch Event* atau *onTouchEvent()* merupakan API yang berfungsi untuk mendeteksi aktivitas sentuhan pada layar pengguna dan mengembalikan nilai koordinat x dan y daripada sentuhan tersebut.

*Class onTouchListener()* dan metode *onTouch* mempunyai akses kepada data *Motion Event* yang mendeskripsikan pergerakan berdasarkan nilai *axis*. Dengan fungsi-fungsi sebagai berikut koordinat didapatkan:

- *getAction()* mengembalikan *Integer* seperti `MotionEvent.ACTION_DOWN`, `MotionEvent.ACTION_MOVE`, `MotionEvent.ACTION_UP`.
  - *getX()* – fungsi ini mengembalikan koordinat X dari aktivitas sentuhan pada layar.
  - *getY()* – fungsi ini mengembalikan koordinat Y dari aktivitas sentuhan pada layar.
- *Android Layouting*

*Layout* atau tampilan tata letak pada *Android*, berfungsi sebagai penata teks, gambar, atau komponen lainnya agar terlihat rapi dan nyaman bagi pengguna. Tampilan pada *Android* diterjemahkan pada file XML. Jenis-jenis dan fungsi *layout* pada *Android* ada berbagai macam, di antaranya:

- *Linear Layout*

*Layout* dengan jenis linear akan menyejajarkan *child-view* dalam vertikal maupun horizontal. Semua anak akan memiliki satu anak per baris. Untuk menetapkan arah vertikal maupun horizontal, dapat ditetapkan pada atribut `android:orientation`.
- *Relative Layout*

*Relative Layout* merupakan jenis tata letak tanpa aturan dan bebas. Sebuah *child-view* dapat diletakkan berada di atas atau di bawah *child-view* lainnya.
- *Constraint Layout*

Tata letak tampilan dengan jenis *Constraint* memiliki sifat yang sama dengan *Relative*. Namun, pada setiap *View* memiliki tali yang terdapat pada titik yang dapat berupa sisi dari *Parent*, *Layout*, *View* lain, ataupun titik bantu.

➤ *Frame Layout*

*Frame Layout* berfungsi untuk membentuk objek yang bertumpukan satu sama lain. Contohnya ketika membentuk tombol di atas *image*.

➤ *Table Layout*

*Table Layout* merupakan tampilan dengan baris dan kolom. Baris dipergunakan untuk penyimpanan satu jenis *record*, hanya satu informasi yang mampu diberikan. Kolom merupakan anggota dari setiap baris, dimana satu baris mampu menampung bermacam jenis kolom.

• *Android Form*

Untuk dapat membuat formulir pada *Android*, diperlukan paduan XML serta Java. XML dibutuhkan untuk membuat tampilan formulir, sedangkan Java dibutuhkan untuk mengambil data isian yang terdapat pada tampilan.

➤ *Isian Teks*

*EditText* adalah elemen yang digunakan untuk mengisi dan memodifikasi teks. Atribut `android:inputType="text"` harus didefinisikan untuk memberitahu bahwa isian berupa teks.

➤ *Radio Button*

*Radio Button* memungkinkan pengguna untuk memilih satu opsi dari beberapa opsi. Form jenis ini digunakan ketika pengguna harus melihat semua opsi yang tersedia berdampingan. Ketika pengguna memilih salah satu dari *Radio Button*, objek *RadioButton* akan mengembalikan *on-click event*. Oleh karena itu, diperlukan *click event handler* dengan memberi tambahan atribut `android:onClick` di dalam elemen `<RadioButton>` pada XML.

➤ *Pickers*

*Android* menyediakan kontrol untuk memilih waktu atau tanggal. Setiap *Picker* memiliki kendali untuk memilah bagian waktu (jam, menit, AM/PM) atau tanggal (bulan, hari, tahun).

Untuk membuat TimePicker atau DatePicker direkomendasikan untuk menggunakan DialogFragment().

- Lapisan Application Library

*Application Library* terbagi atas dua elemen yaitu *Android Runtime* dan *Android Library*. *Android Runtime* mencakup *Core Libraries* dan mesin virtual Java (*Dalvik Virtual Machine*) yang telah dioptimasi oleh Google agar dapat berjalan pada *Android*. Untuk menjalankan fungsi-fungsi *Andorid* diperlukan mesin *Dalcik Virtual Machine* ini. Mesin ini menjalankan *class* hasil kompilasi Java yang telah bertransformasi menjadi format (.dex). *Core Libraries* Java berfungsi untuk menerjemahkan Bahasa Java/C yang digunakan agar dapat berfungsi pada sistem operasi *Android*.

*Android Library* berisi dari C/C++ *libraries* dan *libraries* Java yang berfungsi mendukung *framework Android*. Adapun fungsi dari tiap *libraries* adalah sebagai berikut:

- 1) *Media framework*: *library* media mendukung untuk memutar dan merekam format audio dan video.
- 2) *Surface manager*: bertanggung-jawab untuk menjadi pengelola akses ke subsistem tampilan.
- 3) *SGL & OpenGL*: keduanya merupakan *cross-platform API* yang digunakan untuk mendukung grafis dua dimensi dan tiga dimensi.
- 4) *SQLite*: menyediakan dukungan untuk *database* dan *FreeType* untuk mendukung *font*.
- 5) *Web-Kit*: merupakan *browser* yang bersifat *open source* dengan fungsi menampilkan konten *website* dan juga untuk menyederhanakan pemuatan suatu laman.
- 6) *Secure Sockets Layer (SSL)*: merupakan teknologi keamanan untuk menciptakan tautan terenkripsi antara *web server* dan *web browser*.

- Lapisan kernel Linux

Lapisan kernel Linux terdapat di paling bawah dan merupakan jantung dari arsitektur *Android*. Fungsi dari lapisan ini adalah untuk mengelola *driver* yang ada seperti *display drivers*, *camera drivers*, *Bluetooth drivers*, *audio drivers*, *memory drivers*, dan lain-lain yang diperlukan selama *runtime*.

Fitur-fitur yang ada pada kernel Linux adalah sebagai berikut:

- 1) *Security*: kernel Linux yang berfungsi untuk menangani keamanan antara aplikasi dengan sistem.
- 2) *Memory Management*: kernel Linux yang menangani pengelolaan memori dengan baik sehingga mendukung kebebasan untuk pengembangan aplikasi.
- 3) *Process Management*: mengalokasikan *resources* ke pemrosesan kapanpun dibutuhkan.
- 4) *Network Stack*: secara efektif menangani komunikasi *network*.
- 5) *Driver Model*: kernel Linux memastikan setiap aplikasi berjalan dengan baik pada perangkat.

## 2. *Android Software Development Kit (SDK)*

Pengembangan aplikasi pada sistem operasi *Android* menggunakan *kit* yang umumnya disebut *Android SDK*. *Android SDK* terdiri atas beberapa *tools*; yakni *debugger*, *software libraries*, *emulator*, dokumentasi, *sample code*, dan tutorial.

Contoh dari *Android SDK* salah satunya yaitu Java SE *Development Kit* yang juga merupakan bahasa pemrograman paling sering digunakan dalam pengembangan aplikasi pada *Android*. Selain Java, terdapat bermacam bahasa pemrograman lainnya seperti C++, Go, dan Kotlin yang diterapkan oleh Google pada tahun 2017.

## 3. *Android Studio Integrated Development Environment (IDE)*

*Android Studio* adalah *Integrated Development Environment (IDE)* resmi untuk mengembangkan aplikasi *Android*. IDE *Android Studio* dibuat berdasarkan IntelliJ IDEA dan memiliki banyak fitur yang dapat menambah produktivitas saat

pengembangan aplikasi *Android*.

Sejumlah fitur yang ada pada *Android Studio* di antaranya fleksibilitas sistem versi berbasis Gradle, emulator dengan banyak fitur dan *responsive*, integrasi kode dengan GitHub, dukungan C++ dan NDK, *Instant Run* untuk membuat perubahan tanpa membuat APK baru dan alat pengujian serta kerangka kerja yang ekstensif.

### C. Pengantar Flask

*Flask* merupakan *web framework* yang didasari oleh bahasa pemrograman Python. *Web framework* merupakan koleksi dari modul dan *packages* yang memungkinkan pengembang untuk membuat aplikasi web atau *web service* tanpa harus membuat detail-detail dasar seperti protokol, soket, atau manajemen proses.

*Flask* memiliki *core* yang sederhana dan bersifat ringan, selain itu *Flask* juga memiliki sifat *simplicity* dan *flexibility* sehingga pengembangan dapat disesuaikan dengan kebutuhan oleh penambahan ekstensi yang tersedia. Salah satu ekstensi yang dimiliki oleh *Flask* adalah Blueprint. Blueprint berfungsi untuk mempermudah pembuatan pengaturan minimal RESTful APIs. RESTful APIs adalah layanan atau metode yang digunakan untuk metransmisikan data dengan menggunakan protokol HTTP.

*Routing* pada *Flask* dapat didefinisikan dengan bantuan ekstensi Blueprint yang mempermudah akses ke beberapa metode *Hypertext Transfer Protocol* (HTTP) hanya dengan mendefinisikan metode yang dipakai pada *routing* yang digunakan. Metode permintaan HTTP yang dapat dipergunakan antara lain sebagai berikut:

Tabel 2.2 Fungsi-fungsi Metode HTTP

Metode HTTP	Fungsi
GET	Menerima informasi dari <i>server</i> yang diberikan menggunakan URI yang spesifik. Permintaan menggunakan metode GET hanya menerima data tanpa adanya efek perubahan pada data.

Metode HTTP	Fungsi
POST	Mengirimkan data ke <i>server</i> seperti unggahan <i>file</i> , informasi pelanggan dan lain-lain menggunakan form HTML.
HEAD	Sama seperti metode GET, namun hanya memberikan data status dan seksi <i>header</i> saja.
PUT	Mengganti semua representasi dari target <i>resource</i> dengan konten yang diunggah.
DELETE	Menghapus semua representasi dari target yang didefinisikan pada URL.

```
import functools

from flask import (
    Blueprint, flash, g, redirect, render_template, request, session, url_for
)
from werkzeug.security import check_password_hash, generate_password_hash

from flaskr.db import get_db

bp = Blueprint('auth', __name__, url_prefix='/auth')
```

Gambar 2.5 Contoh registrasi Blueprint pada dokumen “auth.py”

Sumber: <https://flask.palletsprojects.com/en/2.1.x/tutorial/views/>

Kode pada gambar di atas berfungsi untuk menerjemahkan penggunaan ekstensi Blueprint pada suatu dokumen dengan nama ‘auth’. Agar routing dapat berjalan maka perlu di registrasikan pada dokumen *init.py* yang merupakan tempat Flask akan berjalan.

```

def create_app():
    app = ...
    # existing code omitted

    from . import auth
    app.register_blueprint(auth.bp)

    return app

```

Gambar 2.6 Dokumen\_init\_.py

Sumber: <https://flask.palletsprojects.com/en/2.1.x/tutorial/views/>

```

@bp.route('/login', methods=('GET', 'POST'))
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        db = get_db()
        error = None
        user = db.execute(
            'SELECT * FROM user WHERE username = ?',
            (username,)
        ).fetchone()

        if user is None:
            error = 'Incorrect username.'
        elif not check_password_hash(user['password'], password):
            error = 'Incorrect password.'

        if error is None:
            session.clear()
            session['user_id'] = user['id']
            return redirect(url_for('index'))

        flash(error)

    return render_template('auth/login.html')

```

Gambar 2.7 Contoh pembuatan routing dengan Blueprint

Sumber: <https://flask.palletsprojects.com/en/2.1.x/tutorial/views/>

Gambar 2.7 merupakan contoh dari pembuatan *routing* untuk *log in* dengan URL routing “/login” yang mendefinisikan metode GET dan POST. Jika proses *log in* berhasil maka akan diarahkan menuju URL “auth/login.html”.

#### D. MongoDB

*MongoDB* merupakan basis data yang menerapkan konsep *Not Only SQL* (*NoSQL*) yang menyimpan data berorientasikan dokumen. *NoSQL* tidak memiliki sistem tabular dan mempunyai penyimpanan yang berbeda dari tabel relasional. *Database* dengan konsep *NoSQL* memungkinkan pengembang untuk menyimpan struktur data dalam jumlah besar dan memberikan fleksibilitas kepada mereka.

Kunci perbedaan NoSQL dengan *Relational Database Management System* (RDBMS) adalah bagaimana sebuah data dimodelkan di dalam *database*. Pemodelan pada RDBMS masih menggunakan tabel berstruktur dengan setiap kolom baris bersifat tetap antara satu dengan lainnya, sedangkan pemodelan data pada NoSQL, khususnya pada MongoDB, menggunakan dokumen dimana setiap barisnya mempunyai kolom yang dapat berbeda dengan baris yang lain. Adapun contoh perbedaan basis data berkonsep RDBMS dan NoSQL dapat dilihat pada Gambar (Pemodelan RDBMS) dan

<b>Users</b>				
ID	first_name	last_name	cell	city
1	Leslie	Yepp	8125552344	Pawnee
<b>Hobbies</b>				
ID	user_id	hobby		
10	1	scrapbooking		
11	1	eating waffles		
12	1	working		

Gambar 2.8 Pemodelan RDBMS

Sumber: <https://www.mongodb.com/nosql-explained/>

```
{
  "_id": 1,
  "first_name": "Leslie",
  "last_name": "Yepp",
  "cell": "8125552344",
  "city": "Pawnee",
  "hobbies": ["scrapbooking", "eating waffles", "working"]
}
```

Gambar 2.9 Pemodelan NoSQL

Sumber: <https://www.mongodb.com/nosql-explained/>

Berdasarkan Gambar 2.8, untuk menyimpan data *user* dan data *hobbies* diperlukan dua tabel terpisah dimana hal ini tidak terjadi pada pemodelan NoSQL (Gambar 2.9) yang menggabungkan dua data *user* dan *hobbies* pada satu dokumen

serta baris yang sama. Dengan NoSQL ketika ingin mengambil dua data tersebut secara bersamaan hanya dibutuhkan satu dokumen saja tanpa adanya *joins*, yang menghasilkan *queries* yang lebih cepat dibandingkan dengan RDBMS.

Meskipun bersifat NoSQL, perancangan database tetap dilakukan. Rancangan database memungkinkan kita untuk mengkonfirmasi dan mendokumentasikan pemahaman terhadap *database* serta memastikan orang melihat lanskap informasi dengan cara yang sama. Dengan kata lain, rancangan database merupakan alat komunikasi (Hoberman, S., 2014).

### 1. Integrasi MongoDB dan Flask

*Database MongoDB* dapat diintegrasikan dengan *framework* *Flask* menggunakan ekstensi yang tersedia salah satunya adalah PyMongo. PyMongo memiliki perintah yang mirip dengan perintah CLI MongoDB diantaranya membuat data, mengakses data, dan memodifikasi data. Untuk mengintegrasikan *Flask* dan *MongoDB* perlu menginisialisai projek *Flask* dan mengimpor ekstensi Flask-PyMongo terlebih dahulu.

```
from flask import Flask
from flask_pymongo import PyMongo

app = Flask(__name__)
app.config["MONGO_URI"] = "mongodb://localhost:27017/myDatabase"
mongo = PyMongo(app)
```

Gambar 2.10 Menambahkan Ekstensi PyMongo pada Flask

Sumber: Dokumentasi PyMongo, <https://flask-pymongo.readthedocs.io/en/latest/>

Inisialisasi *MongoDB* pada projek *Flask* dilakukan dengan menggunakan konstruktor PyMongo yang menerima objek *app* *Flask* dan URI string dari *database MongoDB*. Adapun fungsi-fungsi yang dapat kita lakukan setelah *Flask* dan *MongoDB* terintegrasi adalah sebagai berikut:

- Membuat Dokumen

Untuk menambahkan data ke dalam *Database* metode PyMongo yang digunakan adalah `db.collection.insert_one()` jika hanya satu data dan `db.collection.insert_many()` jika terdapat lebih dari satu data. Ketika menambahkan dokumen ke dalam koleksi *MongoDB*, diperlukan untuk mendefinisikan *dictionary* yang terdiri atas *fields* dan *values*.

```
@bp.route("/add_many")
def add_many():
    db.collection.insert_many([
        {'_id': 1, 'judul': "todo title one ", 'desc': "desc body one "},
        {'_id': 2, 'judul': "todo title two", 'desc': "desc body two"},
        {'_id': 3, 'judul': "todo title three", 'desc': "desc body three"},
        {'_id': 4, 'judul': "todo title four", 'desc': "desc body four"},
        {'_id': 5, 'judul': "todo title five", 'desc': "desc body five"},
        {'_id': 1, 'judul': "todo title six", 'desc': "desc body six"},
    ])
    return flask.jsonify('message':True)
```

Ketika menerjemahkan lebih dari satu data yang sama *BulkWriteError* akan muncul, artinya hanya akan ada satu data yang terekam dan data selanjutnya yang sama akan hilang. Untuk menghindari hal tersebut, parameter *ordered* pada fungsi `insert_many()` harus diterjemahkan sebagai *false* kemudian mengangkap eksepsi *BulkWriteError*.

- Membaca Dokumen

Flask-PyMongo menyediakan beberapa metode untuk menerima data dari *database*. Penerimaan semua dokumen dari koleksi digunakan metode `find()` untuk menerima semua data di database dan `find_one()` untuk menerima satu data sesuai dengan ID yang diberikan. Metode `find()` dapat menerima parameter yang digunakan sebagai filter. Parameter filter yang digunakan merepresentasikan diksi yang menerjemahkan properti yang akan dicari.

- Memperbaharui dan Mengganti Dokumen

Metode yang digunakan untuk memperbaharui data dalam *database* adalah `update_one()` atau `replace_one()`. Metode `replace_one()` memiliki beberapa argumen sebagai berikut:

- 1) Filter: berupa *query* yang mendefinisikan data ID yang akan diganti,
  - 2) *Replacement*: berupa data yang akan menggantikan data yang dihapus.
  - 3) *Upsert*: merupakan opsi Boolean yang jika dijadikan sebagai *True* dapat membuat dokumen baru jika tidak ada target dokumen yang dimaksud.
- Menghapus Dokumen

Untuk menghapus satu atau lebih koleksi *database*, PyMongo menyediakan dua metode yaitu `delete_one()`, untuk menghapus satu koleksi dan `delete_many()` untuk menghapus beberapa koleksi.

```
@bp.route("/delete_todo/<int:ID>", methods=['DELETE'])
def delete_todo(ID):
    todo = db.todos.delete_one({'_id': 1})
    return todo.raw_results
```

Pada contoh kode di atas ketika menjalankan *request* seperti [http://localhost:5000/delete\\_todo/5](http://localhost:5000/delete_todo/5) PyMongo akan mencari entri berdasarkan ID yang diberikan dan menghapusnya.

- Menyimpan dan Menerima *Files*

*MongoDB* memungkinkan pengembang untuk menyimpan data biner ke dalam *database* menggunakan spesifikasi GridFS. Ekstensi Flask- PyMmongo menyediakan metode `save_file()` untuk menyimpan *file* ke GridFS dan metode `send_file()` untuk menerima *file* dari GridFS.

```
@bp.route("/uploads/<filename>", methods=["POST"])
def save_upload(filename):
    mongo.save_file(filename, request.files["file"])
    return redirect(url_for("get_upload", filename=filename))
```

Pada contoh kode di atas, dibuat form untuk menangani unggahan *file* dan mengembalikan nama *file* yang telah terunggah.

## E. Scrum

*Scrum* adalah struktur kerja yang dapat digunakan dalam pengembangan produk. *Scrum* pertama kali diumumkan oleh Ken Schwaber pada tahun 1995 di konferensi Austin, namun fondasi daripada metode *Scrum* sudah ada sejak tahun 1980 (Ozierańska *et al.*, 2016; Rubin, 2013). *Scrum* dibuat berdasarkan empirisme yang dicapai dengan beberapa kualitas. Berdasarkan hasil survei dari literatur, kualitas yang membangun empirisme *Scrum* adalah kejelasan dari setiap proses, inspeksi untuk mendeteksi masalah dan adaptasi terhadap perubahan (Schwaber & Sutherland, 2020).

Dalam kerangka kerja *Scrum*, tiap produk ditransmisikan dengan cara yang fleksibel dan iteratif dimana setiap akhir *Sprint* terdapat produk nyata yang dapat dihantarkan. *Requirement* yang dibutuhkan dalam suatu proyek adalah *Product Backlog* yang diperbarui secara berkala.



Gambar 2.11 Kerangka kerja metode pengembangan Scrum  
Sumber: (Ningrum, 2019)

*Scrum* mempunyai tiga elemen, di antaranya:

### 1. Roles

*Roles* dalam *Scrum* terbagi menjadi empat *role* utama, yaitu:

- Tim *Scrum*

Tim *Scrum* adalah kelompok kecil yang terdiri atas satu *Scrum Master*, satu orang sebagai *Product Owner*, dan Pengembang. Dalam tim *Scrum* tidak terdapat tim kecil ataupun hierarki. Semua anggota tim memiliki kemampuan penting untuk memberikan nilai ke dalam setiap *Sprint* dan berfokus pada satu tujuan di satu waku, *Product*

### *Goal.*

Tim *Scrum* bertanggung-jawab dalam setiap aktivitas produk seperti kolaborasi dengan *stakeholder*, verifikasi, *maintenance*, *operation*, *experimentation*, *research* dan pengembangan. Tim *Scrum* mentransmisikan produk secara *iterative* menggunakan *Sprint*, oleh karena itu tim *Scrum* juga bertanggung-jawab untuk menciptakan nilai pada setiap *Sprint*-nya.

- *Scrum Master*

*Scrum Master* mempunyai tanggung-jawab dalam merealisasikan *Scrum* yang terdefinisi pada panduan *Scrum*. Setiap anggota tim dibantu oleh *Scrum Master* untuk memahami bagaimana teori dan praktik kerangka kerja *Scrum*. Selain itu, menjaga efektivitas dari tim *Scrum* juga menjadi tanggung-jawab mereka.

- *Product Owner*

Tanggung-jawab *Product Owner* adalah untuk meningkatkan nilai komersial produk yang dihasilkan oleh *Development Team* dan mengelola *Product Backlog* agar lebih maksimal. *Product Owner* adalah *role* satu-satunya yang memiliki tanggung-jawab untuk mengelola *Product Backlog*. Adapun pengelolaan *Product Backlog*:

- 1) Penyampaian isi *Product Backlog*.
- 2) Memastikan *Development Team* memahami *Product Backlog*.
- 3) Memastikan isi daripada *Product Backlog* transparan dan jelas bagi seluruh anggota tim.
- 4) Mengurutkan *item* pada *Product Backlog* untuk mencapai tujuan secara optimal.

- *Development Team*

*Development Team* atau Tim Pengembang adalah seorang profesional yang menjalankan isi yang sudah tercantum di dalam *Product Backlog*. Tim Pengembang berkomitmen untuk membuat semua aspek *increment* dapat berfungsi pada setiap *Sprint*. Namun, Tim Pengembang juga selalu bertanggung jawab untuk:

- 1) Membuat rancangan *Sprint* atau dikenal dengan *Sprint*

*Backlog.*

- 2) Membuat definisi penyelesaian sebuah *task*.
- 3) Mengadaptasikan semua *plan* setiap hari sampai *Sprint Goal*.

## 2. *Artifacts*

Artefak *Scrum* dirancang untuk memaksimalkan transparansi informasi utama dan kesempatan untuk menginspeksi dan mengadaptasi.

- *Product Backlog*

*Product Backlog* merupakan kumpulan fitur-fitur yang terdapat pada suatu produk umumnya disebut dengan *User Stories*. *User Stories* dapat ditambahkan, dimodifikasi, atau dihilangkan dari *Product Backlog* selama proyek berlangsung.

- *Sprint Backlog*

Beberapa *User Stories* yang diambil dari *Product Backlog* untuk dijalankan pada satu *Sprint* disebut dengan *Sprint Backlog*. *Sprint Backlog* mencakup semua kegiatan kerja yang diperlukan untuk sampai pada *Sprint Goal*.

Dalam satu *Sprint* terdapat *increment* yang merupakan manifestasi dari *User Stories* yang diselesaikan dan total *increment* dari seluruh *Sprint* sebelumnya.

## 3. *Events*

*Sprint* adalah wadah dari semua *event* yang terdapat di *Scrum*. *Event* dibuat sebagai perwujudan salah satu dari tiga tiang *Scrum*. Semua *event* berjalan pada saat yang bersamaan untuk mengurangi kompleksitas.

- *Sprint*

*Sprint* merupakan komponen utama dari kerangka kerja *Scrum*, dimana sebuah ide menjadi sebuah nilai. Panjang durasi *Sprint* bersifat tetap yaitu satu sampai empat minggu untuk menjaga konsistensi.

*Sprint* berfokus untuk memindahkan beberapa *User Stories* pada *Product Backlog*. Dalam satu *Sprint* terdapat beberapa kegiatan

diantaranya *Sprint Planning*, *Sprint Review* dan *Sprint Retrospective*.

- *Sprint Planning*

Sebelum *Sprint* dimulai, perencanaan yang dilaksanakan saat *Sprint* dilakukan ialah *Sprint Planning* yang dilakukan oleh seluruh anggota Tim *Scrum*. Waktu untuk melaksanakan *Sprint Planning* terbatas dengan durasi paling lama delapan jam. Fungsi dari *Sprint Planning* adalah memutuskan apa yang dapat dimasukkan ke dalam *increment* dari *Sprint* dan bagaimana penyelesaian yang dibutuhkan untuk mengantarkan *increment*.

- *Daily Scrum*

*Daily Scrum* merupakan kegiatan lima belas menit bagi para pengembang untuk memeriksa perkembangan menuju *Sprint Goal* dan menyesuaikan pekerjaan yang akan dikerjakan selama dia puluh empat jam ke depan. Pada kegiatan *Daily Scrum*, hal apa saja yang akan dikerjakan hari ini, apa yang telah dikerjakan kemarin dan hambatan yang telah dialami dalam mencapai *Sprint Goal* akan didiskusikan oleh *Development Team*.

- *Sprint Review*

Tahap ini dilakukan pada akhir *Sprint*, fungsinya untuk mengawasi apa yang telah diselesaikan di *Sprint*. Menurut hasil tinjauan serta perubahan *Product Backlog*, Tim *Scrum* menentukan kembali pekerjaan selanjutnya yang dapat mengoptimalkan produk.

*Sprint Review* bersifat informal dan diselenggarakan dalam durasi empat jam untuk *Sprint* dalam waktu satu bulan. Semakin singkat durasi *Sprint*, maka semakin singkat pula durasi *Sprint Review*.

## F. Unified Modelling Language (UML)

*Unified Modelling Language* (UML) merupakan bahasa dasar untuk memvisualisasikan, mendokumentasikan serta mendesain sistem perangkat lunak (Dharwiyanti & Wahono, 2003). UML adalah metode yang ada pada proses mengembangkan sistem berorientasi objek yang juga adalah alat pendukung untuk

pergembangan sistem (Urva & Siregar, 2015).

Dalam perancangan perangkat lunak, terdapat beberapa alat bantu berbasis UML yang digunakan, di antaranya adalah:

1. *Use case Diagram.*

*Use case Diagram* merupakan permodelan dari kelakuan (*behaviour*) sistem informasi yang dibentuk (Urva & Siregar, 2015). Diagram *use case* berfungsi dalam pendeskripsi fitur-fitur yang dapat digunakan oleh pengguna ketika berinteraksi dengan sistem (Hendy, 2019). Dalam pembuatan *use case diagram* digunakan notasi atau simbol-simbol, di antaranya:

Tabel 2.3 Notasi Use Case Diagram

Gambar	Nama	Keterangan
	<i>Use case</i>	Menunjukkan fungsi yang terdapat dalam sistem, umumnya menggunakan kata kerja.
	Aktor	Orang yang menjalankan fungsi dari target sistem,
	Asosiasi Dua Arah	Garis tanpa panah yang menggambarkan apa atau siapa yang berinteraksi secara langsung.
	Asosiasi Satu Arah	Aktor dan sistem saling interaksi secara pasif.
	<i>Include</i>	Melakukan panggilan <i>use case</i> oleh <i>use case</i> lain.
	<i>Extend</i>	Meluaskan <i>use case</i> jika syarat terpenuhi.
	Generalisasi	Relasi beberapa aktor ke dalam satu actor.

## 2. Activity Diagram

Diagram *Activity* berfungsi untuk pendeskripsi alur yang berjalan pada sistem. Pada aplikasi seluler, diagram *activity* menunjukkan *behaviour* dan tahapan aktitivitas serta aksi (Chouhan *et al.*, 2012). Notasi yang dipergunakan untuk membuat *activity* diagram yaitu:

Tabel 2.4 Notasi Activity Diagram

Gambar	Nama	Keterangan
	<i>Start point</i>	Awal dari aktivitas, diletakkan pada pojok kiri atas.
	<i>End point</i>	Akhir dari aktivitas.
	<i>Activities</i>	Mendefinisikan suatu proses kerja.
	<i>Fork</i>	Percabangan dari kegiatan untuk mengalir dalam arah yang berbeda.
	<i>Join</i>	Penggabungan kegiatan.
	<i>Decision</i>	Pilihan untuk pengambilan keputusan.
	<i>Swimlane</i>	Pembagian pada diagram untuk menunjukkan siapa yang menjalankan aktivitas.

## 3. Entity Relationship Diagram (ERD)

*Entity Relationship Diagram* (ERD) adalah diagram konseptual yang dipakai dalam membuat rancangan suatu *database*. ERD menggambarkan bagaimana data yang disimpan dalam sebuah sistem beserta batasan yang dimilikinya. *Constraint*, *entity*, dan relasi merupakan

komponen utama sebuah ERD (Otair & Odat, 2015). Notasi yang perlu diperhatikan saat membuat ERD adalah sebagai berikut:

Tabel 2.5 Notasi Entity Relationship Diagram

Gambar	Nama	Keterangan
	<i>Entity</i>	Objek yang terdefinisi atau konsep dalam sistem.
	<i>One to one</i>	Digunakan untuk membagi <i>entity</i> menjadi dua untuk memberikan informasi secara ringkas
	<i>One to many</i>	Relasi antara dua <i>entity</i> dimana sebuah <i>instance</i> dari satu entitas dapat dihubungkan ke banyak <i>instance</i> dari entitas lainnya

## G. Unit Testing

*Unit testing* merupakan salah satu tipe pengujian perangkat lunak dimana setiap fungsi atau komponen dari perangkat lunak diuji. Tujuan dari *unit testing* adalah untuk memastikan fungsi pada perangkat lunak sudah berjalan sesuai dengan ekspektasi (Hamilton, 2022). Menurut Rosa dan Shalahuddin (2013: 277), “*Unit testing* berfokus pada pengujian unit terkecil (komponen perangkat lunak atau modul) dari desain perangkat lunak. Semua fungsi pada perangkat lunak diuji untuk memastikan bahwa *input* dan *output* unit sesuai dengan yang diinginkan”.

Pressman (2012) menyebutkan bahwa teknik yang dilakukan pada *unit testing* adalah “berfokus pada setiap unit perangkat lunak (misalnya komponen, kelas, atau objek konten dari aplikasi atau web) seperti yang diterapkan dalam kode program”. Kode program dikaji apakah terdapat kesalahan. Kesalahan pada kode program dapat diidentifikasi dengan menggunakan White-Box Testing. Sedangkan menurut Rosa dan Shalahuddin (2013), “White-Box Testing (pengujian kotak putih) merupakan pengujian perangkat lunak atau aplikasi dari segi desain dan kode program untuk mengetahui apakah perangkat lunak mampu menghasilkan fungsi-fungsi, masukan, dan keluaran yang sesuai dengan spesifikasi kebutuhan”.

Proses *unit testing* memastikan fungsi-fungsi pada aplikasi yang telah dikembangkan peneliti memenuhi persyaratan, dapat berjalan dengan baik, dan memiliki *input* serta *output* sesuai yang diinginkan. Berdasarkan definisi di atas dapat disimpulkan bahwa *unit testing* merupakan salah satu tipe pengujian fungsi atau unit pada perangkat lunak untuk memastikan apakah perangkat lunak mampu untuk menghasilkan *input* dan *output* sesuai dengan spesifikasi yang telah ditentukan.

#### **H. User Acceptance Test (UAT)**

Menurut Perry, William E, (2006) *User Acceptance Test* (UAT) yaitu pengujian untuk verifikasi apakah fungsi pada sistem telah berjalan dengan kebutuhan, pengujian dilakukan oleh user dimana user tersebut adalah staff/karyawan perusahaan yang langsung berinteraksi dengan sistem. Menurut Black, *acceptance testing* pada umumnya menunjukkan bahwa sistem telah memenuhi persyaratan-persyaratan tertentu. Pada pengembangan software dan hardware komersial, acceptance test biasanya disebut juga "alpha tests" (yang dilakukan oleh pengguna in-house) dan "beta tests" (yang dilakukan oleh pengguna yang sedang menggunakan atau akan menggunakan sistem tersebut).

Hady, Haryono, Rahayu, (2019) menyebutkan bahwa pelaksanaan UAT terdapat pada akhir proses pengujian saat sistem siap digunakan. Tujuan utama UAT adalah untuk memvalidasi apakah sistem diterima atau ditolak, memenuhi spesifikasi sistem, dan mengembangkan perangkat lunak yang mampu memenuhi kebutuhan user.

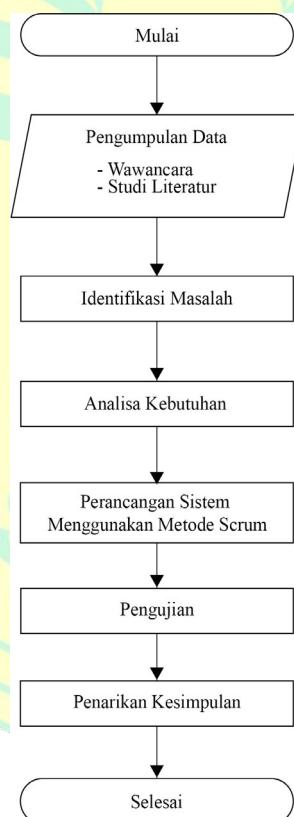
Proses UAT memastikan bahwa aplikasi pengkajian luka kronis khususnya modul pengolahan citra yang peneliti implementasikan, sudah layak diujikan secara masif, memenuhi harapan pengguna, dan bekerja seperti yang diharapkan. Dari definisi yang telah dipaparkan dapat disimpulkan bahwa UAT adalah pengujian pada akhir proses yang dilakukan oleh pengguna pada sebuah sistem untuk memastikan fungsi-fungsi yang terdapat pada sistem tersebut telah berjalan dengan baik dan sesuai dengan kebutuhan pengguna.

### BAB III

## METODOLOGI PENELITIAN

Penelitian yang dilakukan oleh penulis akan menghasilkan produk tertentu dan akan dilakukan pengujian keefektifannya (Purnama, 2013). Menurut Suhadi Ibnu (dalam Purnama, 2013), penelitian pengembangan bertujuan untuk menghasilkan suatu produk baik itu *software* ataupun *hardware* melalui prosedur yang umumnya dimulai dengan menganalisis kebutuhan, kemudian lanjut ke proses pengembangan, dan diakhiri dengan evaluasi.

Berdasarkan pengertian tersebut, penelitian yang akan dilaksanakan oleh penulis masuk ke dalam jenis Penelitian dan Pengembangan/*Research and Development*. Tahapan penelitian yang akan dilaksanakan penulis dalam perancangan aplikasi dapat dilihat pada Gambar 3.1.



Gambar 3.1 Tahapan penelitian

## A. Pengumpulan Data

Data diambil dari hasil wawancara dengan dosen Politeknik Kesehatan Jakarta I, Ns. Ratna Aryani, M.Kep., yang sekaligus merupakan klien dari penelitian ini. Selain itu, dilakukan studi literatur dengan membaca jurnal-jurnal yang terkait dengan topik penelitian. Untuk transkrip wawancara dapat dibaca pada Lampiran 1. dan Lampiran 2.

## B. Analisa Kebutuhan

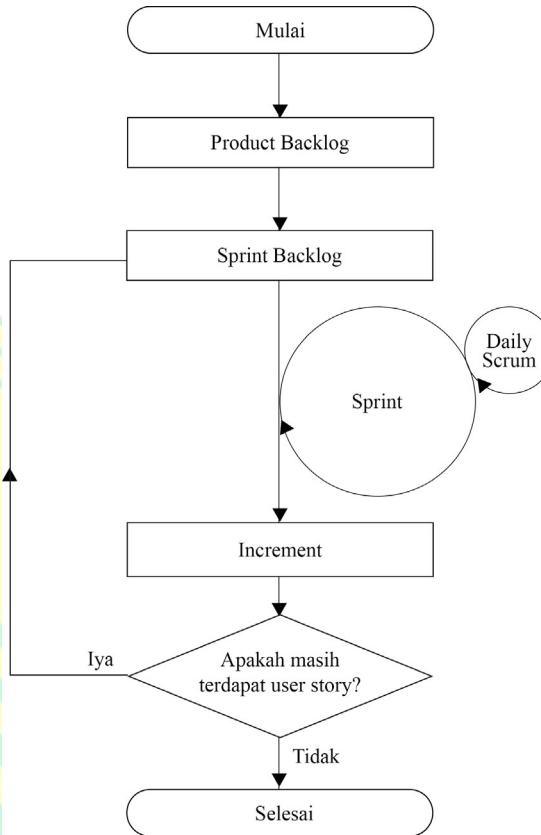
Berdasarkan uraian pada lampiran 2., prioritas fitur pada aplikasi pengkajian lukakronis berbasis *Android* terfokus pada kategori kajian luka yang memiliki data gambar seperti ukuran luka, tepi luka, dan epitalisasi luka. Skalar kategori diambil berdasarkan instrumen pengkajian luka BWAT.

Perangkat keras dan perangkat lunak yang penulis butuhkan adalah sebagai berikut:

1. Perangkat keras, terdiri dari:
  - a. Laptop dengan spesifikasi *Processor* Intel Core i5 generasi ke-7 & RAM 8 GB.
  - b. Perangkat *Android* Redmi Note 7.
2. Perangkat lunak, terdiri atas:
  - a. Windows 10 *Operating System*.
  - b. Figma sebagai alat untuk mendesain tampilan antarmuka.
  - c. *Android Studio* sebagai IDE untuk pembuatan aplikasi berbasis *Android*.
  - d. Java sebagai bahasa pemrograman penyusun aplikasi.
  - e. *Flask* sebagai *web framework* yang akan digunakan.
  - f. *MongoDB* sebagai basis data.

## C. Perancangan Sistem Menggunakan *Scrum*

Aplikasi yang akan dibuat pada penelitian ini dikembangkan dengan metode *Scrum*. Tahapan ini dilakukan setelah dilakukannya tahapan analisis kebutuhan. Untuk penjelasan metode secara detail akan dipaparkan pada sub-bab di bawah ini.



Gambar 3.2 Tahapan metode pengembangan menggunakan Scrum

### 1. Product Backlog

Tahap ini bertujuan untuk menerjemahkan seluruh fitur yang akan dibuat pada aplikasi. Fitur-fitur ini dibuat berdasarkan kebutuhan pengguna. Umumnya fitur pada *Product Backlog* didefinisikan sebagai *User Story*. Rincian *Product Backlog* dari aplikasi yang akan dibuat dapat dilihat pada tabel di bawah ini.

Tabel 3.1 Product Backlog

No.	User Story	Priority	Sprint No.
1	Konfigurasi Server	High	1
2	Konfigurasi Database	High	1
3	Membuat akun	Medium	1
4	Masuk ke akun yang sudah dibuat	Medium	1
5	Menambah pasien dan memilih pasien	High	2
6	Manajemen fotografi luka	High	3-4

No.	User Story	Priority	Sprint No.
7	Melakukan penambahan pengkajian luka	High	4
8	Menganotasi tepi luka dan menganotasi orientasi luka	High	4-6
9	Mengunggah luka kronis yang akan dikaji	Medium	5
10	Mengarsir warna luka	High	7
11	Galeri luka kronis semua pasien	High	8
12	Unduh dataset luka milik user tersebut	High	8
13	Melihat histori kajian pasien dan status luka pasien	Medium	8
14	Melihat detail profil akun perawat atau pengguna	Low	8
15	Keluar dari akun	Low	8
16	Mencatat log activity user	Low	9

*Product Backlog* yang dibuat memiliki lima kolom yang diantaranya adalah sebagai berikut:

- *User Story*  
Kolom *User Story* berisi fitur-fitur atau *User Stories* dari aplikasi yang akan dibuat.
- *Priortirity*  
Kolom ini menentukan tingkat prioritas dari sebuah *User Story*, dimana prioritas *High* merupakan fitur yang mempunyai peran penting pada penelitian ini.
- *Sprint No.*  
Pada kolom ini terdapat informasi tentang pada *Sprint* keberapa fitur tersebut akan dibuat.

## 2. *Sprint Backlog*

*Sprint Backlog* merupakan tahapan sebelum *sprint* dimulai, di dalamnya terdapat daftar pekerjaan yang keputusannya diambil dari

*Product Backlog*. Dengan *Sprint Backlog*, seluruh anggota tim dapat melihat perkembangan dari setiap pekerjaan. Pada penelitian ini terdapat tiga status perkembangan yang dipakai yaitu harus dikerjakan, sedang dikerjakan, selesai, dan *next sprint*.

### 3. Sprint

Setelah dilakukan perencanaan pada *Sprint Backlog*, maka pengerjaan *sprint* sudah dapat dimulai dan harus mengikuti jadwal pengerjaan yang telah disepakati bersama tim. Dalam penelitian ini, interval *sprint* yang digunakan adalah dua minggu.

### 4. Deploy

Aplikasi akan di *Deploy* setelah seluruh pekerjaan *sprint* yang telah direncanakan pada *sprint backlog* selesai. Setelah aplikasi telah di *deploy* kemudian dilakukan pengujian aplikasi dengan menggunakan *unit testing* dan *User Acceptance Test* (UAT).

## D. Pengujian

Pada tahap ini peneliti akan melakukan uji aplikasi pengkajian luka kronis khususnya modul pengolahan citra menggunakan dua jenis pengujian yaitu *unit testing* dan *User Acceptance Test* (UAT). Pengujian *unit testing* dilaksanakan oleh tim *internal developer* untuk memastikan fungsi-fungsi pada aplikasi yang telah dikembangkan dapat berjalan dengan baik. Sedangkan UAT dilaksanakan oleh pengguna untuk mengetahui apakah aplikasi sudah sesuai kebutuhan dan layak untuk diuji secara masif.

### 1. Unit Testing

Skenario pada *unit testing* dibuat berdasarkan *product backlog*. Adapun skenario dari *unit testing* yang akan dilaksanakan terdapat pada tabel 3.2.

Tabel 3.2 Skenario *Unit Testing*

Uji Fitur	Skenario Pengujian
Halaman Awal	Saat aplikasi dibuka akan muncul permission request untuk menggunakan kamera dan akses ke storage
	Saat permission disetujui maka akan masuk ke tampilan awal, jika tidak, maka tidak dapat masuk ke aplikasi
	Jika tombol "LOG IN" ditekan, guest akan masuk ke halaman Login

<b>Uji Fitur</b>	<b>Skenario Pengujian</b>
	Jika tombol "Buat Akun" ditekan maka akan masuk ke halaman Buat Akun
Registrasi User/Perawat	Ketika mengisi form registrasi dengan lengkap kemudian submit, maka akan masuk ke halaman beranda
	Ketika mengisi form registrasi tidak lengkap kemudian submit, maka akan menampilkan pesan kesalahan
Login	Ketika mengisi form login dengan data yang sesuai kemudian klik submit, maka akan masuk ke halaman beranda
	Ketika mengisi form login dengan data yang tidak sesuai kemudian klik submit, maka akan menampilkan pesan kesalahan
Beranda	Saat ikon profil pada kanan atas di-klik maka akan muncul halaman profil perawat
	Saat tombol daftar pasien di-klik maka akan muncul halaman daftar pasien
	Saat tombol galeri foto luka di-klik maka akan muncul halaman galeri foto luka
	Saat tombol arsir warna luka di-klik maka akan muncul halaman arsir warna luka
Halaman daftar pasien	Saat ikon panah pada kiri atas diklik akan kembali ke halaman beranda
	Saat ikon lup pada kanan atas diklik maka akan muncul isian untuk mencari pasien
	Saat mencari pasien dengan input nama yang terdapat pada database, maka akan muncul nama pasien tersebut
	Saat mencari pasien dengan input nama yang tidak terdapat pada database, maka akan muncul pesan bahwa tidak ditemukan
	Saat mencari pasien dengan input nomor rekam medis pasien yang terdapat pada database, maka akan muncul nama pasien sesuai dengan nomor registrasi yang telah di-input
	Saat mencari pasien dengan input nomor rekam medis pasien yang tidak terdapat pada database, maka akan muncul pesan bahwa tidak ditemukan
	Saat menekan tombol Laki-laki maka akan muncul seluruh pasien laki-laki
	Saat menekan tombol Perempuan maka akan muncul seluruh pasien perempuan
	Saat menekan tombol tambah pasien baru pada kanan bawah, maka akan muncul halaman penambahan pasien baru
	Saat klik nama pasien, maka akan muncul halaman detail pasien
Tambah pasien baru	Ketika mengisi form pasien dengan lengkap kemudian klik submit, maka akan masuk ke halaman detail pasien yang baru saja diinput
	Ketika mengisi form pasien tidak lengkap kemudian submit, maka akan menampilkan pesan kesalahan
Detail pasien	Saat masuk pertama kali ke halaman detail pasien, maka

<b>Uji Fitur</b>	<b>Skenario Pengujian</b>
	informasi umum pasien terlihat
	Saat klik ikon pensil maka akan masuk ke halaman edit data pasien
	Saat tab Histori Kajian diklik maka akan muncul histori kajian pasien
	Saat tab Galeri Luka diklik maka akan muncul seluruh foto luka milik pasien
	Saat tombol Raw pada tab Galeri Luka diklik, maka akan muncul semua foto luka tanpa anotasi dengan format JPG.
	Saat tombol Anotasi Tepi pada tab Galeri Luka diklik, maka akan muncul semua foto luka beranotasi tepi dengan format JPG.
	Saat tombol Anotasi Diameter pada tab Galeri Luka diklik, maka akan muncul semua foto luka beranotasi diameter dengan format JPG.
	Saat klik tombol "Tambah Hasil Kajian" pada tab Histori Kajian, maka akan diarahkan ke halaman tambah hasil kajian.
	Saat klik tombol tanggal yang terdapat pada tab Histori Kajian, maka akan diarahkan menuju halaman detail kajian luka sesuai dengan tanggal yang dipilih.
	Saat klik ikon pensil pada sebelah kanan teks "Informasi Umum Pasien" maka akan diarahkan menuju halaman Edit Data Pasien
Edit data pasien	Saat pertama kali masuk ke halaman Edit Data Pasien maka akan ditampilkan kembali informasi umum yang telah ada serta ikon pensil di sebelah kanannya
	Saat klik ikon pensil, maka akan ditampilkan pesan perintah, kolom isian, dan tombol submit
	Ketika tidak mengisi kolom kemudian klik submit, maka akan ditampilkan pesan error
	Ketika mengisi kolom dengan lengkap kemudian klik submit, maka akan masuk ke halaman Edit Data Pasien dengan data yang sudah diperbarui
Tambah hasil kajian	Saat pertama kali masuk ke halaman tambah hasil kajian baru hanya terdapat nama pasien, nomor rekam medis, jenis kelamin, usia, dan tombol kamera.
	Ketika pengguna belum mengambil foto luka maka tidak akan muncul form kajian luka.
	Ketika pengguna mengklik tombol kamera, maka akan masuk ke halaman pengambilan foto luka.
	Ketika pengguna selesai mengambil foto luka, maka akan diarahkan ke halaman anotasi tepi luka.
	Ketika pengguna selesai menganotasi tepi luka dan klik tombol "SAVE" maka akan diarahkan untuk menganotasi diameter koordinat X.
	Ketika pengguna selesai menganotasi diameter luka koordinat X dan klik tombol "SAVE" maka diarahkan untuk menganotasi diameter koordinat Y.
	Ketika pengguna selesai menganotasi diameter luka koordinat X dan klik tombol "SAVE" maka diarahkan

<b>Uji Fitur</b>	<b>Skenario Pengujian</b>
	untuk menganotasi diameter koordinat Y
	Ketika pengguna selesai menganotasi diameter luka koordinat Y dan klik tombol "SAVE" maka diarahkan untuk mengisi form kajian luka.
	Ketika mengisi form kajian dengan lengkap kemudian klik submit, maka akan masuk ke halaman detail pasien yang baru saja diinput.
	Ketika mengisi form kajian secara tidak lengkap kemudian klik submit, maka akan masuk ke halaman detail pasien yang baru saja diinput.
Halaman detail kajian	<p>Saat masuk pertama kali ke halaman detail kajian, maka akan ditampilkan informasi tanggal kajian, nama pasien, perawat yang menangani, NIP perawat, foto luka, anotasi tepi luka, anotasi diameter luka, skoring kajian luka, dan ikon pensil untuk mengedit data</p> <p>Saat klik ikon pensil di atas foto luka atau foto anotasi, maka akan masuk ke halaman untuk melakukan foto ulang atau anotasi ulang</p> <p>Saat klik ikon pensil pada sisi kanan tulisan "Skoring Kajian Luka" maka akan masuk ke halaman Edit Skoring Kajian</p> <p>Saat klik ikon pensil pada sisi kanan tulisan "Skoring Kajian Luka" maka akan masuk ke halaman Edit Skoring Kajian</p>
Edit skoring kajian	<p>Saat pertama kali masuk ke halaman Edit Skoring Kajian maka akan ditampilkan kembali informasi skoring yang telah ada serta ikon pensil di sebelah kanannya</p> <p>Saat klik ikon pensil, maka akan ditampilkan pesan perintah, kolom isian, dan tombol submit</p> <p>Ketika tidak mengisi kolom kemudian klik submit, maka akan ditampilkan pesan error</p> <p>Ketika mengisi kolom dengan lengkap kemudian klik submit, maka akan masuk ke halaman Edit Skoring Kajian dengan data yang sudah diperbarui</p>
Halaman detail foto	Saat pertama kali masuk ke halaman detail foto, maka akan ditampilkan foto tersebut (kecuali yang berformat SVG), informasi filename, informasi nomor rekam medis pasien, informasi NIP perawat yang mengambil foto, tanggal unggah, dan kategori.
Arsir warna luka	<p>Saat klik ikon galeri biru yang terdapat saat pertama kali membuka fitur arsir warna luka, maka pengguna diarahkan untuk mengunggah foto dari galeri.</p> <p>Ketika pengguna klik tombol panah pada kiri atas, maka akan diarahkan kembali ke halaman sebelumnya.</p> <p>Pada saat pengguna menggambar, warna yang pertama kali dipakai adalah warna hitam.</p>
	Ketika tombol "Red" diklik maka warna stroke akan berubah menjadi warna merah.
	Ketika tombol "Yellow" diklik maka warna stroke akan berubah menjadi warna kuning.
	Ketika tombol "Black" diklik maka warna stroke akan

<b>Uji Fitur</b>	<b>Skenario Pengujian</b>
	berubah menjadi warna hitam.
	Ketika tombol "Undo" diklik maka stroke yang baru saja digambar akan hilang
	Ketika ikon bulat biru digeser ke kanan, maka stroke akan semakin lebar.
	Ketika selesai mengarsir warna luka dan klik tombol save, maka foto akan diunggah ke server lalu kembali ke halaman beranda.
Profil perawat	Saat pertama kali masuk ke halaman Profil Perawat maka akan ditampilkan nama perawat, NIP perawat, email perawat, tombol log out, dan ikon pensil
	Ketika tombol "LOG OUT" diklik, maka pengguna akan kembali ke halaman awal dan tidak bisa mengakses fitur pada aplikasi.
	Ketika ikon pensil diklik maka akan diarahkan ke halaman edit data perawat.
Edit data perawat	Saat pertama kali masuk ke halaman Edit Profil Perawat maka akan ditampilkan kalimat perintah, kolom teks untuk mengedit, dan tombol submit.
	Ketika mengisi kolom dengan lengkap kemudian klik submit, maka akan masuk ke halaman Profil Pasien
	Ketika tidak mengisi kolom kemudian klik submit, maka akan ditampilkan pesan error
Log activity user	Saat mengunjungi link <a href="http://jft.web.id/woundapi/get_logs">http://jft.web.id/woundapi/get_logs</a> maka akan ditampilkan histori logging seluruh pengguna.
Unduh dataset <i>image</i> luka	Saat mengunjungi link <a href="http://jft.web.id/woundapi/">http://jft.web.id/woundapi/</a> maka akan ditampilkan halaman untuk mengunduh dataset foto luka kronis.
	Saat tombol "Unduh Semua Foto" diklik, maka akan mengunduh semua foto luka kronis tanpa terkecuali.
	Saat tombol "Unduh" pada card Raw Image diklik, maka akan mengunduh semua foto luka kronis tanpa anotasi.
	Saat tombol "Unduh" pada card Anotasi Tepi diklik, maka akan mengunduh semua foto anotasi tepi luka kronis.
	Saat tombol "Unduh" pada card Anotasi Diameter diklik, maka akan mengunduh semua foto anotasi diameter luka kronis.
	Setelah mengisikan nomor rekam medis pada form yang tersedia lalu menekan tombol "Unduh", maka akan mengunduh semua foto pasien berdasarkan nomor rekam medis yang telah di-input.

## 2. User Acceptance Test

Skenario pada *User Acceptance Test* dibuat berdasarkan fitur-fitur yang dapat diakses oleh pengguna pada *product backlog*. Adapun skenario dari *UAT* yang akan dilaksanakan terdapat pada tabel 3.3.

Tabel 3.3 Skenario Pengujian *User Acceptance Test*

Uji Fitur	Skenario Pengujian	Jenis Pengujian
Registrasi	Menekan tombol buat akun lalu mengisi form registrasi kemudian submit	UAT
Login	Menekan tombol login lalu mengisi form login dengan NIP dan password kemudian submit	UAT
Daftar pasien	Klik tombol daftar pasien pada beranda kemudian menampilkan daftar pasien	UAT
Menambahkan pasien	Klik tombol tambah pada tampilan daftar pasien kemudian mengisi form tambah pasien kemudian submit	UAT
Melihat detail pasien	Klik tombol daftar pasien pada beranda kemudian klik pasien yang ingin dilihat detailnya	UAT
Galeri luka	Klik tombol galeri luka kronis pada beranda	UAT
Arsir warna luka	Klik tombol arsir warna luka kronis pada beranda kemudian unggah gambar melalui galeri lalu mengarsir warna luka dan submit	UAT
Menambahkan kajian luka baru	Buka halaman detail pasien lalu klik tab histori kajian luka kemudian klik tombol tambah hasil kajian dilanjutkan dengan isi form kajian dan submit	UAT
Anotasi tepi luka	Buka halaman detail pasien lalu klik tab histori kajian luka kemudian klik icon tambah dilanjutkan dengan memfoto luka kronis lalu melakukan anotasi tepi luka dan submit	UAT
Anotasi diameter luka	Setelah melakukan anotasi tepi luka, masuk ke halaman anotasi diameter luka kemudian melakukan anotasi diameter luka dan submit	UAT
Menambahkan kajian luka baru	Setelah melakukan anotasi tepi luka, isi form kajian dan submit.	UAT
Melihat histori kajian luka	Buka halaman detail pasien lalu klik tab histori kajian luka	UAT
Melihat detail dari histori kajian luka	Buka halaman detail pasien lalu klik tab histori kajian luka kemudian klik card sesuai dengan tanggal kajian luka yang ingin dilihat	UAT
Galeri luka satu pasien	Buka halaman detail pasien lalu klik tab galeri luka pasien	UAT
Profil perawat	Dari halaman beranda, klik ikon profil pada kanan atas	UAT
Logout	Pada halaman profil perawat, klik tombol logout	UAT

## BAB IV

### HASIL DAN PEMBAHASAN

#### A. Pembahasan

*Web Service* beserta Aplikasi pengkajian luka kronis berbasis Android khususnya modul pengolahan citra, dirancang dengan menggunakan metode *Scrum*. Pada metode *Scrum*, proses pengembangan sistem dilakukan secara bertahap yang disebut dengan *Sprint*. Pada penelitian ini terdapat sembilan *Sprint* dimana satu putaran *Sprint* memiliki durasi selama dua minggu. Pada tiap awal pekan, dilakukan perencanaan *Sprint Backlog* berdasarkan *Product Backlog* telah disepakati. Adapun laporan setiap *Sprint* yang dilakukan pada proses pengembangan sistem adalah sebagai berikut:

##### a. *Sprint-1*

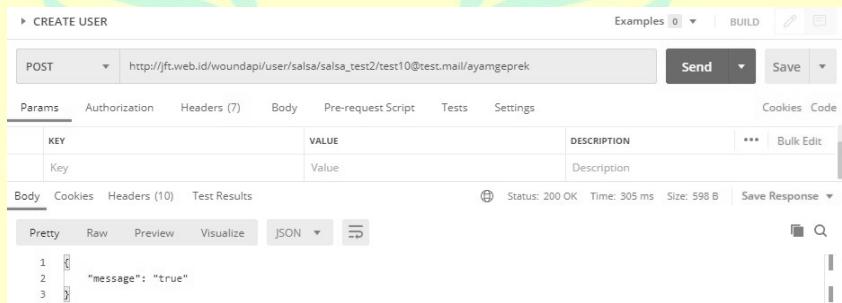
Tabel 4.1 Sprint-1 Backlog

No.	User Story	Task	Status
1.	Konfigurasi Server	<ol style="list-style-type: none"><li>1. Mengunggah file <i>code</i> ke <i>server</i></li><li>2. Menambahkan <i>code</i> pada <i>httpd.conf</i></li></ol>	Selesai
2.	Konfigurasi Database	<ol style="list-style-type: none"><li>1. Merubah variabel database yang digunakan pada file “<i>settings.cfg</i>”</li><li>2. Mengunggah file <i>code</i> ke <i>server</i></li></ol>	Selesai
3.	Dapat Membuat Akun	<ol style="list-style-type: none"><li>1. Membuat <i>database user</i></li><li>2. Membuat REST API untuk pembuatan akun</li><li>3. Membuat <i>mock up</i> tampilan awal dan tampilan pembuatan akun</li><li>4. Implementasi <i>layouting mock up</i> pada Android menggunakan XML</li><li>5. Implementasi <i>logic</i> di <i>FrontEnd</i> untuk pemanggilan REST pada <i>BackEnd</i></li></ol>	Selesai

No.	User Story	Task	Status
4.	Dapat masuk ke akun yang sudah saya buat	1. Membuat REST API untuk validasi akun 2. Membuat mock up tampilan Sign In 3. Implementasi layouting Sign In pada Android 4. Implementasi logic di Front End untuk pemanggilan REST pada BackEnd	Selesai

### 1) Konfigurasi Server

Konfigurasi *server* dilakukan dengan mengunggah *file code* ke *server* dan menambahkan kode pada httpd.conf. Pengecekan bahwa REST API sudah dapat digunakan dilakukan menggunakan POSTMAN dapat dilihat pada Gambar 4.1.



Gambar 4.1 Pemanggilan REST API buat akun

### 2) Konfigurasi Database

Konfigurasi *database* dikembangkan terlebih dahulu dilakukan untuk menerjemahkan koneksi MongoDB dan Flask. Dimana *MongoDB* dan *Flask* dikoneksikan dengan menggunakan bantuan *library* pymongo.

```

SECRET_KEY='dev'
DATABASE='wounddb'
MONGO_CON='mongodb://localhost:27017/'
UPLOAD_DIR = 'media'
ALLOWED_EXTENSIONS = {'pdf', 'png', 'jpg', 'jpeg'}
PASS_LENGTH=6
  
```

Kode di atas adalah kode pada *file* 'settings.cfg' untuk mengatur variabel *database* yang akan digunakan.

```

import click
import pymongo
from flask import current_app, g
from flask.cli import with_appcontext

def get_db():
    mongocon = current_app.config['MONGO_CON']
    dbclient = pymongo.MongoClient(mongocon)
    g.db = dbclient[current_app.config['DATABASE']]
    return g.db

```

Variabel tersebut akan dipanggil oleh pymongo untuk mengoneksikan *database* pada file ‘db.py’ pada kode di atas.

### 3) Membuat Database User

*Database User* yang nantinya akan dipakai untuk fitur *Sign In* dan buat akun. Tabel *User* memuat beberapa atribut yaitu ID, nama, *username*, *e-mail* dan *password*. Data tersebut akan disimpan menggunakan *database MongoDB*, umumnya data yang disimpan menggunakan format JSON. Tabel atau *collection* *User* akan dibuat bersamaan dengan pemanggilan REST API pembuatan akun.

user		
ID	integer(10)	
Nama	varchar(255)	N
Username	varchar(255)	N
Email	varchar(255)	N
Password	varchar(255)	N

Gambar 4.2 Tabel User

### 4) Membuat REST API untuk pembuatan akun

REST API dibuat menggunakan *framework Flask*. *Flask* adalah web *service* berbasis python. URL *Routing* untuk melakukan buat akun adalah `jft.web.id/woundapi/user/<name>/<username>/<email>/<pass>` sehingga data pengguna akan dikirim melalui *path* pada URL.

Kode di bawah merupakan kode untuk REST API buat akun, metode yang digunakan pada REST API ini adalah POST. Pada saat pemanggilan REST API, terlebih dahulu dilakukan pengecekan apakah

sudah ada data yang sama atau belum tersedia, jika sudah ada maka akan dikembalikan status ke HTTP 404 untuk menandakan *error*, jika data belum tersedia maka akan dikembalikan status ke HTTP 200 yang menandakan bahwa buat akun berhasil.

```
@bp.route('/user/<name>/<username>/<email>/<pa  
ssw>', methods=["POST"])  
def create_user(name, username, email,  
passw): try:  
    filter = {}  
    filter["name"]  
    = name  
    filter["username"] =  
    username  
    filter["email"] = email  
    filter["password"] =  
    passw cek =  
    get_user(filter)  
  
    if cek == None:  
        a = list(db.get_users())  
  
        data = {"_id": 8200000 +  
                len(a) + 1,  
                "name":name,  
                "username":username,  
                "email": email,  
                "password": passw}  
  
        row = insert_user(data)  
        print("berhasil input user  
baru") return  
        Response(response =  
json.dumps({"message" : "true"}),  
mimetype="application/json",  
status=200)  
    else:  
        return  
Response(response =  
json.dumps({"message" : "false"}),  
mimetype="application/json",  
status=404)  
  
except Exception as ex:  
    print("Input user baru")
```

```
gagal")
return Response(response =
json.dumps({"message"
: "false"}), mimetype="application/json",
status=500)
```

5) *Mockup* tampilan awal dan buat akun



Gambar 4.3 Tampilan Awal

Gambar di atas merupakan halaman awal sebagai pembuka ketika pengguna mengakses aplikasi. Pada tampilan ini terdapat nama aplikasi serta deskripsi tentang apa yang dapat dilakukan pengguna dengan aplikasi tersebut, tombol *Sign In* untuk masuk ke akun yang telah dibuat dan tombol *Buat Akun* untuk mendaftarkan pengguna baru.

### Buat Akun

**Nama**  
Ns. Jane Doe

**Username**  
Username

**Email**  
Example@email.com

**Password**  
\*\*\*\*\*

**Buat Akun**

Gambar 4.4 Tampilan Buat Akun

Ketika pengguna menekan tombol Buat Akun pada Gambar 4.3, maka akan diarahkan ke Gambar 4.4. Pada tampilan buat akun, terlebih dahulu pengguna harus mengisikan *field* nama, *username*, *e-mail*, dan *password*. Pembuatan akun ini wajib dilakukan agar pengguna dapat mengakses fitur utama aplikasi.

- 6) Implementasi *layouting mockup* tampilan awal dan tampilan buat akun  
Implementasi *mockup* dibuat dengan menggunakan *Android Studio*.

*Encoder* yang digunakan adalah *Extensible Markup Language* (XML).

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/FirstTime"
    android:visibility="invisible"
    tools:context="com.example.chronicwound.LoginActivity"

    xmlns:android="http://schemas.android.com/apk/res/android">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_marginTop="-20dp"
        android:layout_marginLeft="40dp"
    "
    android:src="@drawable/ic_corne
    relips" />

    <LinearLayout
        android:layout_width="match_
        parent"
        android:layout_height="match
        _parent"
        android:layout_marginHorizon
        tal="30dp"
        android:gravity="bottom"
        android:orientation="vertica
        l">

        <ImageView
            android:id="@+id/imageView
            3"
            android:layout_width="132d
            p"
            android:layout_height="173
            dp"
            android:layout_marginBotto
            m="8dp"
            android:scaleType="fitStar
            t"
            android:src="@drawable/ic_
            tangan" />

        <TextView
            android:layout_width="match_paren
            t"
            android:layout_height="wrap_ conte
            nt"
            android:layout_marginBottom="8dp"
            android:lineHeight="46sp"
            android:text="Chronic Wound
            Assesment Tool"

            android:textAppearance="@style/TextAppearance.App
            Compat.Dis play1"
            android:textColor
            ="#000000"
            android:textSize=
            "34sp" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_paren
            "
            android:layout_height="wrap_ conten
            t"
            android:layout_marginBottom="34dp"
            android:lineHeight="26sp"
```

```
        android:text="Anda dapat membantu  
        kami untuk  
        mengumpulkan dataset citra warna luka dan tepi  
        luka kronik serta mengunduhnya lewat aplikasi  
        ini"  
        android:textSize="18sp" />  
  
<Button  
        android:id="@+id/log_in"  
        android:layout_width="match_parent"  
        android:layout_height="70dp"  
        android:layout_marginBottom="8dp"  
        android:background="@drawable/prima  
        ry_button" android:text="Log In" />  
  
<Button  
        android:id="@+id/buat_akun"  
        android:layout_width="match_  
        parent"  
        android:layout_height="70dp"  
        android:layout_marginBottom=  
        "20dp"  
        android:backgroundTint="@col  
        or/Accent"  
        android:textColor="@color/Ac  
        cent"  
        android:background="@drawable/secondary_b  
        utton" android:text="Buat Akun" />  
  
</LinearLayout>  
  
</RelativeLayout>
```

Adapun kode di atas adalah implementasi dari *mockup* tampilan awal dari aplikasi dimana susunan atas Relative Layout sebagai *parent view* dan Linear Layout sebagai *child view*. Untuk kode yang terlampir di bawah merupakan kode implementasi dari *mockup* tampilan pembuatan akun, terdapat beberapa input teks yang nantinya akan dikirimkan ke server menggunakan REST API.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/regisForm"
        android:visibility="invisible"
        tools:context="com.example.chronicwound.LoginActivity"
        xmlns:app="http://schemas.android.com/apk/res-auto">

    <LinearLayout
        android:layout_width="match_parent"
        "
        android:layout_height="match_parent"
        android:layout_marginHorizontal="30dp"
        android:layout_marginTop="20dp"
        android:orientation="vertical">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="40dp" android:text="Buat Akun"
            android:textAppearance="@style/TextAppearance.AppCompat
            .Display1"
            android:textColor="#000
            000"
            android:textSize="34sp"
            android:layout_weight="0"/>

        <!-- Nama -->
        <com.google.android.material.textfield.TextInputLayout
            style="@style/Widget.MaterialComponents.TextInputLayout
            .OutlinedBox"
            android:id="@+id/textInputLayoutNama"
```

```
        app:errorEnabled="true"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Nama"
        android:layout_weight="0"
        >

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/editNama"
        android:layout_width="match_parent"
        android:layout_height="70dp"
        />

</com.google.android.material.textfield.TextInputLayout>
<!-- Username-->
<com.google.android.material.textfield.TextInputLayout
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
    android:id="@+id/textInputLayoutUserName"
    app:errorEnabled="true"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" android:hint="Username"
    app:helperText="Hanya abjad A-Z dan angka"
    app:counterEnabled="true"
    app:counterMaxLength="10"
    android:layout_weight="0"
    android:layout_marginBottom="8dp"
    >

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/editTextUserName"
    android:layout_width="match_parent"
    android:layout_height="70dp"
    android:maxLength="10"
    />

</com.google.android.material.textfield.TextInputLayout>
<!-- Email-->
<com.google.android.material.textfield.TextInputLayout
```

```
        style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
            app:errorEnabled="true"
            android:layout_width="match_parent"
            "
            android:id="@+id/textInputLayoutEmail"
            android:layout_height="wrap_content" android:hint="Email"
            app:helperText="Masukkan E-mail
            Anda disini"
            android:layout_weight="0"
            android:layout_marginBottom="8dp">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/editTextEmail"
            android:layout_width="match_parent"
            " android:layout_height="70dp"
            android:inputType="textEmailAddress"
            />

    </com.google.android.material.textfield.TextInputLayout>

    <!-- Password-->
    <com.google.android.material.textfield.TextInputLayout
        style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
            app:errorEnabled="true"
            app:passwordToggleEnabled =
            "true"
            android:id="@+id/textInputLayoutPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" android:hint="Password"
            app:helperText="Minimal 6
            karakter"
            android:layout_weight="0"
            android:layout_marginBottom="8d
            p">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/editTextPassword"
            android:layout_width="match_parent"
            " android:layout_height="70dp"
            android:inputType="textPassword"
            />
```

```
</com.google.android.material.textfield.TextInput  
Layout>  
  
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="0dp"  
    android:layout_marginTop="20dp"  
    android:layout_weight="1"  
    android:orientation="vertical">  
  
    <Button  
        android:id="@+id/buttonRegister"  
        android:layout_width="match_parent"  
        android:layout_height="70dp"  
  
        android:layout_gravity="center_horizontal|center"  
        android:background="@drawable/primary_button"  
        android:layout_marginBottom="20dp"  
        android:text="Buat Akun" />  
    </LinearLayout>  
</LinearLayout>  
</RelativeLayout>
```

- 7) Implementasi *logic* di *FrontEnd* untuk pemanggilan REST pada *BackEnd*

Pada tahap ini dibuat *logic* untuk memanggil REST API pembuatan akun yang telah dibuat. Pemanggilan REST API dibantu dengan kelas *Retrofit* terlampir di bawah yang mengubah respon HTTP menjadi antar-muka Java.

```
package com.example.chronicwound.remote;

import okhttp3.OkHttpClient;
import
okhttp3.logging.HttpLoggingInterceptor;
import retrofit2.Retrofit;
import

retrofit2.converter.gson.GsonConverterFactory;

public class RetrofitClient {

    public static Retrofit getRetrofit(){

        HttpLoggingInterceptor logging
= new HttpLoggingInterceptor();

logging.setLevel(HttpLoggingInterceptor.Level.B

ODY); OkHttpClient okHttpClient = new
OkHttpClient.Builder().addInterceptor(logging).bu

ld(); Retrofit retrofit = new
Retrofit.Builder()

.addConverterFactory(GsonConverterFactory.create())
.baseUrl("http://jft.web.id/woundapi/")
.client(okHttpClient)
.build();

    return retrofit;
}

    public static UserService
getService(){ UserService
userService =
getRetrofit().create(UserService.cl
ass); return userService;
}
}
```

*Controller class ini diperuntukkan untuk membuat client  
Retrofit serta memanggil class userService pada kode di bawah.*

```
package  
com.example.chronicwound.remote;  
import retrofit2.Call;  
import  
retrofit2.http.GET;  
import  
retrofit2.http.POST;  
import  
retrofit2.http.Path;  
  
public interface UserService {  
  
    @GET("user/find/{username}/{password}")  
    Call<LoginResponse> login(@Path("username")  
String username, @Path("password") String  
password);  
  
    @POST("user/{name}/{username}/{email}/{passw}")  
    Call<RegisterResponse> signup(@Path("name")  
String name, @Path("username") String username,  
@Path("email")  
String email, @Path("passw") String password);  
}
```

Kemudian konstruktor dari atribut yang diperlukan pada REST pembuatan akun beserta *setter* dan *getter* untuk masing-masing atribut tersebut diterjemahkan pada *Class RegisterRequest.java*

```
package  
com.example.chronicwound.remote  
;  
public class RegisterRequest  
{  
    private String  
    name; private  
    String  
    username;  
    private String  
    email; private  
    String  
    password;  
  
    public String  
    getName() {  
        return name;  
    }  
  
    public void  
    setName(String name)  
    { this.name = name;  
    }  
  
    public String  
    getUsername() {  
        return username;  
    }  
  
    public void setUsername(String  
    username) { this.username =  
    username;  
    }  
  
    public String  
    getEmail() {  
        return email;  
    }  
  
    public void setEmail(String  
    email) { this.email =  
    email;  
    }  
  
    public String  
    getPassword() {  
        return password;  
    }  
  
    public void setPassword(String  
    password) { this.password =  
    password;  
    }  
}
```

RegisterResponse.java pada kode di bawah berfungsi untuk mendefinisikan respon setelah REST dipanggil.

```
package com.example.chronicwound.remote;

public class RegisterResponse {

    private String message;

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

Pada *class* RegisterActivity.java, *class controller* dipanggil dalam fungsi doRegister(), fungsi tersebut memerlukan parameter masukan berupa nama, *username*, *e-mail* dan *password*.

```
package com.example.chronicwound;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RelativeLayout;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import com.example.chronicwound.remote.RegisterResponse;
import com.example.chronicwound.remote.ResObj;
import com.example.chronicwound.remote.RetrofitClient
```

```
com.example.chronicwound.remote.UserService;
import
com.google.android.material.snackbar.Snackba
r; import
com.google.android.material.textfield.TextIn
putLayout;

import
retrofit2.Call;
import
retrofit2.Callba
ck; import
retrofit2.Respon
se;

public class RegisterActivity extends AppCompatActivity
{

    //Declaration
    EditTexts
    EditText
    editTextUserName
    ; EditText
    editTextEmail;
    EditText
    editTextPassword
    ; EditText
    editTextNama;

    //Declaration TextInputLayout
    TextInputLayout
    textInputLayoutUserName;
    TextInputLayout
    textInputLayoutEmail;
    TextInputLayout
    textInputLayoutPassword;
    TextInputLayout
    textInputLayoutNama;

    //Declaration Button
    Button
    buttonRegister
    ; UserService

    userService;

    //Declaration Layout
    RelativeLayout registForm;

    @Override
    protected void
    onCreate(@Nullable Bundle
    savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
setContentView(R.layout.tampilan_register);

registForm = findViewById(R.id.registForm);
// Check if UserResponse is Already Logged In

if(SaveSharedPreference.getLoggedStatus(getApplicationContext())) {
    Intent intent = new
Intent(getApplicationContext(), MainActivity.class);
startActivity(intent);
} else {
registForm.setVisibility(View.VISIBLE);
}

initViews();

buttonRegister.setOnClickListener(new
View.OnClickListener() {
    @RequiresApi(api =
Build.VERSION_CODES.GINGERBREAD)
    @Override
    public void onClick(View view) { if
        (validate()) {
            String UserName =
editTextUserName.getText().toString();
            String Email =
editTextEmail.getText().toString();
            String Password =
editTextPassword.getText().toString();
            String Nama =
editTextNama.getText().toString();

            doRegister(Nama, UserName, Email,
Password);
        }
    });
}

//this method is used to connect XML views to its
Objects
private void initViews() {
    editTextEmail = (EditText)
findViewById(R.id.editTextEmail); editTextPassword
    = (EditText)
findViewById(R.id.editTextPassword);
    editTextUserName = (EditText)
findViewById(R.id.editTextUserName);
    editTextNama = (EditText)
findViewById(R.id.editNama); TextInputLayoutEmail =
(TextInputLayout)
```

```
findViewById(R.id.textInputLayoutEmail);
    TextInputLayoutPassword = (TextInputLayout)
findViewById(R.id.textInputLayoutPassword);
    TextInputLayoutUserName = (TextInputLayout)
findViewById(R.id.textInputLayoutUserName);
    TextInputLayoutNama = (TextInputLayout)
findViewById(R.id.textInputLayoutNama);
    buttonRegister = (Button)
findViewById(R.id.buttonRegister);

}
//This method is used to validate input given by user
@RequiresApi(api = Build.VERSION_CODES.GINGERBREAD)
public boolean validate() {
    boolean valid = false;

    //Get values from EditText fields
    String UserName =
editTextUserName.getText().toString();
    String Email = editTextEmail.getText().toString();
    String Password =
editTextPassword.getText().toString();

    //Handling validation for UserName field

    if (UserName.isEmpty()) {
        valid = false;
        TextInputLayoutUserName.setError("Username
tidak boleh kosong");
    } else {
        if
(UserName.matches("^(?=.{0,10}$)(?![_.]) (?!.*[_.]{2})[a-zA-
Z0-9._]+(?![_.]$)")) {
            valid = true;
            TextInputLayoutUserName.setError(null);
        } else {
            valid = false;
            TextInputLayoutUserName.setError("Tidak
boleh selain abjad dan angka");
        }
    }

    //Handling validation for Email field
    if
(!android.util.Patterns.EMAIL_ADDRESS.matcher(Email).matche
s()) {
        valid = false;
        TextInputLayoutEmail.setError("Email tidak
boleh kosong");
    } else {
        valid = true;
        TextInputLayoutEmail.setError(null);
    }
}
```

```
//Handling validation for Password field
if (Password.isEmpty()) {
    valid = false;
    TextInputLayoutPassword.setErrorEnabled(true);
    TextInputLayoutPassword.setError("Password
tidak boleh kosong");

} else {
    if (Password.length() > 5) {
        valid = true;
        TextInputLayoutPassword.setError(null);
    } else {
        valid = false;
    }
}

TextInputLayoutPassword.setErrorEnabled(true);
TextInputLayoutPassword.setError("Minimal 6
karakter");
}

TextInputLayoutPassword.setError(null);
} else {
    valid = false;
}

TextInputLayoutPassword.setErrorEnabled(true);
TextInputLayoutPassword.setError("Minimal 6
karakter");
}

return valid;
}

// do register
public void doRegister(final String name, final String
username, final String email,final String password){
    Call<RegisterResponse> registerResponseCall =
RetrofitClient.getService().signup(name,username,email,pass
word);
    registerResponseCall.enqueue(new
Callback<RegisterResponse>() {
        @Override
        public void onResponse(Call<RegisterResponse>
call, Response<RegisterResponse> response) {

            if(response.isSuccessful()){
                //Login start main activity
                Snackbar.make(buttonRegister, "User
created successfully!", Snackbar.LENGTH_LONG).show();

SaveSharedPreference.setLoggedIn(getApplicationContext(),
true);
}
}
}
```

```
        SharedPreferences
preferences =
getSharedPreferences("preferences",
MODE_PRIVATE);
        SharedPreferences.Editor editor =
preferences.edit();
        editor.putString("username",
username); editor.commit();

        Intent i = new
Intent(RegisterActivity.this,
MainActivity.class);
        startActivity(
i); finish();

    }else {
        Snackbar.make(buttonRegister,
"Unable to register user, maybe you are already
registered", Snackbar.LENGTH_LONG).show();
    }
}

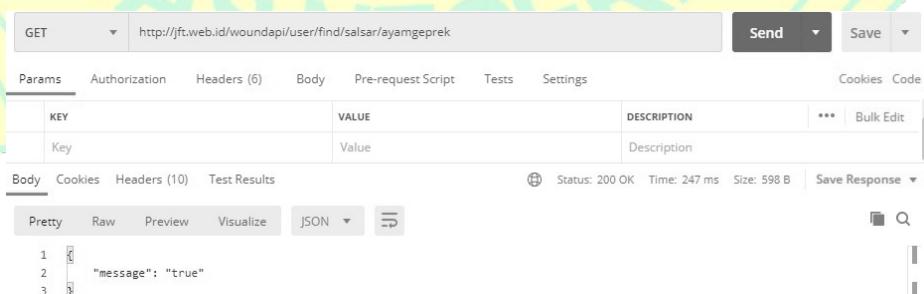
@Override
public void
onFailure(Call<RegisterResponse> call, Throwable t) {
    Toast.makeText(RegisterActivity.th
is, t.getLocalizedMessage(),
Toast.LENGTH_LONG).show();
}
});
```

Jika *response* yang diberikan merupakan kode HTTP 200 atau sukses, maka data akan dimasukkan ke dalam *database* dan pengguna akan diarahkan ke halaman utama atau *class* *MainActivity*.

8) Membuat REST API untuk validasi akun

```
@bp.route('/user/find/<username>/<passw>')
def find_user(username, passw):
    try:
        filter = {}
        filter["username"] = username
        filter["password"] = passw
        a = get_user(filter)
        if a == None:
            print(a)
            return Response(response =
json.dumps({"message" : "false"}),
mimetype="application/json", status=404)
        else:
            print(a)
            return Response(response =
json.dumps({"message" : "true"}),
mimetype="application/json", status=200)
    except Exception as ex:
        print (ex)
        return Response(response = json.dumps({"message"
: "false"}), mimetype="application/json", status=500)
```

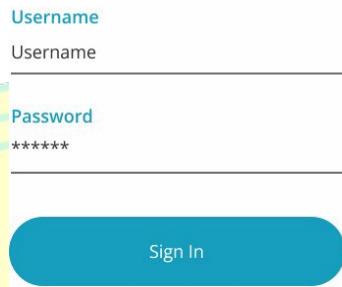
REST API validasi akun berfungsi untuk fitur *Sign In* dimana akan dilakukan pengecekan *username* dan *password*. Jika *username* dan *password* sudah ada dalam *database* maka akan dikembalikan pesan *true* dengan status HTTP 200 (OK). Routing untuk pemanggilan REST API validasi akun adalah `jft.web.id/user/find/<username>/<password>`. Agar REST API dapat terpanggil, diperlukan untuk mengunggah kembali *file* ke dalam *server*.



Gambar 4.5 Pemanggilan REST API validasi akun

9) *Mockup* tampilan *Sign In*

**Sign In**



Gambar 4.6 Tampilan laman *Sign In*

Gambar 4.6 ditampilkan setelah pengguna menekan tombol *Sign In* yang terdapat pada halaman awal (Gambar 3.5). Proses masuk ke akun dilakukan oleh pengguna yang telah membuat akun sebelumnya dan ingin mengakses fitur utama dari aplikasi.

10) Implementasi *layouting mockup* tampilan *Sign In* pada *Android*

Implementasi *layouting mockup* tampilan *Sign In* menggunakan *RelativeLayout* sebagai *parent view* untuk memberi *margin* kiri dan kanan serta *LinearLayout* sebagai tempat form dan *button*.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/loginForm"
    android:visibility="invisible"
    tools:context="com.example.chronicwound.LoginActivity"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginHorizontal="30dp"
        android:layout_marginTop="20
```

```
        dp”
        android:orientation=”vertica
l”>

        <TextView
            android:layout_width=”matc
h_parent”
            android:layout_height=”wrap_
content”
            android:layout_marginBotto
m=”40dp”
            android:text=”Sign In”

            android:textAppearance=”@style/TextAppearance.App
Compat.Dis play1”
            android:textColor
            =”#000000”
            android:textSize=
            ”34sp”
            android:layout_we
ight=”0”/>

        <!--      Username →

        <com.google.android.material.textfield.TextInputLayout
            style=”@style/Widget.MaterialComponents.TextInput
Layout.Out linedBox”
            android:id=”@+id/textInputLayou
tUserName”
            android:layout_width=”match_par
ent”
            android:layout_height=”wrap_ con
tent” app:errorEnabled=”true”
            android:hint=”Username”
            android:layout_weight=”0”
            android:layout_marginBottom=”8d
p”>

            <com.google.android.material.textfield.TextIn
putEditText
                android:layout_width=”mat
ch_parent”
                android:layout_height=”70
dp”
            />

        </com.google.android.material.textfield.TextInputLayout
>

        <!--      Password →

        <com.google.android.material.textfield.TextInputLayout
```

```
style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        android:id="@+id/textInputLayoutPassword"
        app:passwordToggleEnabled =
        "true"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" app:errorEnabled="true"
        android:hint="Password"
        android:layout_weight="0"
        android:layout_marginBottom="12
        dp">

<com.google.android.material.textfield.TextInputEditText
        android:id="@+id/editTextPassword" app:passwordToggleEnabled
        = "true"
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:inputType="textPassword"
        />

</com.google.android.material.textfield.TextInputLayout>

<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0d
        ip"
        android:layout_weight="1"
        android:orientation="vert
        ical">

<Button
        android:id="@+id/buttonLogin"
        android:layout_width="match_parent"
        android:layout_height="70
        dp"
        android:layout_gravity="center_hizont
        al|center"
        android:background="@drawable/primary_b
        utton">
```

```

        android:layout_marginBottom
        tom="20dp"
        android:text="Sign In"
        />
    </LinearLayout>
</LinearLayout>
</RelativeLayout>
```

- 11) Implementasi logic di FrontEnd untuk pemanggilan REST API validasi akun

Disebabkan *class* untuk membuat *client* Retrofit sudah dibuat, maka selanjutnya diberi penambahan *Routing* untuk pemanggilan REST API validasi akun pada *class* userService.

```

package com.example.chronicwound.remote;

public class LoginRequest
{ private String
    username; private
    String password;

    public String getUsername()
        { return username;
    }

    public void setUsername(String username)
        { this.username = username;
    }

    public String getPassword()
        { return password;
    }

    public void setPassword(String password)
        { this.password = password;
    }
```

Kode di atas merupakan pemodelan data yang dibutuhkan tersimpan pada *class* LoginRequest sedangkan *response* yang diberikan terdapat pada *class* LoginResponse pada kode di bawah.

*Class controller* akan dipanggil pada fungsi doLogin() yang terdapat pada *class* LoginActivity di bawah. Fungsi tersebut digunakan ketika *button* ditekan oleh pengguna.

```

package com.example.chronicwound;

import
android.content.SharedPreferences;
import android.os.Bundle;
import
android.view.View;
```

```
import
    android.widget.Button;
import
    android.widget.RelativeLayout;
import
    android.widget.Toast;
import
    android.content.Intent;
import
    android.widget.EditText;
import
    androidx.appcompat.app.AppCompatActivity;
import
    com.example.chronicwound.remote.LoginResponse;
import
    com.example.chronicwound.remote.RegisterResponse;
import
    com.example.chronicwound.remote.ResObj;
import
    com.example.chronicwound.remote.RetrofitClient;
import
    com.example.chronicwound.remote.UserService;
import
    com.google.android.material.snackbar.Snackbar;
import
    com.google.android.material.textfield.TextInputLayout;
import
    retrofit2.Call;
import
    retrofit2.Callback;
import
    retrofit2.Response;
public class LoginActivity extends AppCompatActivity {
    UserService userService;
    //Declaration
    EditTexts
    EditText editTextUserName
    ; EditText
    editTextPassword
    ;
```

```
//Declaration
TextInputLayout
TextInputLayout
textInputLayoutUser
Name;
TextInputLayout
textInputLayoutPass
word;

//Declaration Layout
RelativeLayout loginForm;

//Declaration Button
Button buttonLogin;

@Override
protected void onCreate(Bundle
    savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.tam
    pilan_Login);

    loginForm = findViewById(R.id.LoginForm);
    // Check if UserResponse is Already Logged In

    if(SaveSharedPreference.getLoggedStatus(getApplicationContext())) {
        Intent intent = new
Intent(getApplicationContext(), MainActivity.class);
        startActivity(intent);
    } else {
        loginForm.setVisibility(View.VISIBLE);
    }

    initViews();
    //set click event of login button
    buttonLogin.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String username =
editTextUserName.getText().toString();
            String password =
editTextPassword.getText().toString();

            //Check user input is correct or not
            if (validate(username,password)) {
                //do login
                doLogin(username, password);

            }
        }
    });
}
```

```

//this method is used to connect XML views to its
Objects
private void initViews() {
    editTextUserName = (EditText)
findViewById(R.id.editTextUserName);
    editTextPassword = (EditText)
findViewById(R.id.editTextPassword);
    TextInputLayoutUserName = (TextInputLayout)
findViewById(R.id.textInputLayoutUserName);
    TextInputLayoutPassword = (TextInputLayout)
findViewById(R.id.textInputLayoutPassword);
    buttonLogin = (Button)
findViewById(R.id.buttonLogin);
    loginForm = findViewById(R.id.LoginForm);
}

//This method is used to validate input given by user
public boolean validate(String username, String
password) {
    if(username == null || username.trim().length() ==
0){
        TextInputLayoutUserName.setError("Username
tidak boleh kosong");
        return false;
    }
    if(password == null || password.trim().length() ==
0){
        TextInputLayoutPassword.setError("Password
tidak boleh kodong");
        return false;
    }
    return true;
}

//do Login class
private void doLogin(final String username,final String
password){
    Call<LoginResponse> loginResponseCall =
RetrofitClient.getService().login(username,password);
    loginResponseCall.enqueue(new
Callback<LoginResponse>() {
        @Override
        public void onResponse(Call<LoginResponse>
call, Response<LoginResponse> response) {

            if(response.isSuccessful()){
                //Login start main activity
                Snackbar.make(buttonLogin, "Login
success", Snackbar.LENGTH_LONG).show();

SaveSharedPreference.setLoggedIn(getApplicationContext(),
true);
}
}
}

```

```
        SharedPreferences preferences =
getSharedPreferences("preferences", MODE_PRIVATE);
        SharedPreferences.Editor editor =
preferences.edit();
        editor.putString("username", username);
        editor.commit();

        Intent i = new
Intent(LoginActivity.this, MainActivity.class);
        startActivity(i);
        finish();

    }else {
        Snackbar.make(buttonLogin, "Unable to
register user, maybe you are already registered",
Snackbar.LENGTH_LONG).show();
    }
}

@Override
public void onFailure(Call<LoginResponse> call,
Throwable t) {
    Toast.makeText(LoginActivity.this,
t.getLocalizedMessage(), Toast.LENGTH_LONG).show();
}
}
```

Fungsi `doLogin()` memerlukan parameter berupa `username` dan `password`. Parameter tersebut akan dimasukkan ke dalam `path` REST API validasi akun. Jika respon yang diterima sukses atau kode 200, maka pengguna akan langsung diarahkan ke halaman utama pengkajian luka. Jika respon yang diterima berupa selain HTTP 200, maka pengguna akan diminta untuk mengecek kembali `username` dan `password` yang dimasukkan.

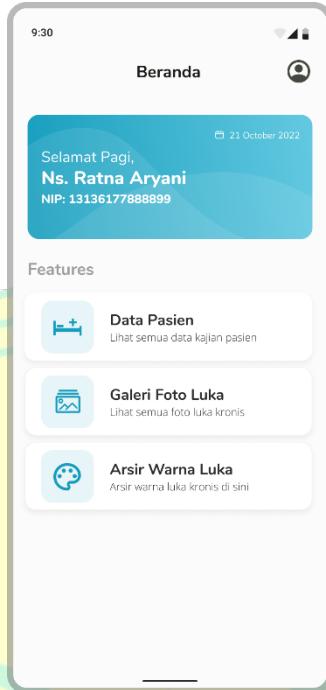
### b. Sprint-2

Tabel 4.2 Sprint-2 Backlog

No.	User Story	Task	Status
1	Menambah pasien dan memilih pasien	Membuat mockup tampilan beranda dari aplikasi	Selesai
		Membuat mockup tampilan penambahan pasien	Selesai
		Membuat mockup tampilan daftar pasien	Selesai
		Membuat mockup tampilan detail pasien	Selesai
		Membuat desain database pasien	Selesai
		Membuat routing table pasien	Selesai
		Membuat REST API di Flask terkait data pasien sesuai dengan <i>routing table</i>	Selesai
		Implementasi tampilan layout pada Android	Selesai
		Implementasi logic untuk pemanggilan REST API penambahan pasien baru	Selesai
		Implementasi logic untuk pemanggilan REST API menampilkan daftar pasien	Selesai

#### 1) Mockup tampilan beranda dari aplikasi

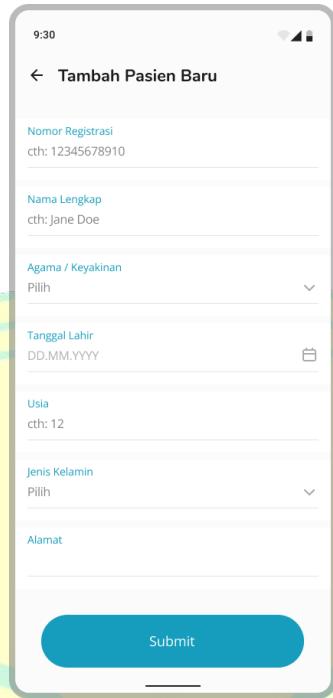
Setelah pengguna melakukan proses autentikasi dengan *Sign In* atau membuat akun, pengguna akan diarahkan ke laman beranda, yaitu laman yang memuat fitur-fitur yang dapat diakses dan melihat detail profil dari pengguna tersebut.



Gambar 4.7 Tampilan Laman Beranda

2) Mockup tampilan penambahan pasien

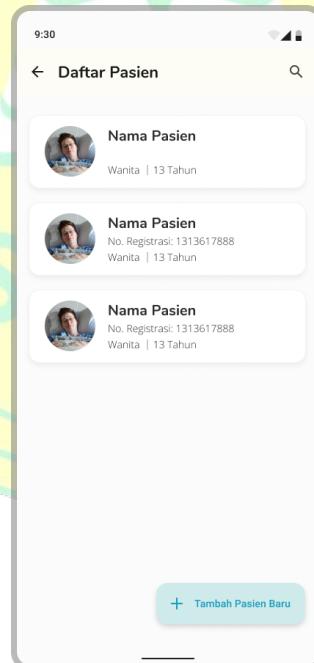
Tampilan penambahan pasien nantinya akan digunakan oleh pengguna atau perawat untuk menambahkan pasien baru. Perawat akan diminta untuk mengisi formulir nomor registrasi, nama lengkap, agama / keyakinan, tanggal lahir, usia, jenis kelamin, alamat, nomor hp, dan alamat surel.



Gambar 4.8 Mockup Tampilan Penambahan Pasien

### 3) Mockup tampilan daftar pasien

Daftar pasien memuat semua pasien yang memiliki kajian luka kronis. Seluruh daftar pasien dapat dilihat oleh semua pengguna yang terdaftar pada sistem.



Gambar 4.9 Mockup Tampilan Daftar Pasien

#### 4) Mockup tampilan detail pasien

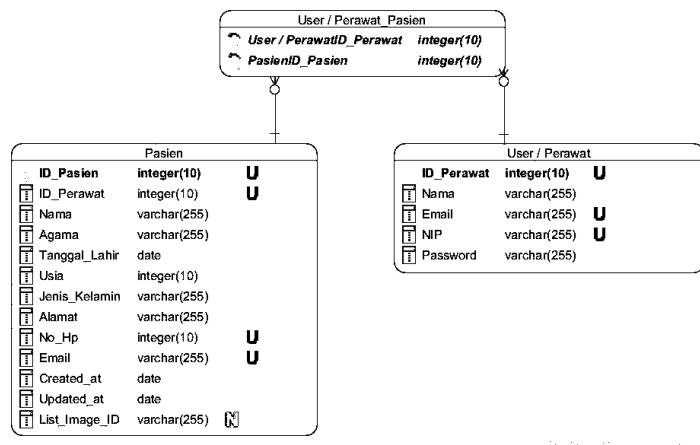
Detail pasien adalah halaman dimana informasi umum, histori kajian luka pasien, dan galeri foto luka pasien ditampilkan. Tampilan ini dibuat agar mempermudah perawat untuk melihat data pasien serta histori pengkajian luka pasien.



Gambar 4.10 Mockup Tampilan Detail Pasien Bagian Informasi Umum

#### 5) Desain database pasien

Tabel pasien memuat beberapa atribut yaitu ID\_Pasien, ID\_Perawat, nama, agama, tanggal lahir, usia, jenis kelamin, alamat, nomor hp, e-mail, created at, updated at, dan list image id. Tabel pasien dan tabel user mempunyai relasi many to many dimana satu pasien dapat ditangani oleh banyak perawat dan perawat dapat menangani banyak pasien. Tabel atau collection pasien akan dibuat bersamaan dengan pemanggilan REST API penambahan pasien baru. Pada sprint-2, atribut user yang tadinya adalah username diubah menjadi NIP.



Gambar 4.11 Desain Database Pasien

#### 6) *Routing table* pasien

Informasi mengenai jalur data yang akan digunakan melalui REST API dapat dilihat pada tabel di bawah ini.

Tabel 4.3 *Routing Table* Pasien

<i>Group</i>	<i>Name</i>	<i>API Endpoint</i>	<i>HTTP Verb</i>	Keterangan
Pasien	INDEX	/pasien	GET	menampilkan seluruh pasien
	READ	/pasien/<nomor_reka_medis>	GET	menampilkan data pasien berdasarkan nomor rekam medis
	CREATE	/pasien	POST	menambahkan pasien baru
	DESTROY	/pasien/<nomor_reka_medis>	DELETE	menghapus data pasien berdasarkan nomor rekam medis pasien

#### 7) REST API di Flask terkait data pasien sesuai dengan *routing table*

Pembuatan REST API dilakukan setelah routing table selesai dibuat. REST API dibuat menggunakan Flask dan Python yang diunggah ke server agar dapat diakses melalui Android.

- REST API menampilkan seluruh pasien

```

jft.web.id/woundapi / GET ALL PASIEN
GET http://jft.web.id/woundapi/pasien...
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Body Cookies Headers (11) Test Results
Pretty Raw Preview Visualize JSON ...
1 [
2   {
3     "_id": "1",
4     "id_perawat": 8200001,
5     "nama": "Arifin Amran",
6     "usia": "59",
7     "berat": "88",
8     "tinggi": "168"
9   },
10   {
11     "_id": "2",
12     "id_perawat": 8200001,
13     "nama": "Jane Doe",
14     "usia": "32",
15     "berat": "64",
16     "tinggi": "165"
17   },
18   {
19     "_id": "3",
20     "id_perawat": 8200002,
21     "nama": "Janita",
22     "usia": "53",
23     "berat": "63",
24     "tinggi": "164"
25   }
]

```

Gambar 4.12 REST API Semua Pasien

- REST API menampilkan data pasien berdasarkan nomor rekam medis

```

jft.web.id/woundapi / GET ALL PASIEN
GET http://jft.web.id/woundapi/pasien/012
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Body Cookies Headers (11) Test Results
Pretty Raw Preview Visualize JSON ...
1 {
2   "_id": "012",
3   "id_perawat": "8200001",
4   "nama": "Arifin Foster",
5   "agama": "Islam",
6   "born_date": "23/12/1999",
7   "usia": "23",
8   "kelamin": "Perempuan",
9   "alamat": "Jl. Jl tanah ga nyape-nyape nomor 13 blok c2 rt003",
10  "email": "0312101@gmail.com",
11  "created_at": "24/09/2022 08:46:01",
12  "updated_at": "24/09/2022 08:46:01",
13  "list_image": [
14    "97f7e071941432207e7x1d61c02x7",
15    "c97f7e071941432207e7x1d61c02x7",
16    "a85465579fa446868aa14160580887",
17    "8ea4a48ab706a49b28673743734808059",
18    "98a4a48ab706a49b28673743734808059",
19    "98a4a48ab706a49b28673743734808059",
20    "98a4a48ab706a49b28673743734808059",
21    "775080578ca339a464564584c55d",
22    "d4d4212c4a77479a0cef119b321423",
23    "9494275c9023149979946364744d44d4",
24    "71109x707064479979379494cc8d81",
25    "0bdaadef4f5841459959925e75cd7326815",
26    "051a248441599499499423080118",
27    "9494275c9023149979946364744d44d4",
28    "0d33909ec1410790210bf4f594271",
29    "34f398fb039744859050180a3017cA3H",
30    "080205958824441afCEB6824CB8621",
31    "224405958824441afCEB6824CB8621",
32    "82f9212133573443186ff739377157",
33    "0ffC9174669487470504f19031271097",
34    ...
35  ]
}

```

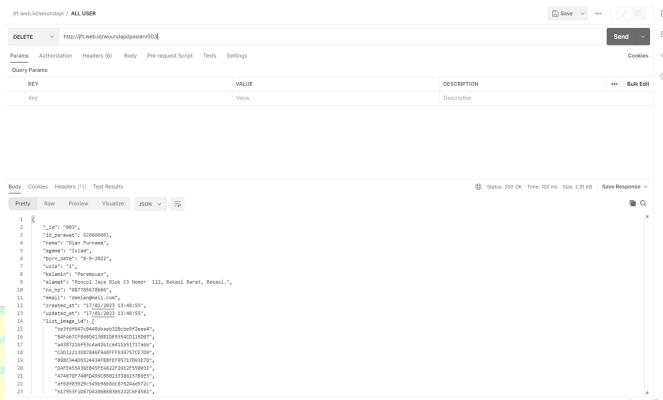
Gambar 4.13 REST API Pasien Berdasarkan NRM

- REST API menambahkan pasien baru

KEY	VALUE	DESCRIPTION	Bulk Edit
<input checked="" type="checkbox"/> _id	09		
<input checked="" type="checkbox"/> id_perawat	8200001		
<input checked="" type="checkbox"/> nama	Arifin Amran		
<input checked="" type="checkbox"/> agama	Islam		
<input checked="" type="checkbox"/> born_date	23/12/1999		
<input checked="" type="checkbox"/> usia	23		

Gambar 4.14 REST API Menambahkan Pasien Baru

- REST API menghapus data pasien berdasarkan nomor rekam medis



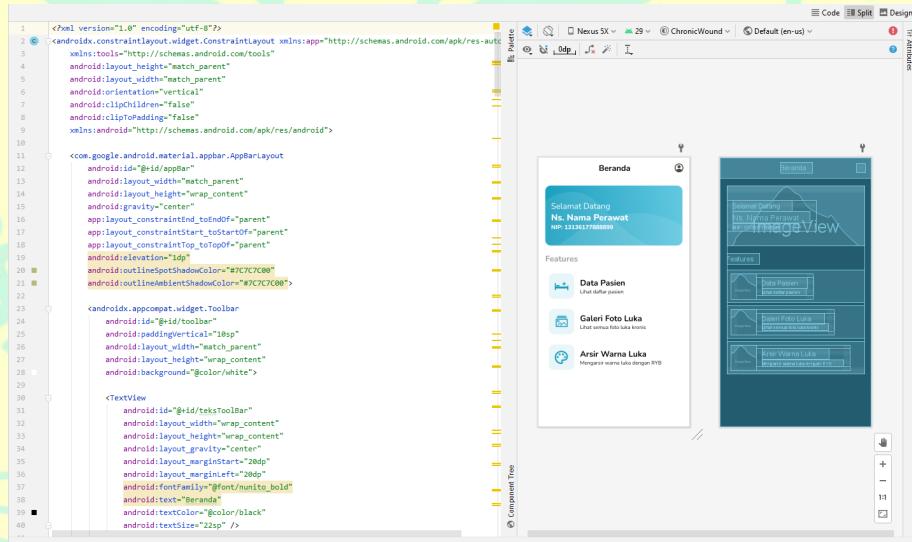
Gambar 4.15 REST API Menghapus Data Pasien

Dari gambar di atas, menunjukkan bahwa REST API seluruhnya berhasil dibuat karena mengembalikan status 200 (OK).

#### 8) Implementasi tampilan layout pada Android

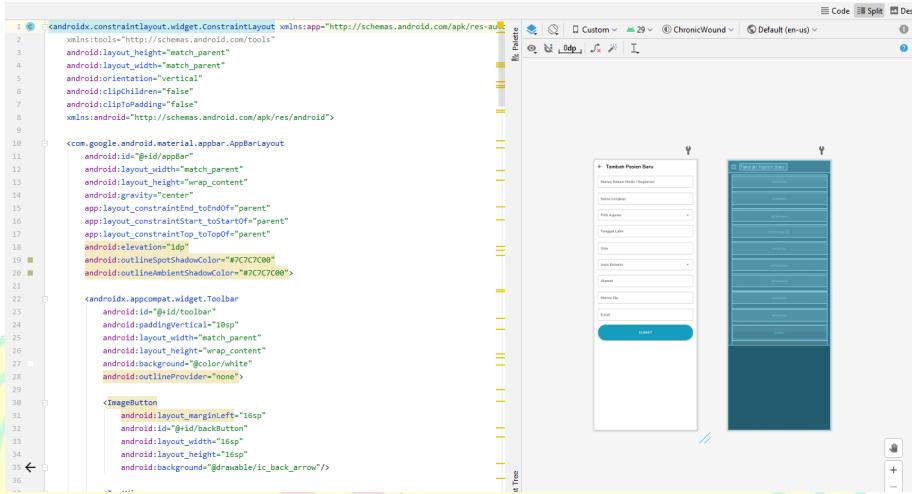
Tampilan *mockup* beranda, menambahkan pasien, dan tampilan detail pasien selanjutnya diimplementasikan pada Android dengan menggunakan XML.

- Implementasi tampilan beranda pada Android XML



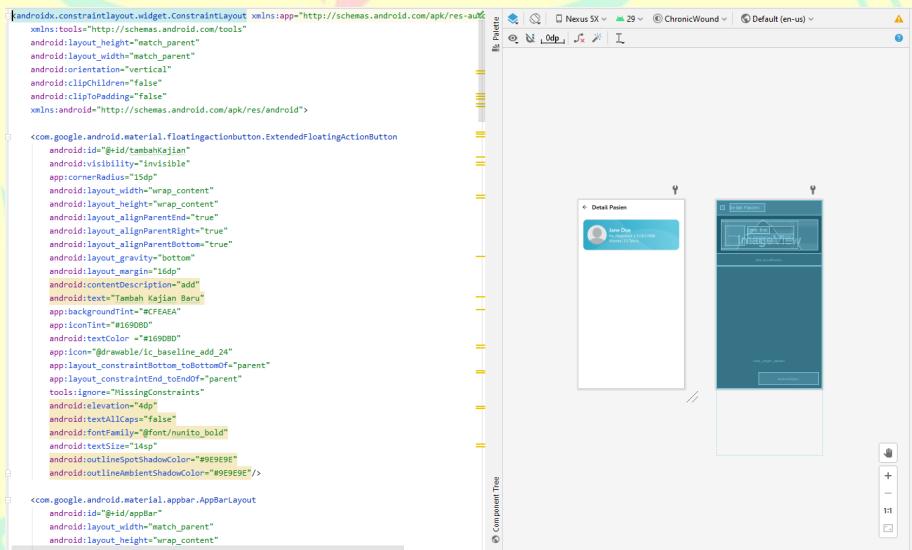
Gambar 4.16 Implementasi Tampilan Beranda pada XML

- Implementasi tampilan penambahan pasien pada Android XML



Gambar 4.17 Implementasi Tampilan Tambah Pasien pada XML

- Implementasi tampilan detail pasien pada Android XML



Gambar 4.18 Implementasi Tampilan Detail Pasien pada XML

- 9) Implementasi logic untuk pemanggilan REST API penambahan pasien baru

Dalam implementasinya terlebih dahulu dibuatkan *Routing* untuk pemanggilan REST API validasi akun pada *class* userService.

```
//register pasien versi baru
@FormUrlEncoded
@POST("pasien")
Call<PasienResponse> createPasien(@Field("_id") String
_id, @Field("id_perawat") String id_perawat,
                                         @Field("nama") String
nama, @Field("agama") String agama, @Field("born_date")
String born_date,
```

```
        @Field("usia") String  
usia, @Field("kelamin") String kelamin,  
@Field("alamat") String alamat,  
                @Field("no_hp") String  
no_hp, @Field("email") String email);
```

Kemudian dibuat pemodelan data yang dibutuhkan lalu disimpan pada *class Pasien Response*.

```
package com.example.chronicwound.remote;  
  
public class PasienResponse {  
    private String _id;  
    private String id_perawat;  
    private String nama, agama, born_date, usia,  
    kelamin, alamat, no_hp, email, created_at, updated_at;  
  
    public String get_id() {  
        return _id;  
    }  
  
    public void set_id(String _id) {  
        this._id = _id;  
    }  
  
    public String getId_perawat() {  
        return id_perawat;  
    }  
  
    public void setId_perawat(String id_perawat) {  
        this.id_perawat = id_perawat;  
    }  
  
    public String getNama() {return nama;}  
  
    public void setNama(String nama) {this.nama =  
nama;}  
  
    public String getAgama() {return agama;}  
  
    public void setAgama(String agama){this.agama=  
agama;}  
  
    public String getBorn_date(){return born_date;}  
  
    public void setBorn_date(String born_date)  
{this.born_date = born_date;}  
  
    public String getUsia(){return usia;}  
  
    public void setUsia(String usia){this.usia = usia;}  
  
    public String getKelamin(){return kelamin;}}
```

```

        public void setKelamin(String kelamin){this.kelamin
= kelamin;}

        public String getAlamat(){return alamat;}

        public void setAlamat(String alamat){this.alamat =
alamat;}

        public String getNo_hp(){return no_hp;}

        public void setNo_hp(String no_hp){this.no_hp =
no_hp;}

        public String getEmail(){return email;}

        public void setEmail(String email){this.email =
email;}

        public String getCreated_at(){return created_at;}

        public void setCreated_at(String
created_at){this.created_at = created_at;}

        public String getUpdated_at(){return updated_at;}

        public void setUpdated_at(String
updated_at){this.updated_at = updated_at;}


    }
}

```

*Class controller* untuk memanggil REST API terdapat Click Listener Button Submit. Ketika Button Submit ditekan oleh pengguna, maka data pasien akan langsung dikirimkan ke *database*.

```

buttonSubmit.setOnClickListener(new View.OnClickListener() {
    @RequiresApi(api = Build.VERSION_CODES.GINGERBREAD)
    @Override
    public void onClick(View view) {
        InsertLog(id_nurse, "Menekan tombol submit pada
halaman tambah pasien");

        String nrm = editTextNRM.getText().toString();
        String id_perawat = IDperawat.toString();
        String nama = editTextNama.getText().toString();
        String agama = OpsiAgama.getText().toString();
        String born_date =
editTextTanggalLahir.getText().toString();
        String usia = editTextUsia.getText().toString();
        String kelamin =
OpsiKelamin.getText().toString();
        String alamat =
editTextAlamat.getText().toString();
        String no_hp = editTextNoHp.getText().toString();
        String email =
editTextEmail.getText().toString();
    }
}

```

```

        //PasienRequest pasien = new PasienRequest(nrm,
        id_perawat, nama, agama, born_date, usia, kelamin, alamat,
        no_hp, email);
        Call<PasienResponse> call =
RetrofitClient.getService().createPasien(nrm, id_perawat,
        nama, agama, born_date, usia, kelamin, alamat, no_hp, email);
        call.enqueue(new Callback<PasienResponse>() {
            @Override
            public void onResponse(Call<PasienResponse>
call, Response<PasienResponse> response) {
                if(response.isSuccessful()){
                    //Login start main activity
                    Snackbar.make(buttonSubmit, "Patient
created successfully!", Snackbar.LENGTH_LONG).show();
                    Intent i = new
Intent(tambahPasienActivity.this, listPasienActivity.class);
                    i.putExtra(KEY_USERNAME, IDperawat);
                    startActivity(i);
                    finish();
                } else {
                    Snackbar.make(buttonSubmit, "Unable
to add new patient", Snackbar.LENGTH_LONG).show();
                }
            }

            @Override
            public void onFailure(Call<PasienResponse>
call, Throwable t) {
                call.cancel();
                Toast.makeText(getApplicationContext(),
                "input pasien gagal", Toast.LENGTH_SHORT).show();
            }
        });
    }
});
}

```

- 10) Implementasi *logic* untuk pemanggilan REST API menampilkan daftar pasien

*Routing* yang digunakan untuk menerima seluruh daftar pasien adalah <http://jft.web.id/woundapi/pasien>, daftar diterima dalam bentuk ArrayList yang mana harus didefinisikan pada Retrofit dengan mendeklarasi penggunaan ArrayList. Kode di bawah adalah implementasi pada kelas UserService.

```

//semua pasien
@GET("pasien")
Call<ArrayList<PasienResponse>> getAllCourses();

```

Karena daftar pasien masih menggunakan model yang sama yaitu PasienResponse, maka tidak dibutuhkan lagi untuk membuat *class*

model yang baru. Adapun logic untuk pemanggilan REST API terdapat pada *activity* listPasienActivity.java.

- 11) Implementasi *logic* untuk pemanggilan REST API menampilkan detail pasien

Guna menampilkan detail pasien, diperlukan input pada URL berupa Nomor Rekam Medis milik pasien tersebut pada *route* <http://jft.web.id/woundapi/pasien/<NRM>>. Kolom `@Path` perlu dikembangkan pada pembuatan *routing* Retrofit dengan maksud untuk memberi tahu bahwa input akan dikirimkan melalui *path url*.

```
//data detail pasien
@GET("pasien/{nrm}")
Call<PasienResponse> cariPasienNRM(@Path("nrm") String nrm);
```

*Logic* pemanggilan REST API terdapat pada *activity* detailPasienActivity.java melalui fungsi cariPasien(String nrm). Fungsi ‘CariPasien’ hanya memerlukan input berupa Nomor Rekam Medis dengan tipe *String*.

#### 12) Source Code

Seluruh kode untuk *sprint-2* dapat dilihat pada link berikut:

- Android: <https://github.com/apraira/Chronic-Wound-Dataset/tree/sprint-2>
- Web Service: <https://github.com/apraira/woundapi/tree/sprint-2>

#### c. Sprint-3

Tabel 4.4 Sprint-3 Backlog

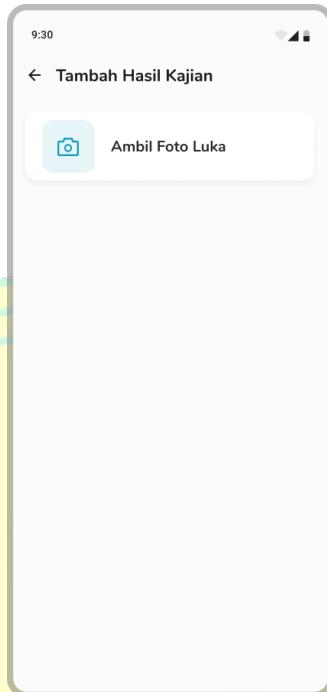
No.	User Story	Task	Status
1	Manajemen fotografi luka	Membuat tampilan UI tombol Kamera	Selesai
		Implementasi fungsi UI tombol Kamera untuk mengambil gambar luka melalui kamera	Selesai
		Membuat desain <i>database image</i>	Selesai

No.	User Story	Task	Status
		Implementasi <i>database image</i>	Selesai
		Membuat <i>routing table</i>	Selesai
		Membuat REST API untuk menyimpan <i>image</i> ke server	Selesai
		Membuat REST API untuk menampilkan <i>image</i>	Selesai
		Menampilkan gambar luka yang telah diambil melalui kamera pada <i>ImageView</i>	Selesai
		Membuat mockup tampilan galeri foto luka pada 1 pasien	Selesai
		Menghapus gambar luka (menu integrasi ke view dan galeri)	Next Sprint
		Galeri foto luka (secara umum untuk 1 pasien)	Next Sprint
		Menyimpan gambar luka ke server	Next Sprint

1) Membuat tampilan tombol kamera

Tombol kamera pada aplikasi yang akan dibuat berfungsi untuk mengambil foto luka kronis sebelum dilakukannya pengkajian luka. Peneliti terlebih dahulu membuat desain dari tombol kamera yang nantinya akan diterapkan pada fungsinya. Hasil desain daripada tombol kamera dapat dilihat pada Gambar 4.21.

Desain yang telah dibuat, kemudian dibuat implementasinya pada Android. Dalam implementasinya digunakan Card View sebagai wadah yang berisikan LinearLayout berorientasi horizontal, dimana pada sisi kiri ditambahkan ikon kamera dan pada sisi kanan ditambahkan tulisan keterangan “Ambil Foto Luka”.



Gambar 4.19 Tampilan Tombol Kamera

## 2) Implementasi fungsi tombol kamera

Untuk membaca media pada aplikasi, maka dapat menambahkan *permission* pada Android Manifest pada kode di bawah ini:

```
<uses-permission
    android:name="android.permission.READ_EXTERNAL_STORAGE"
/>
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    />
<uses-permission
    android:name="android.permission.MANAGE_EXTERNAL_STORAGE"
    />
```

Selanjutnya agar tombol kamera dapat membuka halaman kamera, digunakan *class* MediaStore berisikan seluruh metadata media yang tersedia di perangkat. Secara spesifik, *collection* yang digunakan adalah MediaStore.ACTION\_IMAGE\_CAPTURE. Saat *collection* ini dipanggil aplikasi kamera akan terbuka dan mengembalikannya baik dalam format File, URI, dan Bitmap. Fungsi ini diterjemahkan pada cameraIntent() dan *output* ditangani oleh onActivityResult. Setelah itu fungsi cameraIntent() dimasukkan ke dalam ClickListener dari tombol kamera.

```
CardView fab = findViewById(R.id.cameraButton);

fab.setOnClickListener(new View.OnClickListener() {
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    public void onClick(View view) {
        InsertLog(id_nurse, "Memasuki halaman untuk mengambil
foto luka");
        cameraIntent();
    }
});
@Override
public void onRequestPermissionsResult(int requestCode,
@NonNull String[] permissions, @NonNull int[] grantResults)
{
    super.onRequestPermissionsResult(requestCode,
permissions, grantResults);
    if (requestCode == MY_CAMERA_PERMISSION_CODE)
    {
        if (grantResults[0] ==
PackageManager.PERMISSION_GRANTED)
        {
            Toast.makeText(this, "camera permission granted",
Toast.LENGTH_LONG).show();
            Intent cameraIntent = new
Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
            startActivityForResult(cameraIntent,
CAMERA_REQUEST);
        }
        else
        {
            Toast.makeText(this, "camera permission denied",
Toast.LENGTH_LONG).show();
        }
    }
}

@RequiresApi(api = Build.VERSION_CODES.GINGERBREAD)
private void cameraIntent() {
    InsertLog(id_nurse, "Membuka kamera");
    StrictMode.VmPolicy.Builder builder = new
StrictMode.VmPolicy.Builder();
    StrictMode.setVmPolicy(builder.build());
    Intent intent = new Intent();
    intent.setAction(MediaStore.ACTION_IMAGE_CAPTURE);

    Date date = new Date();
    DateFormat df = new SimpleDateFormat("-mm-ss");

    String newPicFile = df.format(date) + ".jpg";
    String outPath = "/sdcard/" + newPicFile;
    File outFile = new File(outPath);

    mCameraFileName = outFile.toString();
    Uri outuri = Uri.fromFile(outFile);
    intent.putExtra(MediaStore.EXTRA_OUTPUT, outuri);
    startActivityForResult(intent, 2);
}

@Override
public void onActivityResult(int requestCode, int resultCode,
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
```

```

        if (resultCode == Activity.RESULT_OK) {
            if (requestCode == 2) {
                if (data != null) {
                    image = data.getData();

                    Intent IntentCamera = new
Intent(tambahKajianActivity.this,
konfirmasiFotoActivity.class);
                    IntentCamera.putExtra(KEY_PHOTO, image);
                    IntentCamera.putExtra("id_perawat",
id_perawat);
                    startActivityForResult(IntentCamera);
                }
                if (image == null && mCameraFileName != null) {
                    File file = new File(mCameraFileName);

                    // save raw image ke shared preferences
                    SharedPreferences preferences =
getSharedPreferences("preferences", MODE_PRIVATE);
                    SharedPreferences.Editor editor =
preferences.edit();
                    editor.putString("rawImage",
file.getAbsolutePath().toString());

                    System.out.println(file.getAbsolutePath().toString());
                    editor.commit();
                    // end of save raw image ke shared
                    preferences

                    String rawPath =
file.getAbsolutePath().toString(); // Get the full path
                    System.out.println(rawPath);
                    /** upload image below line ***/
                    Bundle extras = getIntent().getExtras();
                    String id_pasien = NRM;
                    String category = "Raw";
                    String id_gambar =
UUID.randomUUID().toString().replaceAll("-",
 "").toUpperCase();
                    String filename = file.getName();

                    RequestBody requestBody =
RequestBody.create(MediaType.parse("multipart/form-data"),
file);

                    MultipartBody.Part body =
MultipartBody.Part.createFormData("image", file.getName(),
requestBody);
                    RequestBody id =
RequestBody.create(MediaType.parse("multipart/form-data"),
id_gambar);
                    RequestBody pasien_id =
RequestBody.create(MediaType.parse("multipart/form-
data"), id_pasien);
                    RequestBody perawat_id =
RequestBody.create(MediaType.parse("multipart/form-data"),
String.valueOf(id_perawat));
                    RequestBody kategori =
RequestBody.create(MediaType.parse("multipart/form-data"),
category));
    
```

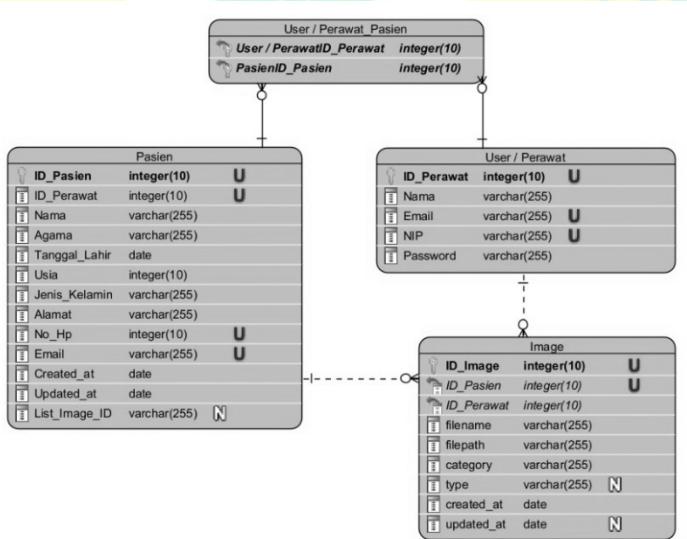
```
// uploadImage(body, id, pasien_id,
perawat_id, kategori);

image = Uri.fromFile(new File(rawPath));
Intent IntentCamera = new
Intent(tambahKajianActivity.this, anotasiTepi.class);
IntentCamera.putExtra(KEY_PHOTO, image);
IntentCamera.putExtra("raw_path", rawPath);
IntentCamera.putExtra("id_gambar",
id_gambar);
IntentCamera.putExtra("id_perawat",
id_perawat);
IntentCamera.putExtra("id_pasien",
id_pasien);
System.out.println("sent from tambah kajian
activity 1:" + id_gambar+ "," + id_perawat + "," +
id_pasien);
startActivity(IntentCamera);
}

}
}
```

### 3) Desain *database image*

Tabel *image* memuat beberapa atribut yaitu *ID\_Image*, *ID\_Pasien*, *ID\_Perawat*, *filename*, *filepath*, *category*, *type*, *created at*, dan *updated at*. Tabel *image* dan tabel *user* mempunyai relasi *one to many* dimana satu perawat dapat memiliki banyak *image*, namun satu *image* hanya dapat dimiliki oleh satu perawat, hal yang sama berlaku juga kepada relasi tabel pasien dan tabel *image*.



Gambar 4.20 Desain Database Image

#### 4) Implementasi *database image*

Database atau *collection image* akan dibuat bersamaan dengan pemanggilan REST API unggah foto luka dimana terdapat dalam kode python pymongo di bawah ini:

```
from typing import Collection
import click
import pymongo
from flask import current_app, g
from flask.cli import with_appcontext
import gridfs

def get_db():
    mongocon = current_app.config['MONGO_CON']
    dbclient = pymongo.MongoClient(mongocon)
    g.db = dbclient[current_app.config['DATABASE']]
    return g.db

def get_collection(colname):
    if 'db' not in g:
        get_db()
    return g.db[colname]

def insert_image(data):
    collection = get_collection("image")
    row = collection.insert_one(data)
    return row
```

#### 5) Routing table REST API *image*

Informasi mengenai jalur data yang akan digunakan melalui REST API dapat dilihat pada tabel di bawah ini.

Tabel 4.5 *Routing Table Image*

<b>Group</b>	<b>Name</b>	<b>API Endpoint</b>	<b>HTTP Verb</b>	<b>Keterangan</b>
Image	INDEX	/get_images	GET	menampilkan seluruh database image
	READ	/get_image/<id>	GET	menampilkan satu data image berdasarkan id image
	CREATE	/upload	POST	upload image baru ke folder static
	READ	image/find/<id_pasien>	GET	menampilkan semua image yang dimiliki satu pasien
	DESTROY	/delete_image/<id>	DELETE	menghapus satu data image pada database

## 6) REST API untuk menyimpan image ke server

REST API untuk menyimpan *image* ke *server* dapat diakses melalui URL <http://jft.web.id/woundapi/upload>. Dapat dilihat pada gambar 4.23 bahwa REST telah berhasil dibuat karena mengembalikan status 200.



Gambar 4.21 REST API Unggah *Image*

## 7) Membuat REST API untuk menampilkan image

Untuk menampilkan *image* dari server pada aplikasi Android, dibutuhkan *library* tambahan bernama Glide. *Library* ini memungkinkan pengembang untuk menampilkan gambar melalui URL dari server oleh REST API yang dipakai untuk mendapatkan seluruh isi dari database *image*.

```

#get all images data
@bp.route('/get_images', methods =['GET'])
def get_all_images():
    a = get_images()
    print(a)
    return Response(response = json.dumps(list(a)),
mimetype="application/json", status=200)

```

## 8) Menampilkan gambar luka yang telah diambil melalui kamera pada *ImageView*

```

@Override
public void onActivityResult(int requestCode, int resultCode,
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == Activity.RESULT_OK) {
        if (requestCode == 2) {
            if (data != null) {
                image = data.getData();

```

```

        Intent IntentCamera = new
Intent(tambahKajianActivity.this,
konfirmasiFotoActivity.class);
        IntentCamera.putExtra(KEY_PHOTO, image);
        IntentCamera.putExtra("id_perawat",
id_perawat);
        startActivityForResult(IntentCamera);
    }
    if (image == null && mCameraFileName != null) {
        File file = new File(mCameraFileName);

        // save raw image ke shared preferences
        SharedPreferences preferences =
getSharedPreferences("preferences", MODE_PRIVATE);
        SharedPreferences.Editor editor =
preferences.edit();
        editor.putString("rawImage",
file.getAbsolutePath().toString());

        System.out.println(file.getAbsolutePath().toString());
        editor.commit();
        // end of save raw image ke shared
        preferences

        String rawPath =
file.getAbsolutePath().toString(); // Get the full path
        System.out.println(rawPath);
        /** upload image below line ***/
        Bundle extras = getIntent().getExtras();
        String id_pasien = NRM;
        String category = "Raw";
        String id_gambar =
UUID.randomUUID().toString().replaceAll("-",
 "").toUpperCase();
        String filename = file.getName();

        RequestBody requestBody =
RequestBody.create(MediaType.parse("multipart/form-data"),
file);

        MultipartBody.Part body =
MultipartBody.Part.createFormData("image", file.getName(),
requestBody);
        RequestBody id =
RequestBody.create(MediaType.parse("multipart/form-data"),
id_gambar);
        RequestBody pasien_id =
RequestBody.create(MediaType.parse("multipart/form-
data"), id_pasien);
        RequestBody perawat_id =
RequestBody.create(MediaType.parse("multipart/form-data"),
String.valueOf(id_perawat));
        RequestBody kategori =
RequestBody.create(MediaType.parse("multipart/form-data"),
category);
        // uploadImage(body, id, pasien_id,
perawat_id, kategori);

        image = Uri.fromFile(new File(rawPath));
    }
}

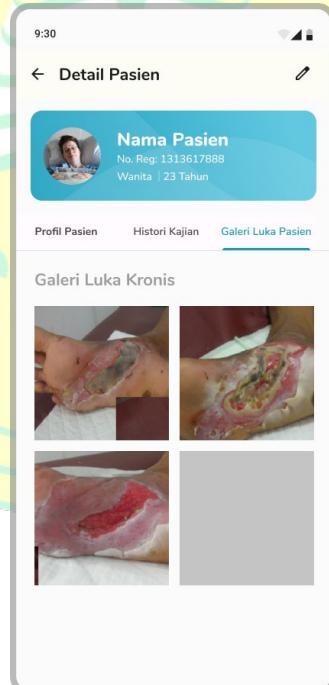
```

```
        Intent IntentCamera = new
Intent(tambahKajianActivity.this, anotasiTepi.class);
        IntentCamera.putExtra(KEY_PHOTO, image);
        IntentCamera.putExtra("raw_path", rawPath);
        IntentCamera.putExtra("id_gambar",
id_gambar);
        IntentCamera.putExtra("id_perawat",
id_perawat);
        IntentCamera.putExtra("id_pasien",
id_pasien);
        System.out.println("sent from tambah kajian
activity 1:" + id_gambar + "," + id_perawat + "," +
id_pasien);
        startActivityForResult(IntentCamera);
    }
}
}
```

Kode di atas berfungsi untuk menangani keluaran dari hasil foto kamera, dimana *filepath* akan dikirimkan ke *Activity* lain kemudian akan *di-load* sebagai Bitmap pada *ImageView*. Untuk merubah *filepath* menjadi digunakan *class* `BitmapFactory.decodeFile(filepath)` lalu *di-load* pada *ImageView* menggunakan perintah `setImageBitmap`.

```
Bitmap image = BitmapFactory.decodeFile(picturePath);
ImageView.setImageBitmap(image);
```

9) Membuat mockup tampilan galeri foto luka pada 1 pasien



Gambar 4.22 Mockup Tampilan Detail Pasien Bagian Galeri Luka

Tampilan galeri luka pasien adalah bagian dari tampilan detail pasien mengenai penyakit yang sedang dideritanya. Tampilan ini dapat diakses dengan menekan tab berjudul Galeri Luka Pasien.

#### 10) Menghapus gambar luka

Menghapus gambar luka artinya perlu menghapus file yang ada di penyimpanan *server* berserta dengan dokumen yang terdapat di database, oleh karena itu dibuat REST API yang dapat menghapus luka berdasarkan id.

```
#delete 1 image berdasarkan id
@bp.route('/delete_image/<id>', methods= ['DELETE'])
def delete_image(id):
    ide = id
    filename = search_filename_from_id(ide)
    filter = {}
    filter["_id"] = ide
    cek = get_image(filter)
    path = os.path.join(current_app.instance_path,
current_app.config['UPLOAD_DIR'])
    try:
        os.makedirs(path)
    except OSError:
        pass
    filepath = os.path.join(path, filename)
    os.remove(filepath)
    helper.delete_one_image(ide)
    return Response(response = json.dumps(dict(cek)),
mimetype="application/json", status=200)
```

Dari *input* id akan dicari data dokumen pada database, kemudian diambil *filename* beserta *filepathnya* yang akan dihapus menggunakan perintah `os.remove` yang tersedia ada Flask dan `delete_one` untuk menghapus data yang terdapat di database.

```
def delete_one_image(id):
    collection = get_collection("image")
    return collection.delete_one({"_id":id})
```

#### 11) Source code

Kode keseluruhan untuk *sprint-3* terdapat pada link berikut ini:

- Android: <https://github.com/apraira/Chronic-Wound->

### Dataset/tree/sprint-3

- Web Service: <https://github.com/apraira/woundapi/tree/sprint-3>

Karena waktu penggeraan tidak cukup, maka *task* dengan status

*Next Sprint* akan dikerjakan pada *sprint* selanjutnya yaitu *sprint-4*.

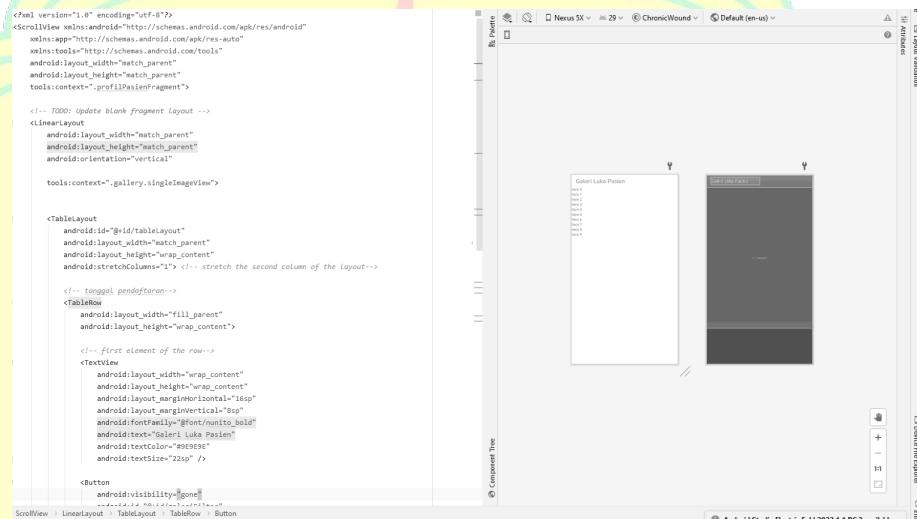
#### d. *Sprint-4*

Tabel 4.6 *Sprint-4 Backlog*

No.	User Story	Task	Status
1	Manajemen fotografi luka	Implementasi layout tampilan galeri foto luka pada 1 pasien pada XML	Selesai
		Menghapus gambar luka	Next Sprint
		Implementasi fungsi untuk menyimpan foto luka yang telah diambil melalui kamera ke server	Next Sprint
		Implementasi bagian load view image dari galeri pasien	Selesai
2	Melakukan penambahan pengkajian luka	Membuat mockup layout UI penambahan data pengkajian luka pada 1 pasien pada XML	Selesai
		Pembuatan desain database data kajian luka	Selesai
		Pembuatan database data kajian luka dengan flask	Selesai
		Pembuatan table routing	Selesai
		Membuat REST API untuk menambahkan kajian luka	Selesai
		Implementasi fungsi item kajian	Next Sprint
		memanggil webservice untuk menyimpan data	Next Sprint
3	Menganotasi pusat luka dan ditarik diameter ke tepi luka, diameter major dan minor	Membuat Mockup	Next Sprint
		Cari API untuk draw path untuk anotasi stroke, sama yang titik titik	Next Sprint

1) Implementasi *layout* tampilan galeri foto luka pada 1 pasien pada XML

Tampilan mockup galeri luka pasien, selanjutnya diimplementasikan pada Android dengan menggunakan XML. Karena tampilan galeri luka terdapat pada tab, maka tampilan ini akan diimplementasikan menggunakan *Fragment* yang kemudian ditampilkan di dalam *ViewPager* pada tampilan detail pasien.



Gambar 4.23 Implementasi Tampilan Galeri Luka Pasien

2) Implementasi bagian load view dari galeri pasien

Untuk menampilkan gambar dari server ke galeri luka, dibutuhkan RecyclerView berisikan ImageView. Nantinya akan dipanggil REST API untuk menerima semua data gambar yang dimiliki oleh pasien spesifik, lalu data URL dari gambar luka yang dimiliki pasien akan di-*load* menggunakan *library* Glide ke dalam ImageView.

Kode untuk adapter dari RecyclerView terdapat pada ImageAdapter.java di bawah ini. Pada kelas adapter inilah URL dari image didapat dan di-*load* ke dalam ImageView.

```

public class ImageAdapter extends
    RecyclerView.Adapter<com.example.chronicwound.gallery.ImageAd
    apter.ImageViewHolder> {
    private AdapterView.OnItemClickListener listener;
    private String KEY_NAME = "NRM"; //nomor registrasi
    pasien
    private String id_image, filename;
    galeriLukaPasien myFragment;

```

```

private ArrayList<GalleryRequest> dataList;

public ImageAdapter(ArrayList<GalleryRequest> dataList,
galeriLukaPasien galeriLukaPasien) {

    this.dataList = dataList;
}

@Override
public
com.example.chronicwound.gallery.ImageAdapter.ImageViewHolder
onCreateViewHolder(ViewGroup parent, int viewType) {
    LayoutInflater layoutInflater =
LayoutInflater.from(parent.getContext());
    View view =
layoutInflater.inflate(R.layout.galeri_card, parent, false);
    return new
com.example.chronicwound.gallery.ImageAdapter.ImageViewHolder
(view);
}

@Override
public void
onBindViewHolder(com.example.chronicwound.gallery.ImageAdapter.ImageViewHolder holder, int position) {
    final GalleryRequest imageModel =
(com.example.chronicwound.gallery.GalleryRequest)
dataList.get(position);
    String type = imageModel.getType();

    if ("Vector".equals(type)) {
        System.out.println(type + "Vector Image Detected
is not loaded by glide");
        /*
        SvgLoader.pluck()
            .with((Activity)
holder.itemView.getContext())
                .setPlaceholder(R.mipmap.ic_launcher,
R.mipmap.ic_launcher)

.load("https://jft.web.id/woundapi/instance/uploads/" +
imageModel.getFilename(), holder.imgView);*/
    } else {
        System.out.println(type + " is loaded by Glide");

Glide.with(holder.itemView.getContext()).load("https://jft.we
b.id/woundapi/instance/uploads/" + imageModel.getFilename())
    .centerCrop()
    .thumbnail(0.05f)

    .transition(DrawableTransitionOptions.withCrossFade())
        .into(holder.imgView);
    }

//Picasso.get().Load("https://jft.web.id/woundapi/instance/up
loads/" +
imageModel.getFilename()).resize(200,200).centerCrop().into(h
older.imgView);
}

@Override
public int getItemCount() {
}

```

```

        return (dataList != null) ? dataList.size() : 0;
    }

    public class ImageViewHolder extends
RecyclerView.ViewHolder{
    public ImageView imgView;

    public ImageViewHolder(View itemView) {
        super(itemView);
        imgView = (ImageView)
itemView.findViewById(R.id.iv_photo);

        itemView.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int position = getAdapterPosition();
                String IDImage =
dataList.get(position).getID();
                String IDPasien =
dataList.get(position).getId_pasien();
                Context context = v.getContext();
                Intent i = new Intent(context,
singleImageView.class);
                i.putExtra("IDPasien", IDPasien);
                i.putExtra(KEY_NAME, IDImage);
                context.startActivity(i);
            }
        });
    }
}

```

Selanjutnya, pemanggilan REST API untuk mendapatkan seluruh data *image* yang dimiliki pasien dipanggil pada *galeriLukaPasien.java* yang merupakan Fragment.

```

public class galeriLukaPasien extends Fragment {

    // TODO: Rename parameter arguments, choose names that
    // match
    // the fragment initialization parameters, e.g.
ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;
    private String NRM, id_perawat;
    private RecyclerView recyclerView;
    private com.example.chronicwound.gallery.ImageAdapter
adapter;
    private ArrayList<GalleryRequest> imageArrayList;

    public galeriLukaPasien() {
        // Required empty public constructor
    }
}

```

```

    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment galeriLukaPasien.
     */
    // TODO: Rename and change types and number of parameters
    public static galeriLukaPasien newInstance(String param1,
                                                String param2) {
        galeriLukaPasien fragment = new galeriLukaPasien();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater,
                            ViewGroup container,
                            Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        NRM = getArguments().getString("NRM");
        View inf =
        inflater.inflate(R.layout.fragment_galeri_luka_pasien,
                        container, false);
        recyclerView = (RecyclerView)
        inf.findViewById(R.id.rv_images);

        getImageURL(NRM);
        return inf;
    }

    private void getImageURL(String id_pasien) {
        Call<ArrayList<GalleryRequest>> pasienResponseCall =
        RetrofitClient.getService().getImageByID(id_pasien);

        // on below Line we are calling method to enqueue and
        // calling
        // all the data from array list.
        pasienResponseCall.enqueue(new
        Callback<ArrayList<GalleryRequest>>() {
            @Override
            public void
            onResponse(Call<ArrayList<GalleryRequest>> call,
            Response<ArrayList<GalleryRequest>> response) {
                // inside on response method we are checking
                // if the response is success or not.
                if (response.isSuccessful()) {

```

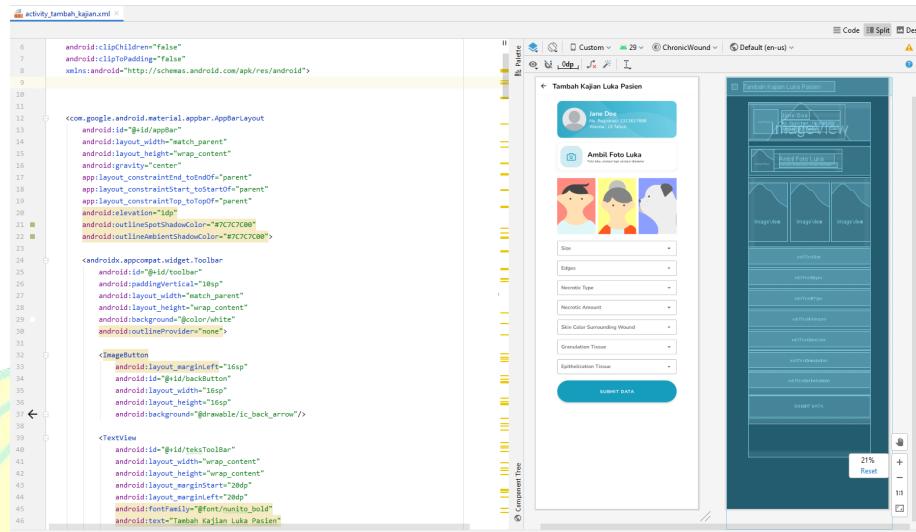
```
// below Line is to add our data from api  
to our array list.  
    imageArrayList = response.body();  
  
    // below Line we are running a Loop to  
add data to our adapter class.  
    for (int i = 0; i <  
imageArrayList.size(); i++) {  
        adapter = new  
ImageAdapter(imageArrayList,  
com.example.chronicwound.galeriLukaPasien.this);  
  
        int numberOfColumns = 2;  
  
        adapter = new  
com.example.chronicwound.gallery.ImageAdapter(imageArrayList,  
com.example.chronicwound.galeriLukaPasien.this);  
  
        RecyclerView.LayoutManager  
layoutManager = new GridLayoutManager(getContext(),  
numberOfColumns);  
  
recyclerView.setLayoutManager(layoutManager);  
  
        recyclerView.setAdapter(adapter);  
    }  
}  
  
@Override  
public void  
onFailure(Call<ArrayList<GalleryRequest>> call, Throwable t)  
{  
    // in the method of on failure we are  
displaying a  
    // toast message for fail to get data.  
    Toast.makeText(getContext(), "Fail to get  
data", Toast.LENGTH_SHORT).show();  
}  
});  
  
}  
}
```

- 3) Membuat layout tampilan penambahan data pengkajian luka pada 1 pasien pada XML

Layout penambahan kajian luka pada satu pasien sangat mirip dengan layout tambah pasien, bedanya terdapat tombol kamera dan terdapat preview image yang akan di-input.

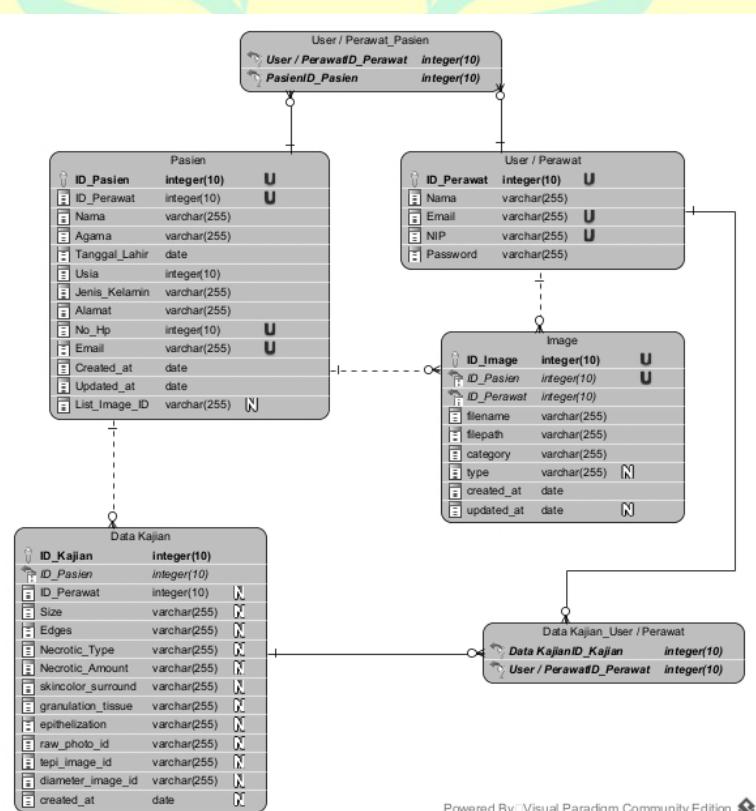
Pada awal akses ke halaman tambah kajian, formulir kajian beserta *preview* hasil anotasi akan hilang dan baru muncul lagi ketika user sudah melakukan proses foto luka, anotasi tepi, dan anotasi

diameter.



Gambar 4.24 Implementasi Tampilan Tambah Kajian

#### 4) Pembuatan desain database data kajian luka



Gambar 4.25 Desain Database Data Kajian Luka

Gambar 4.25 menunjukkan bahwa database dari data kajian luka memuat beberapa atribut umum diantaranya ID\_Kajian, ID\_Pasien, ID\_Perawat, dan created\_at. Selain itu pada tabel ini juga dimuat atribut yang mewakili skoring luka seperti size, edges, necrotic\_type, necrotic\_amount, skincolor\_surround, granulation tissue, epithelialization, raw\_photo\_id, tepi\_image\_id, dan diameter\_image\_id.

### 5) Pembuatan database data kajian luka dengan Flask

Fungsi-fungsi untuk menambahkan data kajian, menghapus data kajian dibuat menggunakan pymongo yang kemudian diaplikasikan dengan Flask.

```
import pymongo
from flask import current_app, g

def get_db():
    mongocon = current_app.config['MONGO_CON']
    dbclient = pymongo.MongoClient(mongocon)
    g.db = dbclient[current_app.config['DATABASE']]
    return g.db

def get_collection(colname):
    if 'db' not in g:
        get_db()
    return g.db[colname]

#semua kajian
def get_kajians(filter={}):
    collection = get_collection("kajian")
    return collection.find(filter)

#get data kajian by nrm pasien
def get_kajian_nrm(data):
    collection = get_collection("kajian")
    return collection.find(data)

#get 1 image
def get_kajian(filter={}):
    collection = get_collection("kajian")
    return collection.find_one(filter)

#tambah kajian baru
def insert_kajian(data):
```

```

collection = get_collection("kajian")
row = collection.insert_one(data)
return row

#delete satu data kajian berdasarkan id
def delete_one_kajian(id):
    collection = get_collection("kajian")
    return collection.delete_one({"_id":id})

def update_kajian(id, filter):
    collection = get_collection('kajian')

    #cursor ke data pasien sesuai dengan id
    x = collection.find_one({ '_id' : id })
    #query ke id pasien + update data
    #a = int(id)
    myquery = { '_id' : id }
    print(filter)
    newvalues = { '$set': filter }
    return collection.update_one(myquery, newvalues,
upsert=False)

def update_id_pasien_kajian(old_id, new_id):
    collection = get_collection('kajian')

    return collection.update_many(
        {"id_pasien": old_id },
        {
            "$set": { "id_pasien" : new_id}
        }
    )

```

#### 6) Pembuatan table routing

Informasi mengenai jalur data yang akan digunakan melalui REST API dapat dilihat pada tabel di bawah ini.

Tabel 4.7 Routing Table Data Kajian

<i>Group</i>	<i>Name</i>	<i>API Endpoint</i>	<i>HTTP Verb</i>	<i>Keterangan</i>
Kajian	INDEX	/get_kajians	GET	menampilkan seluruh database kajian luka
	READ	/get_kajian/<id>	GET	menampilkan data kajian berdasarkan id kajian
	CREATE	/insert_kajian	POST	menambahkan data kajian pada satu pasien

<i>Group</i>	<i>Name</i>	<i>API Endpoint</i>	<i>HTTP Verb</i>	Keterangan
	READ	/get_kajian/pasien/<n rm>	GET	get all kajian data berdasarkan nomor registrasi pasien
	DESTROY	/delete_kajian/<id>	DELETE	menghapus data kajian berdasarkan id kajian

- 7) Membuat REST API untuk menambahkan kajian luka  
 Kode di bawah ini adalah hasil dari implementasi pembuatan REST API dengan Flask.

```
bp = Blueprint('datakajian', __name__, url_prefix='/')

#insert data kajian
@bp.route('/insert_kajian', methods =[ 'POST'])
def post_kajian():

    #next save the file
    id = uuid.uuid4().hex
    id_pasien = request.form['id_pasien']

    try:

        data = {
            "_id" : id,
            "id_pasien": id_pasien,
            "id_perawat":
request.form['id_perawat'],
            "size":request.form['size'],
            "edges":request.form['edges'],
            "necrotic_type":request.form['necro
tic_type'],
            "necrotic_amount":request.form['nec
rotic_amount'],
            "skincolor_surround":request.form['
skincolor_surround'],
            "granulation":request.form['granula
tion'],
            "epithelization":request.form['epit
helization'],
            "raw_photo_id":
request.form['raw_photo_id'],
            "tepi_image_id":
request.form['tepi_image_id'],
            "diameter_image_id":
request.form['diameter_image_id'],
            "created_at" :
time.strftime("%d/%m/%Y %H:%M:%S")
```

```
        }
        insert_kajian(data)
        return Response(response = json.dumps(data),
mimetype="application/json", status=200)

    except Exception as ex:
        print (ex)
        return Response(response =
json.dumps({"message" : "error encountered"}),
mimetype="application/json", status=500)

#get all kajian data
@bp.route('/get_kajians', methods =['GET'])
def get_all_kajian():
    a = get_kajians()
    print(a)
    return Response(response = json.dumps(list(a)),
mimetype="application/json", status=200)

#get 1 kajian berdasarkan id kajian
@bp.route('/get_kajian/<id>', methods =['GET'])
def get_one_kajian(id):
    try:
        filter = {}
        filter["_id"] = id
        cek = get_kajian(filter)

        if cek == None:
            return Response(response =
json.dumps({"message" : "not found"}),
mimetype="application/json", status=404)
        else:
            print(cek)
            return Response(response =
json.dumps(dict(cek)), mimetype="application/json",
status=200)

    except Exception as ex:
        print("internal server error")
        return Response(response =
json.dumps({"message" : "false"}),
mimetype="application/json", status=500)

#delete 1 kajian berdasarkan id kajian
```

```
@bp.route('/delete_kajian/<id>', methods= ['DELETE'])
def delete_kajian(id):
    ide = id
    delete_one_kajian(ide)
    return Response(response = json.dumps({"message" :
"1 kajian deleted"}), mimetype="application/json",
status=200)

#get all kajian data berdasarkan nrm
@bp.route('/get_kajian/pasien/<nrm>', methods =[ 'GET'])
def get_pasien_kajian(nrm):
    data = {"id_pasien" : nrm}
    cek = get_kajian_nrm(data)

    a = []
    for doc in cek:
        a.append(doc)
    return Response(response = json.dumps(a),
mimetype="application/json", status=200)

#update data kajian
@bp.route('/kajian/update', methods =[ 'POST'])
def update_data_kajian():

    id_perawat = request.form['id_kajian']
    jenis = request.form['jenis']
    isian = request.form['isian']

    filter = {}
    filter[jenis] = isian

    try:
        update_kajian(id_perawat, filter)
        return Response(response =
json.dumps({"message" : "berhasil"}),
mimetype="application/json", status=200)

    except Exception as ex:
        print (ex)
        return Response(response =
json.dumps({"message" : "false"}),
mimetype="application/json", status=500)
```

8) *Source code*

- Android: <https://github.com/apraira/Chronic-Wound-Dataset/tree/sprint-4>
- Web Service: <https://github.com/apraira/woundapi/tree/sprint-4>

Karena waktu pengerjaan tidak cukup, maka task dengan status *Next Sprint* akan dikerjakan pada *sprint* selanjutnya yaitu *sprint-5*.

e. *Sprint-5*

Tabel 4.8 Sprint-5 Backlog

No.	User Story	Task	Status
1	Manajemen fotografi luka	Implementasi fungsi untuk menyimpan foto luka yang telah diambil melalui kamera ke server	Selesai
		Implementasi fungsi untuk menghapus foto luka pada Android	Selesai
2	Melakukan penambahan pengkajian luka	Implementasi fungsi input kajian Size luka	Selesai
		Implementasi fungsi input kajian Edges luka	Selesai
		Implementasi fungsi input kajian Necrotic Type luka	Selesai
		Implementasi fungsi input kajian Necrotic Amount luka	Selesai
		Implementasi fungsi input kajian Skin Color Surrounding Wound luka	Selesai
		Implementasi fungsi input kajian Granulation Tissue luka	Selesai
		Implementasi fungsi input kajian Epithelization luka	Selesai
		Menyimpan data kajian yang telah di-input ke server	Selesai
3	Menganotasi pusat luka dan ditarik diameter ke tepi luka, diameter major dan minor	Membuat mockup tampilan anotasi diameter luka	Selesai

- 1) Implementasi fungsi untuk menyimpan foto luka yang telah diambil melalui kamera ke server

Penyimpanan foto luka diimplementasikan menggunakan library Retrofit. Untuk mengirimkan data berupa file ke server, tidak bisa dilakukan dengan kolom bertipe Field namun menggunakan MultipartBody.Part.

```
// upload image
public void uploadImage( final MultipartBody.Part
image, final RequestBody id, final RequestBody
id_pasien, final RequestBody id_perawat, final
RequestBody type, final RequestBody category){
    Call<UploadRequest> uploadRequestCall =
RetrofitClient.getService().uploadImage(image, id,
id_pasien, id_perawat, type, category);
    uploadRequestCall.enqueue(new
Callback<UploadRequest>() {
        @Override
        public void onResponse(Call<UploadRequest>
call, Response<UploadRequest> response) {

            if(response.isSuccessful()){
                Toast.makeText(getApplicationContext(),
"Image uploaded to server", Toast.LENGTH_LONG).show();
                finish();

            }else {
                Toast.makeText(getApplicationContext(),
"gagal upload image" + category + type,
Toast.LENGTH_LONG).show();
            }
        }

        @Override
        public void onFailure(Call<UploadRequest> call,
Throwable t) {
            Toast.makeText(getApplicationContext(),
t.getLocalizedMessage(), Toast.LENGTH_LONG).show();
        }
    });
}
```

Kode di atas berfungsi untuk memanggil REST API *upload image* yang sebelumnya sudah diimplementasikan pada *class controller* UserService.java

```
@Multipart
@POST("upload")
Call<UploadRequest> uploadImage(@Part MultipartBody.Part image,
@Part("id") RequestBody id,
```

```

    @Part("id_pasien") RequestBody
    id_pasien,
    @Part("id_perawat")
    RequestBody id_perawat,@Part("type") RequestBody type,
    @Part("category")
    RequestBody category);

```

## 2) Implementasi fungsi untuk menghapus foto luka pada Android

Fungsi untuk menghapus foto luka diimplementasikan dengan memanggil REST API *delete image* dimana memerlukan input *id\_image* yang akan dihapus. Sebelum REST API dapat dipanggil, REST API perlu didefinisikan terlebih dahulu pada *class controller* UserService.java seperti kode di bawah ini.

```

//delete image
@DELETE("delete_image/{id}")
Call<GalleryResponse> delete_image(@Path("id") String id);

```

Setelah itu dibuat fungsi *deleteImage* dengan variabel masukan berupa id dari *image* itu sendiri. Fungsi *deleteImage* ini kemudian dapat dipanggil pada *onClickListener button delete*.

```

public void deleteImage(final String id) {
    Call<GalleryResponse> galleryResponseCall =
    RetrofitClient.getService().delete_image(id);
    galleryResponseCall.enqueue(new
    Callback<GalleryResponse>() {
        @RequiresApi(api =
        Build.VERSION_CODES.HONEYCOMB)
        @Override
        public void onResponse(Call<GalleryResponse>
        call, Response<GalleryResponse> response) {

            if (response.isSuccessful()) {
                finish();
            } else {
                Toast.makeText(singleImageView.this,
                "gagal menghapus foto", Toast.LENGTH_LONG).show();
            }
        }

        @Override
        public void onFailure(Call<GalleryResponse>
        call, Throwable t) {
            Toast.makeText(singleImageView.this,
            t.getLocalizedMessage(), Toast.LENGTH_LONG).show();
        }
    });
}

```

### 3) Implementasi fungsi input kajian Size luka

Untuk menginput skoring kajian luka digunakan AutoCompleteTextView sehingga user/perawat tidak perlu mengetik ulang hanya memilih opsi saja. Opsi dibuat dalam bentuk list yang kemudian dipakai ke adapter.

```
//list opsi-opsi form size
opsiSize = (AutoCompleteTextView)
findViewById(R.id.editTextSize);
String[] listSize = new String[]{"1= kurang dari
4cm2", "2= antara 4cm2 dan 16cm2", "3= antara 16cm2 dan
36cm2", "4= antara 36cm2 dan 80cm2", "5 = lebih dari
80cm2"};
ArrayAdapter<String> adapterSize = new
ArrayAdapter<>(this, R.layout.list_opsi_agama,
listSize);
opsiSize.setAdapter(adapterSize);
```

### 4) Implementasi fungsi input kajian Edges luka

```
//list opsi-opsi form edges
opsiEdges = (AutoCompleteTextView)
findViewById(R.id.editTextEdges);
String[] listEdges = new String[]{"1 = Tidak terlihat
jelas", "2 = garis besar jelas,\nlekat dengan dasar
luka", "3 = terdefinisikan baik,\n\nidak lekat dasar
luka", "4 = terdefinisikan baik, tidak melekat
pada\nalas, digulung ke bawah, menebal", "5 =
Terdefinisi dengan baik, fibrotik, \nbekas luka atau
hiperkeratotik"};
ArrayAdapter<String> adapterEdges = new
ArrayAdapter<>(this, R.layout.list_opsi_agama,
listEdges);
opsiEdges.setAdapter(adapterEdges);
```

### 5) Implementasi fungsi input kajian Necrotic Type luka

```
//list opsi-opsi form necrotic type
opsiNType = (AutoCompleteTextView)
findViewById(R.id.editTextNType);
String[] listNType = new String[]{"1 = Tidak ada", "2 =
Jaringan berwarna putih/abu-abu \ntidak dapat hidup,
dan atau \nberupa slough yang tidak melekat", "3 =
Slough kuning longgar", "4 = Adherent/menempel, soft,
\nterdapat black eschar", "5 = Firmly adherent/sangat
menempel, \nhard, terdapat black eschar"};
ArrayAdapter<String> adapterNType = new
ArrayAdapter<>(this, R.layout.list_opsi_agama,
listNType);
opsiNType.setAdapter(adapterNType);
```

6) Implementasi fungsi input kajian Necrotic Amount luka

```
//List opsi-opsi form necrotic amount
opsiNAmount = (AutoCompleteTextView)
findViewById(R.id.editTextNAmount);
String[] listNAmount = new String[]{"1 = Tidak ada", "2
= kurang dari 25%\nof wound bed covered", "3 = 25%
sampai 50%\nof wound covered", "4 = 50% sampai 75%\nof
wound covered", "5 = 75% sampai 100%\nof wound
covered"};
ArrayAdapter<String> adapterNAmount = new
ArrayAdapter<>(this, R.layout.list_opsi_agama,
listNAmount);
opsiNAmount.setAdapter(adapterNAmount);
```

7) Implementasi fungsi input kajian Skin Color Surrounding Wound luka

```
//List opsi-opsi form skin color surrounding wound
opsiSkinColor = (AutoCompleteTextView)
findViewById(R.id.editTextSkinColor);
String[] listSkinColor = new String[]{"1 = Pink atau
normal", "2 = Merah terang,\n pucat jika disentuh", "3
= Putih atau abu-abu pucat\natau hipopigmentasi", "4 =
Merah gelap atau ungu,\ntidak pucat jika disentuh", "5
= Hitam atau hyperpigmented"};
ArrayAdapter<String> adapterSkinColor = new
ArrayAdapter<>(this, R.layout.list_opsi_agama,
listSkinColor);
opsiSkinColor.setAdapter(adapterSkinColor);
```

8) Implementasi fungsi input kajian Granulation Tissue luka

```
//List opsi-opsi form granulation tissue
opsiGranulation = (AutoCompleteTextView)
findViewById(R.id.editTextGranulation);
String[] listGranulation = new String[]{"1 = Skin
intact/utuh atau\nberjenis partial thickness wound", "2
= Cerah, merah daging;\n75% sampai 100% luka
terisi\n&/atau jaringan tumbuh berlebih", "3 = Cerah,
merah daging;\n25% sampai 75% wound filled", "4 =
Pink, &/atau kusam, merah kehitaman\n&/or kurang dari
25% wound filled", "5 = Tidak ada jaringan granulasi"};
ArrayAdapter<String> adapterGranulation = new
ArrayAdapter<>(this, R.layout.list_opsi_agama,
listGranulation);
opsiGranulation.setAdapter(adapterGranulation);
```

9) Implementasi fungsi input kajian Granulation Tissue luka

```
//List opsi-opsi form epithelization tissue
opsiEpithelization = (AutoCompleteTextView)
```

```

findViewById(R.id.editTextEpithelization);
String[] listEpithelization = new String[]{"1 = 100% wound covered,\npermukaan utuh", "2 = 75% sampai 100% wound covered\n&/atau jaringan epithelial meluas\nlebih dari 0.5cm into wound bed", "3 = 50% sampai 75% wound covered\n&/atau jaringan epitel meluas kurang dari\n0.5cm pada permukaan luka", "4 = 25% sampai 50% wound covered", "5 = kurang dari 25% wound covered"};
ArrayAdapter<String> adapterEpithelization = new ArrayAdapter<>(this, R.layout.list_opsi_agama, listEpithelization);
opsiEpithelization.setAdapter(adapterEpithelization);

```

10) Implementasi fungsi input kajian Epithelization luka

```

//List opsi-opsi form epithelization tissue
opsiEpithelization = (AutoCompleteTextView)
findViewById(R.id.editTextEpithelization);
String[] listEpithelization = new String[]{"1 = 100% wound covered,\npermukaan utuh", "2 = 75% sampai 100% wound covered\n&/atau jaringan epithelial meluas\nlebih dari 0.5cm into wound bed", "3 = 50% sampai 75% wound covered\n&/atau jaringan epitel meluas kurang dari\n0.5cm pada permukaan luka", "4 = 25% sampai 50% wound covered", "5 = kurang dari 25% wound covered"};
ArrayAdapter<String> adapterEpithelization = new ArrayAdapter<>(this, R.layout.list_opsi_agama, listEpithelization);
opsiEpithelization.setAdapter(adapterEpithelization);

```

11) Menyimpan data kajian yang telah di-input ke server

Setelah fungsi input menggunakan AutoCompleteTextView telah diimplementasikan, maka data akan diambil dengan menggunakan getText() yang kemudian dikirim ke server menggunakan Retrofit.

```

Bundle extras = getIntent().getExtras();
String id_nurse = id_perawat;
String id_pasien = NRM;

String size = opsiSize.getText().toString();
String edges = opsiEdges.getText().toString();
String NType = opsiNType.getText().toString();
String NAmount = opsiNAmount.getText().toString();
String skincolor = opsiSkinColor.getText().toString();
String granulation =
opsiGranulation.getText().toString();
String epithelization =
opsiEpithelization.getText().toString();
Call<dataKajianResponse> call =
RetrofitClient.getService().tambahKajian(id_pasien,id_perawat,size,edges,NType,NAmount,skincolor,granulation,epithelization,rawID,tepiID, diameterID);
call.enqueue(new Callback<dataKajianResponse>() {
    @Override
    public void onResponse(Call<dataKajianResponse>

```

```

call, Response<dataKajianResponse> response) {
    if(response.isSuccessful()){
        //Login start main activity
        Intent i = new
Intent(getApplicationContext(),
detailPasienActivity.class);
        i.putExtra("NRM", NRM);
        anotasiTepi.tepiAct.finish();
        anotasiDiameter.dxAct.finish();
        anotasiDiameterY.dyAct.finish();
        startActivity(i);
        finish();

    }else {

        Toast.makeText(getApplicationContext(),"gagal menambah
kajian baru",
                Toast.LENGTH_SHORT).show();
    }
}

@Override
public void onFailure(Call<dataKajianResponse>
call, Throwable t) {
    call.cancel();
    Toast.makeText(getApplicationContext(), "input
data kajian error", Toast.LENGTH_SHORT).show();
}
});

```

REST API yang digunakan sebelumnya sudah didefinisikan pada *class controller* UserService dengan nama tambahKajian.

```

@FormUrlEncoded
@POST("insert_kajian")
Call<dataKajianResponse>
tambahKajian(@Field("id_pasien") String id_pasien,
@Field("id_perawat") String id_perawat,
@Field("size") String size, @Field("edges") String edges,
@Field("necrotic_type") String necrotic_type,
@Field("necrotic_amount") String necrotic_amount,
@Field("skincolor_surround") String skincolor_surround,
@Field("granulation") String granulation,
@Field("epithelization") String epithelization,
@Field("raw_photo_id") String raw_photo_id,
@Field("tepi_image_id") String tepi_image_id,
@Field("diameter_image_id") String diameter_image_id);

```

12) Membuat mockup tampilan anotasi diameter luka



Gambar 4.26 Mockup Anotasi Diameter Luka

Gambar 4.26 merupakan mockup dari halaman anotasi diameter luka. Pada halaman anotasi diameter, user bisa melakukan pengaturan lebar *stroke*, *undo*, dan *save*.

13) *Source code*

- Android: <https://github.com/apraira/Chronic-Wound-Dataset/tree/sprint-5>
- Web Service: tidak ada penambahan REST pada *sprint-5*

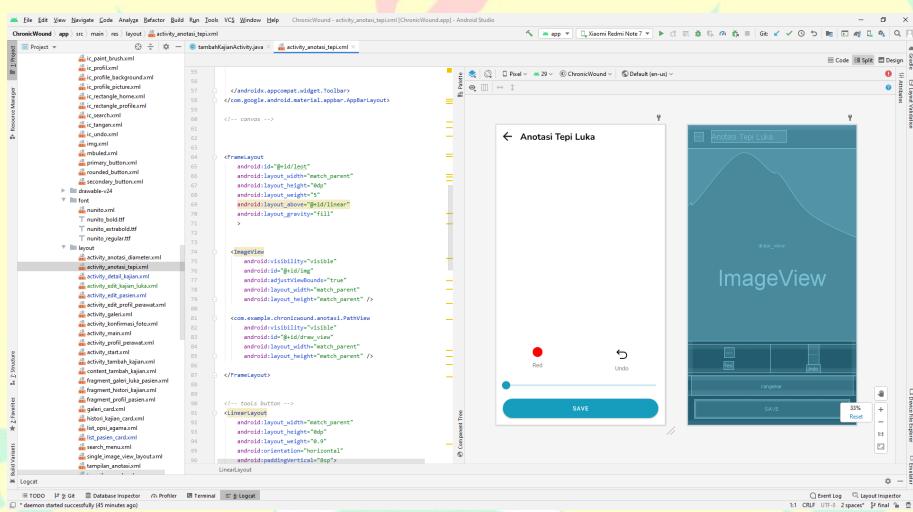
f. *Sprint-6*

Tabel 4.9 Sprint-6 Backlog

No.	User Story	Task	Status
1	Menganotasi tepi luka dan menganotasi orientasi luka	Membuat layout anotasi tepi luka pada XML	Selesai
		Membuat layout anotasi diameter luka pada XML	Selesai
		Membuat flowchart anotasi tepi luka dan anotasi orientasi luka	Next Sprint

No.	User Story	Task	Status
		Implementasi fungsi anotasi tepi luka	Next Sprint
		Implementasi fungsi anotasi diameter luka	Next Sprint
		Menyimpan gambar hasil anotasi tepi luka ke storage internal & server	Next Sprint
		Menyimpan gambar hasil anotasi diameter luka ke storage internal & server	Next Sprint

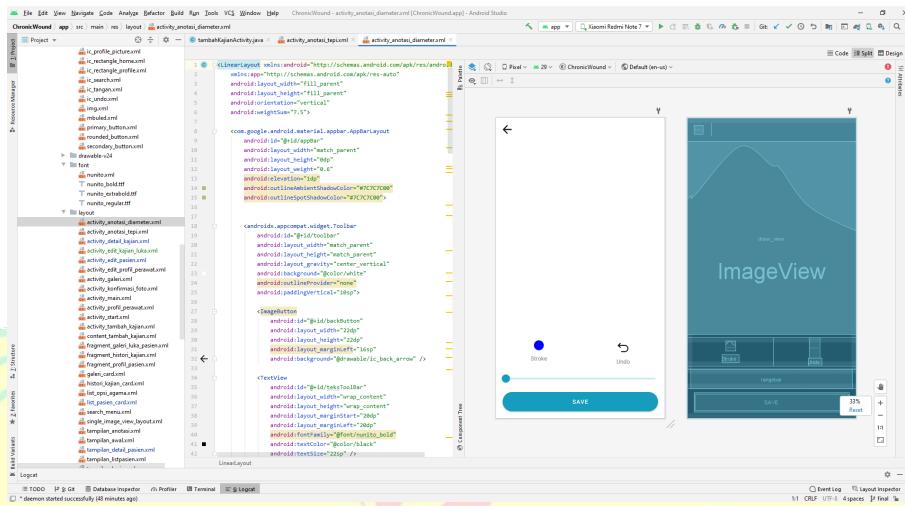
### 1) Membuat layout anotasi tepi luka pada XML



Gambar 4.27 Implementasi Layout Anotasi Tepi Luka

Gambar 4.27 merupakan implementasi dari mockup halaman anotasi tepi luka pada android. Agar anotasi dapat berada di atas foto luka, digunakan FrameLayout yang berisikan ImageView dan PathView.

## 2) Membuat layout anotasi diameter luka pada XML



Gambar 4.28 Implementasi Layout Anotasi Tepi Luka

Gambar 4.28 merupakan implementasi dari mockup halaman anotasi diameter luka pada android. Tampilan dibuat sama dengan anotasi tepi agar *user*/perawat familiar saat menggunakannya.

## 3) Source code

- Android: <https://github.com/apraira/Chronic-Wound-Dataset/tree/sprint-6>
- Web Service: tidak ada penambahan REST pada *sprint-6*

Karena waktu penggeraan tidak cukup, maka task dengan status *Next Sprint* akan dikerjakan pada *sprint* selanjutnya yaitu *sprint-7*.

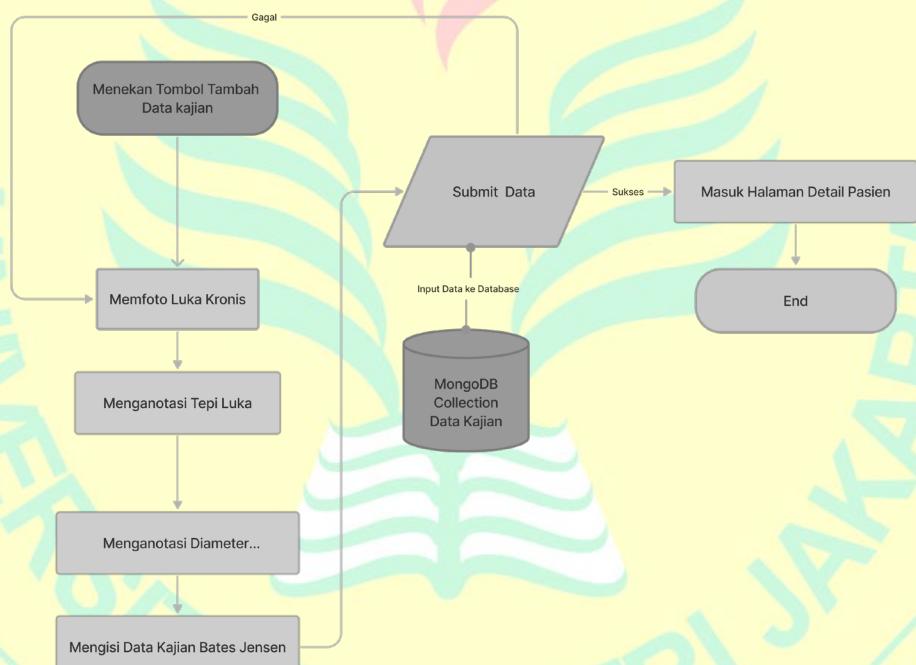
## g. Sprint-7

Tabel 4.10 Sprint-7 Backlog

No.	User Story	Task	Status
1	Menganotasi tepi luka dan menganotasi orientasi luka	Membuat flowchart anotasi tepi luka dan anotasi diameter luka	Selesai
		Implementasi fungsi anotasi tepi luka	Selesai
		Implementasi fungsi anotasi diameter luka	Selesai
		Menyimpan gambar hasil anotasi tepi luka ke storage internal & server	Next Sprint

No.	User Story	Task	Status
		Menyimpan gambar hasil anotasi orientasi luka ke storage internal & server	Next Sprint
2	Mengarsir warna luka	Membuat layout arsir warna luka pada XML	Next Sprint
		Implementasi fungsi arsir warna luka	Next Sprint
		Menyimpan gambar hasil arsir warna luka ke storage internal & server	Next Sprint
		Menghapus arsir warna luka ke storage server	Next Sprint

1) Membuat flowchart anotasi tepi luka dan anotasi diameter luka



Gambar 4.29 Flowchart Anotasi Tepi Luka & Diameter Luka

Fitur anotasi tepi luka & diameter luka diakses ketika perawat menambahkan hasil kajian baru. Pada gambar 4.29 ditunjukkan bahwa sebelum *user*/perawat mengisi skoring kajian luka akan terlebih dahulu melakukan anotasi tepi luka dan anotasi diameter.

2) Implementasi fungsi anotasi tepi luka dan diameter luka

Untuk dapat melakukan anotasi tepi, diperlukan *class* khusus untuk mengimplementasikan Canvas dan Paint dari Android. *Class* ini

kemudian diberi nama PathView.java. Class PathView tidak hanya digunakan untuk anotasi tepi luka, namun juga anotasi diameter luka. Class ini akan diimpor pada class utama yang mana jika ingin memanggil fungsi pada class ini dapat menggunakan perintah seperti variabel \_class\_pathview.undo.

```
package com.example.chronicwound.anotasi;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.PorterDuff;
import android.graphics.Region;
import android.os.Handler;
import android.util.AttributeSet;
import android.util.Log;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewConfiguration;
import android.widget.Toast;

import java.util.ArrayList;

public class PathView extends View {

    private static final float TOUCH_TOLERANCE = 4;
    private float mX, mY;
    private Path mPath;
    Region r;

    private Paint mPaint;

    private ArrayList<Stroke> paths = new
ArrayList<>();
    private ArrayList dataPath = new ArrayList();
    private int currentColor;
    private int strokeWidth;
    private Bitmap mBitmap;
    private Canvas mCanvas;
    private Paint mBitmapPaint = new
Paint(Paint.DITHER_FLAG);

    public PathView(Context context) {
        this(context, null);
    }

    public PathView(Context context, AttributeSet
attrs) {
        super(context, attrs);
        mPaint = new Paint();
```

```
mPaint.setAntiAlias(true);
mPaint.setDither(true);
mPaint.setColor(Color.BLACK);
mPaint.setStyle(Paint.Style.STROKE);
mPaint.setStrokeJoin(Paint.Join.ROUND);
mPaint.setStrokeCap(Paint.Cap.ROUND);

mPaint.setAlpha(0xff);

}

public void init(int height, int width) {

    mBitmap = Bitmap.createBitmap(width, height,
Bitmap.Config.ARGB_8888);
    mCanvas = new Canvas(mBitmap);

    currentColor = Color.RED;

    strokeWidth = 10;
}

public void setColor(int color) {
    currentColor = color;
}

public void erase(){

}

public ArrayList getPathList(){
    return dataPath;
}

public void setStrokeWidth(int width) {
    strokeWidth = width;
}

public void undo() {
    if (paths.size() != 0) {
        paths.remove(paths.size() - 1);
        dataPath.clear();
        invalidate();
    }
}
public Bitmap save() {
    return mBitmap;
}

public void changeHW(int h, int w){
    mBitmap = Bitmap.createBitmap(w, h,
Bitmap.Config.ARGB_8888);
    mCanvas = new Canvas(mBitmap);

    currentColor = Color.BLACK;
```

```
        strokeWidth = 20;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.save();

        int backgroundColor = Color.WHITE;
        mCanvas.drawColor(Color.TRANSPARENT,
PorterDuff.Mode.CLEAR);

        for (Stroke fp : paths) {
            mPaint.setColor(fp.color);
            mPaint.setStrokeWidth(fp.strokeWidth);
            mCanvas.drawPath(fp.path, mPaint);
        }
        canvas.drawBitmap(mBitmap, 0, 0, mBitmapPaint);
        canvas.restore();
    }

    private void touchStart(float x, float y) {
        mPath = new Path();

        Stroke fp = new Stroke(currentColor,
strokeWidth, mPath);
        paths.add(fp);

        mPath.reset();

        mPath.moveTo(x, y);
        dataPath.add("M"+ String.valueOf(x) +"," +
String.valueOf(y));

        mX = x;
        mY = y;
    }

    private void touchMove(float x, float y) {
        float dx = Math.abs(x - mX);
        float dy = Math.abs(y - mY);

        mPath.lineTo(x, y);
        mX = x;
        mY = y;
        dataPath.add("L"+ String.valueOf(x) +"," +
String.valueOf(y));
    }

    private void touchUp() {
        mPath.lineTo(mX, mY);
    }

    Boolean isLongPress = false;
```

```
    final GestureDetector gestureDetector = new
GestureDetector(new
GestureDetector.SimpleOnGestureListener() {
    public void onLongPress(MotionEvent e) {
        mPath.lineTo(mX, mY);
        dataPath.add("z");
        mPath.close();
        System.out.println("paths:" + dataPath);
    }
});

@Override
public boolean onTouchEvent(MotionEvent event) {
    float x = event.getX();
    float y = event.getY();

    gestureDetector.onTouchEvent(event);

    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            Boolean ans = paths.isEmpty();

            if (ans == true){
                touchStart(x, y);
            }
            else {
                touchMove(x,y);
                System.out.println("paths:" +
dataPath);
            }

            invalidate();
            break;

        case MotionEvent.ACTION_UP:
            touchUp();
            invalidate();
            break;
    }
    return true;
}
}
```

3) *Source code:*

- Android: <https://github.com/apraira/Chronic-Wound-Dataset/tree/sprint-7>
- Web Service: tidak ada penambahan REST pada *sprint-7*  
Dikarenakan waktu pengerjaan tidak cukup, maka *task* dengan

status *Next Sprint* akan dikerjakan pada *sprint* selanjutnya yaitu *sprint-8*.

#### **h. *Sprint-8***

Untuk *sprint-8* tidak ada *progress* dikarenakan alasan tertentu.

#### **i. *Sprint-9***

Tabel 4.11 Sprint-9 Backlog

No.	User Story	Task	Status
1	Menganotasi tepi luka dan menganotasi orientasi luka	Menyimpan gambar hasil anotasi tepi luka ke server	Selesai
		Menyimpan gambar hasil anotasi diameter luka ke server	Selesai
2	Mengarsir warna luka	Membuat layout arsir warna luka pada XML	Selesai
		Implementasi fungsi arsir warna luka	Selesai
		Menyimpan gambar hasil arsir warna luka ke server	Selesai
3	Galeri luka kronis semua pasien	Membuat layout Galeri luka kronis semua pasien pada XML	Selesai
		Membuat REST untuk menerima semua foto pada database image	Selesai
		Menampilkan semua foto luka pada database di layout Galeri luka kronis	Selesai
4	Unduh dataset luka milik user tersebut	Membuat REST untuk unduh semua foto luka milik satu pasien	Selesai
5	Melihat histori kajian pasien dan status luka pasien	Membuat Mockup UI tampilan histori kajian pasien	Selesai
		Membuat Mockup UI detail / status kajian pasien	Selesai
		Implementasi Mockup tampilan histori kajian pasien pada XML	Selesai
		Implementasi Mockup tampilan detail/status kajian pasien pada XML	Selesai
		Implementasi fungsi untuk pemanggilan REST API data kajian luka pada BackEnd	Selesai
		Implementasi fungsi untuk menampilkan histori kajian pasien pada Front-End	Selesai

No.	User Story	Task	Status
		Implementasi fungsi untuk menampilkan detail kajian pasien pada Front-End	Selesai
6	Melihat detail profil akun perawat	Membuat Mockup UI detail profil perawat	Selesai
		Implementasi Mockup UI detail profil perawat pada XML	Selesai
		Implementasi fungsi untuk pemanggilan REST API data perawat	Selesai
		Implementasi fungsi untuk menampilkan data perawat pada Front-End	Selesai
7	Keluar dari akun	Implementasi fungsi log-out	Selesai
8	Mencarit dan melihat log activity user	Membuat database untuk log activity	Selesai
		Membuat routing table untuk log activity	Selesai
		Membuat REST API untuk mencatat log activity user	Selesai
		Membuat REST API untuk melihat log activity semua user	Selesai
		Membuat REST API untuk melihat log activity 1 user berdasarkan id user	Selesai
		Membuat REST API untuk menghapus log activity 1 user berdasarkan id user	Selesai
		Implementasi fungsi API untuk mencatat log user pada Android	Selesai

1) Menyimpan gambar hasil anotasi tepi luka ke server

Hasil dari anotasi tepi luka akan disimpan dalam format JPG, PNG, dan SVG. Gambar format JPG diperoleh dengan mengonversi FrameLayout ke dalam bentuk Bitmap yang kemudian diubah menjadi file. Format PNG diperoleh dengan hanya mengonversi bitmap pada Canvas dengan background transparan. Sedangkan untuk menyimpan hasil anotasi ke dalam bentuk SVG, perlu dibuat fungsi pada Python untuk mengubah path anotasi tepi.

```
#upload gambar svg
@bp.route('/upload_svg', methods =['POST'])
```

```
def post_svg():

    #next save the file

    paths= request.form['paths']
    print(paths)
    id = str(uuid.uuid4().hex)
    id_pasien = request.form['id_pasien']

    try:
        output_str = paths.replace(' ', ' ')
        '.replace('[', '').replace(']', '').replace(","," ")
        canvas = '<svg width="1080" height="1441"
viewBox="0 0 1080 1441" fill="none"
xmlns="http://www.w3.org/2000/svg">'
        path = '<path d="' + output_str + '"'
        stroke="black" stroke-width="10"/>'
        close = '</svg>'
        svg = canvas + path + close

        path = os.path.join(current_app.instance_path,
current_app.config['UPLOAD_DIR'])
        try:
            os.makedirs(path)
        except OSError:
            pass

        date =
str(datetime.datetime.now().replace(microsecond=0)).replace("-","");
.replace(" ","_").replace(":", "")

        filename = id_pasien + "_tepi_" + date + ".svg"
        filepath = os.path.join(path, filename)

        svg_file = open(filepath, "w")

        #write string to file
        svg_file.write(svg)

        #close file
        svg_file.close()

        data = {"_id": id,
                "id_pasien": id_pasien,
```

```

        "id_perawat":  

request.form['id_perawat'],  

        "filename":filename,  

        "filepath":path,  

        "type":"Vector",  

        "category":request.form['category']  

,  

        "created_at" :  

time.strftime("%d/%m/%Y %H:%M:%S"),  

        "updated_at" :  

time.strftime("%d/%m/%Y %H:%M:%S"),  

        }  

insert_image(data)  

helper.update_image_user(id_pasien, id)  
  

print(filepath)  

current_app.logger.debug(filepath);  
  

return Response(response =  

json.dumps({"message" : "true"}),  

mimetype="application/json", status=200)  
  

except Exception as ex:  

    print (ex)  

    return Response(response =  

json.dumps({"message" : "error encountered"}),  

mimetype="application/json", status=500)

```

Setelah semua format gambar terkonversi maka akan diunggah menggunakan REST API dengan routing <http://jft.web.id/woundapi/upload> dan [http://jft.web.id/woundapi/upload\\_svg](http://jft.web.id/woundapi/upload_svg) menggunakan Retrofit.

## 2) Menyimpan gambar hasil anotasi diameter luka ke server

Proses penyimpanan anotasi diameter luka sama seperti penyimpanan anotasi tepi luka ke server. Namun, kode konversi ke dalam format SVG-nya berbeda, karena membutuhkan tiga path diantaranya path tepi anotasi, path anotasi diameter koordinat X, dan path anotasi koordinat Y.

```

#upload gambar svg
@bp.route('/upload_svg_3', methods =['POST'])
def post_svg_3():

    #next save the file

    paths= request.form['paths']
    paths2 = request.form ['paths2']
    paths3 = request.form['paths3']
    print(paths)
    id = str(uuid.uuid4().hex)
    id_pasien = request.form['id_pasien']

    try:

        paths = paths.replace(' ', ' ').replace('[',
        '').replace(']', '').replace(","," ")
        paths2 = paths2.replace(' ', ' ').replace('[',
        '').replace(']', '').replace(","," ")
        paths3 = paths3.replace(' ', ' ').replace('[',
        '').replace(']', '').replace(","," ")
        canvas ='<svg width="1080" height="1441"
viewBox="0 0 1080 1441" fill="none"
xmlns="http://www.w3.org/2000/svg">'
        path1 ='<path d="'+ paths + '" stroke="black"
stroke-width="10"/>'
        path2 ='<path d="'+ paths2 + '" stroke="black"
stroke-width="10"/>'
        path3 ='<path d="'+ paths3 + '" stroke="black"
stroke-width="10"/>'

        close = '</svg>'

        svg = canvas + path1 + path2 + path3 + close

        path = os.path.join(current_app.instance_path,
current_app.config['UPLOAD_DIR'])
        try:
            os.makedirs(path)
        except OSError:
            pass

        date =
str(datetime.datetime.now().replace(microsecond=0)).rep
lace("-","").replace(" ","_").replace(':', "")
```

```
filename = id_pasien + "_diameter_" + date +
".svg"
filepath = os.path.join(path, filename)

svg_file = open(filepath, "w")

#write string to file
svg_file.write(svg)

#close file
svg_file.close()

data = {"_id": id,
        "id_pasien": id_pasien,
        "id_perawat":
request.form['id_perawat'],
        "filename":filename,
        "filepath":path,
        "type":"Vector",
        "category":request.form['category']
,
        "created_at" :
time.strftime("%d/%m/%Y %H:%M:%S"),
        "updated_at" :
time.strftime("%d/%m/%Y %H:%M:%S"),
        }
insert_image(data)
helper.update_image_user(id_pasien, id)

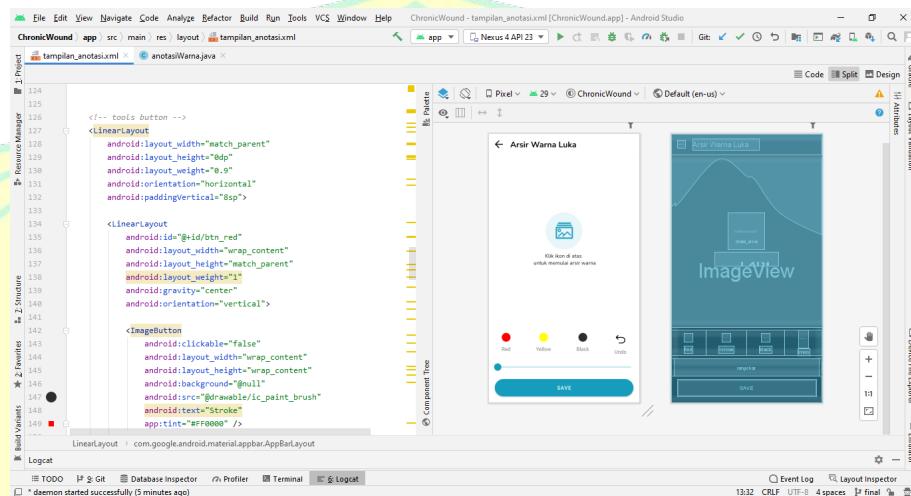
print(filepath)
current_app.logger.debug(filepath);

return Response(response =
json.dumps({"message" : "true"}),
mimetype="application/json", status=200)

except Exception as ex:
    print (ex)
    return Response(response =
json.dumps({"message" : "error encountered"}),
mimetype="application/json", status=500)
```

### 3) Membuat layout arsir warna luka pada XML

Berbeda dengan anotasi tepi dan anotasi diameter luka, untuk arsir warna luka dibutuhkan tiga warna yaitu Merah, Kuning, dan Hitam. Pemilihan warna sesuai dengan metode RYB dimana umum digunakan untuk mengklasifikasi warna luka.



Gambar 4.30 Implementasi Mockup Arsir Warna Luka

### 4) Implementasi fungsi arsir warna luka

Fungsi arsir warna luka dapat digunakan setelah *user* memilih foto melalui galeri dan juga dapat digunakan pada laman detail *image*. Agar dapat mengarsir pada canvas, digunakan *class* custom yang kemudian dinamai *DrawView*. Pada *class* tersebut terdapat fungsi-fungsi seperti merubah warna *stroke*, *undo*, mengubah lebar *stroke*, dan *save*. *Class* *DrawView* kemudian dipanggil di *class* *AnotasiWarna*.

```
public class DrawView extends View {

    private static final float TOUCH_TOLERANCE = 4;
    private float mX, mY;
    private Path mPath;

    private Paint mPaint;

    private ArrayList<Stroke> paths = new ArrayList<>();
    private int currentColor;
    private int strokeWidth;
    private Bitmap mBitmap;
    private Canvas mCanvas;
    private Paint mBitmapPaint = new
    Paint(Paint.DITHER_FLAG);
```

```
public DrawView(Context context) {
    this(context, null);
}

public DrawView(Context context, AttributeSet attrs)
{
    super(context, attrs);
    mPaint = new Paint();

    mPaint.setAntiAlias(true);
    mPaint.setDither(true);
    mPaint.setColor(Color.BLACK);
    mPaint.setStyle(Paint.Style.STROKE);
    mPaint.setStrokeJoin(Paint.Join.ROUND);
    mPaint.setStrokeCap(Paint.Cap.ROUND);

    mPaint.setAlpha(0xff);
}

public void init(int height, int width) {

    mBitmap = Bitmap.createBitmap(width, height,
        Bitmap.Config.ARGB_8888);
    mCanvas = new Canvas(mBitmap);

    currentColor = Color.BLACK;

    strokeWidth = 10;
}

public void setColor(int color) {
    currentColor = color;
}

public void erase(){

}

public void setStrokeWidth(int width) {
    strokeWidth = width;
}

public void undo() {

    if (paths.size() != 0) {
        paths.remove(paths.size() - 1);
        invalidate();
    }
}

public Bitmap save() {
```

```
        return mBitmap;
    }

    public void changeHW(int h, int w){
        mBitmap = Bitmap.createBitmap(w, h,
Bitmap.Config.ARGB_8888);
        mCanvas = new Canvas(mBitmap);

        currentColor = Color.BLACK;

        strokeWidth = 20;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.save();

        int backgroundColor = Color.WHITE;
        mCanvas.drawColor(Color.TRANSPARENT,
PorterDuff.Mode.CLEAR);

        for (Stroke fp : paths) {
            mPaint.setColor(fp.color);
            mPaint.setStrokeWidth(fp.strokeWidth);
            mCanvas.drawPath(fp.path, mPaint);
        }
        canvas.drawBitmap(mBitmap, 0, 0, mBitmapPaint);
        canvas.restore();
    }

    private void touchStart(float x, float y) {
        mPath = new Path();
        Stroke fp = new Stroke(currentColor, strokeWidth,
mPath);
        paths.add(fp);

        mPath.reset();

        mPath.moveTo(x, y);

        mX = x;
        mY = y;
    }

    private void touchMove(float x, float y) {
        float dx = Math.abs(x - mX);
        float dy = Math.abs(y - mY);

        if (dx >= TOUCH_TOLERANCE || dy >=
TOUCH_TOLERANCE) {
            mPath.quadTo(mX, mY, (x + mX) / 2, (y + mY) /
2);
            mX = x;
            mY = y;
        }
    }
}
```

```

    }

    private void touchUp() {
        mPath.lineTo(mX, mY);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        float x = event.getX();
        float y = event.getY();

        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                touchStart(x, y);
                invalidate();
                break;
            case MotionEvent.ACTION_MOVE:
                touchMove(x, y);
                invalidate();
                break;
            case MotionEvent.ACTION_UP:
                touchUp();
                invalidate();
                break;
        }
        return true;
    }
}

```

##### 5) Menyimpan gambar hasil arsir warna luka ke server

Penyimpanan hasil arsir warna luka ke server menggunakan REST API yang sama dengan penyimpanan gambar pada umumnya. Pertama REST API terlebih dahulu didefinisikan pada *class controller* UserService.java kemudian dipanggil pada class dimana fitur akan dibuat.

```

@Multipart
@POST("upload")
Call<UploadRequest> uploadImage(@Part MultipartBody.Part
image, @Part("id") RequestBody id,
                                    @Part("id_pasien")
RequestBody id_pasien,
                                    @Part("id_perawat")
RequestBody id_perawat,@Part("type") RequestBody type,
                                    @Part("category")
RequestBody category);

```

Arsir warna luka akan disimpan dengan format JPG dan PNG, dimana format JPG didapatkan dengan mengonversi keseluruhan Bitmap dari FrameLayout, sedangkan format PNG didapatkan dengan mengonversi Bitmap pada Canvas dengan *background* transparan.

```
// upload image
public void uploadImage(final MultipartBody.Part image,
final RequestBody id, final RequestBody id_pasien,
final RequestBody id_perawat, final RequestBody type,
final RequestBody category){
    Call<UploadRequest> uploadRequestCall =
RetrofitClient.getService().uploadImage(image, id,
id_pasien, id_perawat, type, category);
    uploadRequestCall.enqueue(new
Callback<UploadRequest>() {

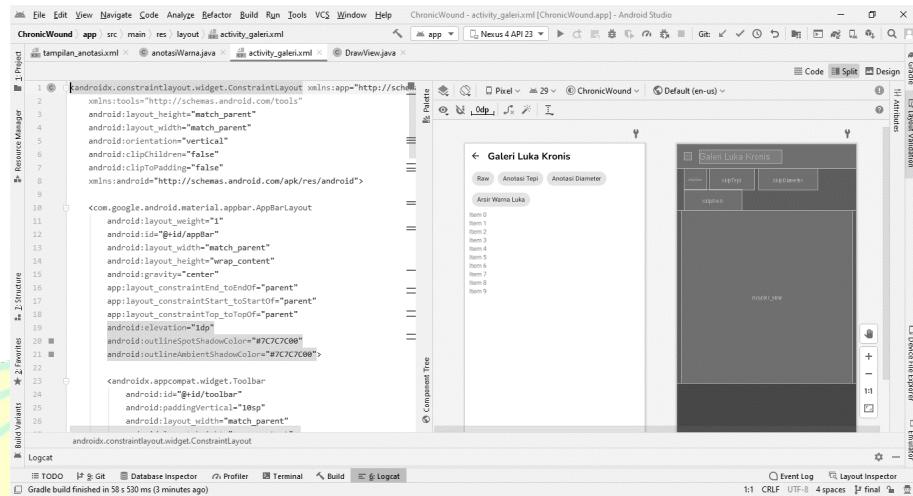
        @Override
        public void onResponse(Call<UploadRequest>
call, Response<UploadRequest> response) {

            if(response.isSuccessful()){
                Toast.makeText(getApplicationContext(),
"Image uploaded to server", Toast.LENGTH_LONG).show();
                InsertLog(id_nurse, "Berhasil upload
gambar ke server");
                Intent intent = new
Intent(getApplicationContext(), MainActivity.class);
                startActivity(intent);
                finish();
            }else {
                Toast.makeText(getApplicationContext(),
"gagal upload image" + category + type,
Toast.LENGTH_LONG).show();
            }
        }

        @Override
        public void onFailure(Call<UploadRequest> call,
Throwable t) {
            Toast.makeText(getApplicationContext(),
t.getLocalizedMessage(), Toast.LENGTH_LONG).show();
        }
    });
}

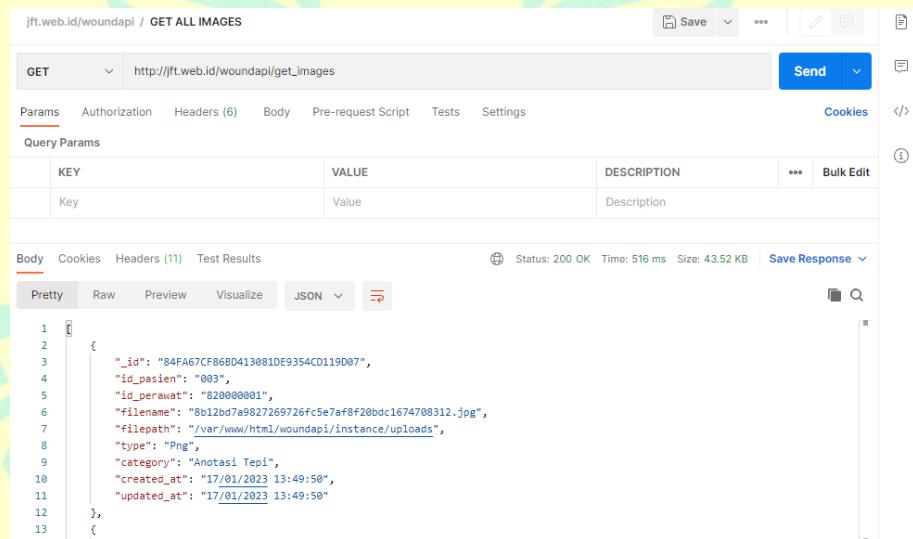
}
```

## 6) Membuat *layout* Galeri luka kronis semua pasien pada XML



Gambar 4.31 Layout Galeri Luka Pada XML

## 7) Membuat REST untuk menerima semua foto pada database image



Gambar 4.32 REST API Get All Image

Dari gambar di atas, ditampilkan bahwa REST API seluruhnya berhasil dibuat karena mengembalikan status 200 (OK).

## 8) Menampilkan semua foto luka pada database di layout Galeri luka kronis

Foto pada halaman Galeri Luka Kronis ditampilkan dengan menggunakan *library* Glide, Retrofit, dan RecyclerView. RecyclerView digunakan untuk menampilkan data serupa secara berulang dan dinamis,

sedangkan Glide digunakan untuk menampilkan gambar berupa URL ke dalam ImageView.

Halaman yang menggunakan RecyclerView, memerlukan *class* adapter yang berperan sebagai jembatan antara komponen tampilan antarmuka dan sumber data.

```
public class AllGalleryAdapter extends
RecyclerView.Adapter<com.example.chronicwound.gallery.AllGalleryAdapter.ImageViewHolder> {
    private AdapterView.OnItemClickListener listener;
    private String KEY_NAME = "NRM"; //nomor registrasi pasien
    private String id_image, filename;
    galerilukaPasien myFragment;

    private ArrayList<GalleryRequest> dataList;

    public AllGalleryAdapter(ArrayList<GalleryRequest> dataList,
GaleriActivity GaleriActivity) {
        this.dataList = dataList;
    }

    // method for filtering our recyclerview items.
    public void filterList(ArrayList<GalleryRequest> filterlist) {
        // below Line is to add our filtered
        // list in our course array list.
        dataList = filterlist;
        // below line is to notify our adapter
        // as change in recycler view data.
        notifyDataSetChanged();
    }

    @Override
    public
com.example.chronicwound.gallery.AllGalleryAdapter.ImageViewHolder
onCreateViewHolder(ViewGroup parent, int viewType) {
    LayoutInflator layoutInflater =
LayoutInflator.from(parent.getContext());
    View view = layoutInflater.inflate(R.layout.galeri_card,
parent, false);
    return new
com.example.chronicwound.gallery.AllGalleryAdapter.ImageViewHolder(vi
ew);
}

    @Override
    public void
onBindViewHolder(com.example.chronicwound.gallery.AllGalleryAdapter.I
mageViewHolder holder, int position) {
        final GalleryRequest imageModel =
(com.example.chronicwound.gallery.GalleryRequest)
dataList.get(position);
        String type = imageModel.getType();

        System.out.println(type + " is loaded by Glide");

        Glide.with(holder.itemView.getContext()).load("https://jft.web.id/wou
ndapi/instance/uploads/" + imageModel.getFilename())
            .centerCrop()
            .thumbnail(0.05f)

        .transition(DrawableTransitionOptions.withCrossFade())
            .into(holder.imageView);
    }
}
```

```

@Override
public int getItemCount() {
    return (dataList != null) ? dataList.size() : 0;
}

public class ImageViewHolder extends RecyclerView.ViewHolder{
    public ImageView imgView;
    public LinearLayout.LayoutParams params;
    public LinearLayout rootView; //the outermost view from your
Layout. Note that it doesn't necessarily have to be a LinearLayout.

    public ImageViewHolder(View itemView) {
        super(itemView);
        imgView = (ImageView)
itemView.findViewById(R.id.iv_photo);
        rootView = itemView.findViewById(R.id.galeriCard);
        params = new LinearLayout.LayoutParams(0, 0);

        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int position = getAdapterPosition();
                String IDImage = dataList.get(position).getID();
                String IDPasien =
dataList.get(position).getId_pasien();
                Context context = v.getContext();
                Intent i = new Intent(context,
singleImageView.class);

                if
(dataList.get(position).getType().toLowerCase().contains("jpg")){
                    i.putExtra("type", "jpg");
                } else{
                    i.putExtra("type", "none");
                }
                i.putExtra("IDPasien", IDPasien);
                i.putExtra(KEY_NAME, IDImage);
                context.startActivity(i);
            }
        });
    }
}

```

Layout yang digunakan di dalam RecycleView didefinisikan pada fungsi ImageViewHolder. Fungsi ImageViewHolder kemudian dipakai pada *class* OnBindViewHolder dimana *class* ini bertugas untuk menampilkan data pada layout yang sebelumnya didefinisikan pada *class* ImageViewHolder. Sumber data dikirimkan dari *class* GaleriActivity di dalam fungsi getAllImages().

```

private void getAllImages() {
    Call<ArrayList<GalleryRequest>> pasienResponseCall =
RetrofitClient.getService().getAllImages();

    // on below line we are calling method to enqueue and calling
    // all the data from array list.
    pasienResponseCall.enqueue(new Callback<ArrayList<GalleryRequest>>() {
        @Override
        public void onResponse(Call<ArrayList<GalleryRequest>> call,
Response<ArrayList<GalleryRequest>> response) {
            // inside on response method we are checking
            // if the response is success or not.
            if (response.isSuccessful()) {

```

```

        // below line is to add our data from api to our array
list.
        imageArrayList = response.body();

        // below line we are running a Loop to add data to our
adapter class.
        for (int i = 0; i < imageArrayList.size(); i++) {
            adapter = new AllGalleryAdapter(imageArrayList,
GaleriActivity.this);

            int numberOfColumns = 2;

            adapter = new
com.example.chronicwound.gallery.AllGalleryAdapter(imageArrayList,
com.example.chronicwound.gallery.GaleriActivity.this);

            RecyclerView.LayoutManager layoutManager = new
GridLayoutManager(com.example.chronicwound.gallery.GaleriActivity.this,
numberOfColumns);

            recyclerView.setLayoutManager(layoutManager);

            recyclerView.setAdapter(adapter);
        }
    }

    @Override
    public void onFailure(Call<ArrayList<GalleryRequest>> call,
Throwable t) {
    // in the method of on failure we are displaying a
    // toast message for fail to get data.
    Toast.makeText(GaleriActivity.this, "Fail to get data",
Toast.LENGTH_SHORT).show();
}
}
}

```

- 9) Membuat REST untuk unduh semua foto luka milik satu pasien

*Routing* yang digunakan adalah /download\_files\_pasien/ dengan metode GET. Setelah menerima *input* berupa ID Pasien, maka akan dijalankan fungsi untuk menerima semua dokumen pada *collection* image dengan ID pasien yang sama. Dari setiap dokumen diambil nama file yang kemudian akan dilakukan pengecekan di folder “upload” sehingga semua file yang dimiliki oleh pasien tersebut dapat digabung menggunakan *library Zipfile*.

```

#download all from id pasien
@bp.route('/download_files_pasien/', methods=["GET"])
def download_pasien():
    id_pasien = request.args.get("nrm")
    filter = {}
    filter["id_pasien"] = id_pasien
    data = {"id_pasien" : id_pasien}
    cek = image_list_by_id(data)
    a = []

```

```
if cek == None:
    return Response(response =
json.dumps({"message" : "not found"}),
mimetype="application/json", status=404)
else:

    for doc in cek:
        a.append(doc)

#for i in a:
#    print(i["filename"])

path = os.path.join(current_app.instance_path,
current_app.config['UPLOAD_DIR']).replace("uploads", "")

#lokasi folder
folderLocation =
os.path.join(current_app.instance_path,
current_app.config['UPLOAD_DIR'])
dir = os.path.join(path, "zip")
namafile = "imagefiles.zip"
fixedfilename = os.path.join(dir, namafile)

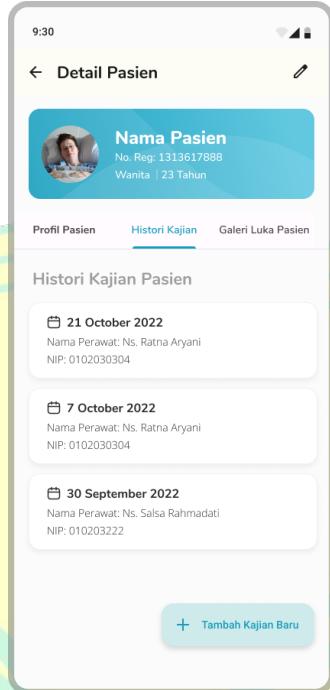
print(fixedfilename)

# zip all the files which are inside in the
folder
zf = zipfile.ZipFile(fixedfilename, "w")
for dirname, subdirs, files in
os.walk(folderLocation):
    zf.write(dirname)
    for filename in files:
        for i in a:
            if filename == i["filename"]:
                print(filename)
                zf.write(os.path.join(dirname,
filename))
zf.close()

dir = os.path.join(path, "zip")

return send_from_directory( dir,
"imagefiles.zip", as_attachment=True)
```

10) Membuat Mockup UI tampilan histori kajian pasien



Gambar 4.33 Mockup Tampilan Detail Pasien Bagian Histori Kajian

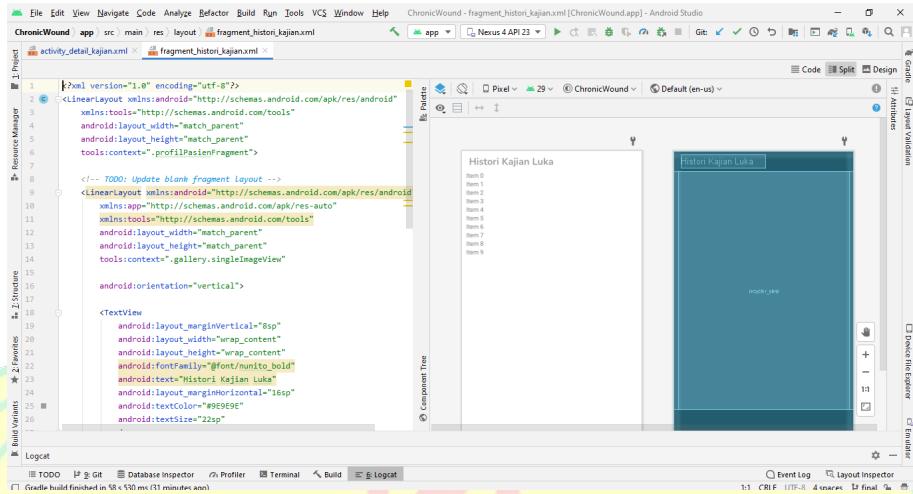
Gambar di atas merupakan tampilan dari halaman histori kajian yang mana di dalamnya dimuat tanggal kajian itu dibuat, perawat yang bertanggung jawab, dan NIP dari perawat tersebut.

11) Membuat Mockup UI detail / status kajian pasien



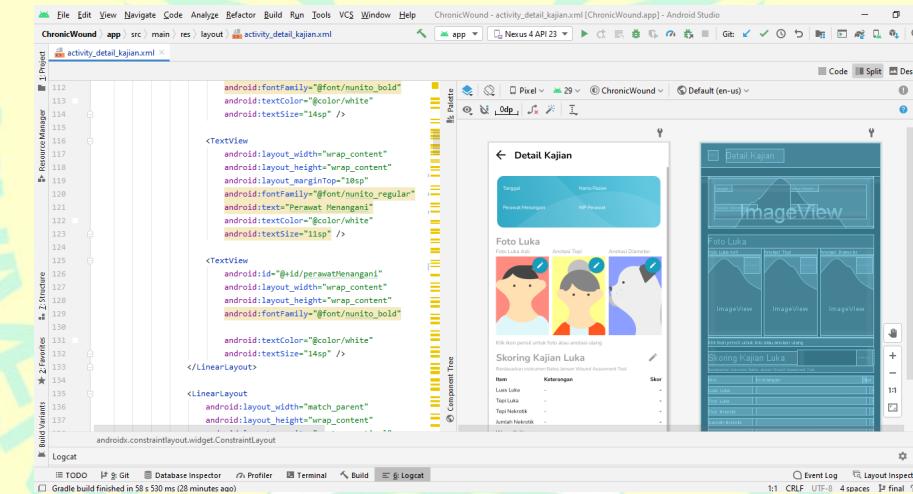
Gambar 4.34 Tampilan Detail Kajian

## 12) Implementasi Mockup tampilan histori kajian pasien pada XML



Gambar 4.35 Tampilan Histori Kajian pada XML

## 13) Implementasi Mockup tampilan detail/status kajian pasien pada XML



Gambar 4.36 Tampilan Detail Kajian pada XML

Pada implementasinya, tampilan mengalami penambahan keterangan foto serta ikon edit anotasi dan foto luka.

## 14) Implementasi fungsi untuk pemanggilan REST API data kajian luka pada BackEnd

Fungsi untuk mendapatkan data kajian luka diimplementasikan dengan memanggil REST API `getHistoriPasien(nrm)` dimana memerlukan input nomor rekam medis pasien yang diinginkan. Sebelum REST API dapat dipanggil, REST API perlu didefinisikan

terlebih dahulu pada *class controller* UserService.java seperti kode di bawah ini.

```
@GET("get_kajian/pasien/{id_pasien}")
Call<ArrayList<KajianResponse>>
getHistoriPasien(@Path("id_pasien") String id);
```

Setelah itu dibuat fungsi getHistoriKajian dengan variabel masukan berupa nomor rekam medis pasien. Fungsi getHistoriKajian juga sekaligus mengirimkan data berupa ArrayList yang kemudian akan ditampilkan di RecyclerView.

```
private void getHistoriKajian(final String NRM) {
    Call<ArrayList<KajianResponse>> KajianResponseCall =
RetrofitClient.getService().getHistoriPasien(NRM);

    KajianResponseCall.enqueue(new
Callback<ArrayList<KajianResponse>>() {
        @Override
        public void
onResponse(Call<ArrayList<KajianResponse>> call,
Response<ArrayList<KajianResponse>> response) {

            if (response.isSuccessful()) {

                kajianArrayList =
response.body();
                Collections.reverse(kajianArrayList);

                for (int i = 0; i < kajianArrayList.size();
i++) {
                    adapter = new
kajianAdapter(kajianArrayList,
com.example.chronicwound.historikajianFragment.this);

                    LinearLayoutManager manager = new
LinearLayoutManager(getContext());

                    recyclerView.setLayoutManager(manager);

                    recyclerView.setAdapter(adapter);
                }
            }
        }

        @Override
        public void onFailure(Call<ArrayList<KajianResponse>>
call, Throwable t) {
            Toast.makeText(getContext(), t.toString(),
Toast.LENGTH_SHORT).show();
            System.out.println("ERROR: " + t.toString());
        }
    });
}
```

- 15) Implementasi fungsi untuk menampilkan detail kajian pasien pada Front-End

Halaman detail kajian akan ditampilkan setelah pengguna klik kartu histori kajian yang dipilih. Untuk menampilkannya, setelah pengguna klik tombol tersebut, maka akan dikirimkan menggunakan Intent Extra dengan value id dari kajian tersebut ke *class* detailKajian. Kemudian nilai dari id kajian tersebut digunakan sebagai parameter pada fungsi *getDetail()*.

```
private void initViews() {
    //umum
    tanggalKajian = (TextView)
findViewById(R.id.tanggalKajian);
    namaPasien = (TextView) findViewById(R.id.namaPasien);
    perawatMenangani = (TextView)
findViewById(R.id.perawatMenangani);
    NIPperawat = (TextView) findViewById(R.id.NIPperawat);

    //ukuran
    panjangX = (TextView) findViewById(R.id.panjangX);
    panjangY = (TextView) findViewById(R.id.panjangY);
    luasLuka = (TextView) findViewById(R.id.LuasLuka);

    //keterangan BWAT
    textKeteranganLuas = (TextView)
findViewById(R.id.textKeteranganLuas);
    textKeteranganTepi = (TextView)
findViewById(R.id.textKeteranganTepi);
    textKeteranganTipeNekrotik = (TextView)
findViewById(R.id.textKeteranganTipeNekrotik);
    textKeteranganJumlahNekrotik = (TextView)
findViewById(R.id.textKeteranganJumlahNekrotik);
    textKeteranganWarnaKulit = (TextView)
findViewById(R.id.textKeteranganWarnaKulit);
    textKeteranganGranulasi = (TextView)
findViewById(R.id.textKeteranganGranulasi);
    textKeteranganEpitelisasi = (TextView)
findViewById(R.id.textKeteranganEpitelisasi);

    //skor BWAT
    textSkorLuas = (TextView)
findViewById(R.id.textSkorLuas);
    textSkorTepi = (TextView)
findViewById(R.id.textSkorTepi);
    textSkorTipeNekrotik = (TextView)
findViewById(R.id.textSkorTipeNekrotik);
    textSkorJumlahNekrotik = (TextView)
findViewById(R.id.textSkorJumlahNekrotik);
    textSkorWarnaKulit = (TextView)
findViewById(R.id.textSkorWarnaKulit);
    textSkorGranulasi = (TextView)
findViewById(R.id.textSkorGranulasi);
    textSkorEpitelisasi = (TextView)
findViewById(R.id.textSkorEpitelisasi);
    textSkorJumlah = (TextView)
findViewById(R.id.textSkorJumlah);
    textSkorJumlah = (TextView)
findViewById(R.id.textSkorJumlah);
```

```
//Image View Luka
    rawImageView = (ImageView)
findViewById(R.id.rawImageView);
    anotasiTepiView = (ImageView)
findViewById(R.id.anotasiTepiView);
    anotasiDiameterView = (ImageView)
findViewById(R.id.anotasiDiameterView);

//Button
buttonDelete = (Button) findViewById(R.id.buttonDelete);
editFoto = (MaterialButton) findViewById(R.id.editFoto);
editTepi = (MaterialButton) findViewById(R.id.editTepi);
editDiameter = (MaterialButton)
findViewById(R.id.editDiameter);
    editSkoring = (MaterialButton)
findViewById(R.id.editSkoring);

}

@Override
protected void onCreate(Bundle savedInstanceState) {
    InsertLog(id_nurse, "Memasuki halaman detail kajian");
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_detail_kajian);

    initViews();

    Intent intent = getIntent();
    id_kajian = intent.getStringExtra("id_kajian");

    getDetail(id_kajian);
}

public void getDetail(final String id) {
    Call<KajianResponse> pasienResponseCall =
RetrofitClient.getService().cariDetailKajian(id);
    pasienResponseCall.enqueue(new Callback<KajianResponse>(){
{
        @RequiresApi(api = Build.VERSION_CODES.GINGERBREAD)
        @Override
        public void onResponse(Call<KajianResponse> call,
Response<KajianResponse> response) {

            if (response.isSuccessful()) {
                //Login start main activity
                Integer jml = 0;
                Integer hasil = 0;

                //umum
                tanggalKajian.setText(response.body().getCreated_at());
                cariPasien(response.body().getId_pasien());
                cariPerawat(response.body().getId_perawat());

                //gambar
                RawImage(response.body().getRaw_photo_id());

                TepiImage(response.body().getTepi_image_id());
                DiameterImage(response.body().getDiameter_image_id());
                RawID = response.body().getRaw_photo_id();
            }
        }
    }
}
```

```
TepiID = response.body().getTepi_image_id();
DiameterID =
response.body().getDiameter_image_id();

//Luas
if( response.body().getSize().isEmpty()){
    System.out.println("empty");
} else{

textKeteranganLuas.setText(response.body().getSize().substrin
g(3).trim());

textSkorLuas.setText(response.body().getSize().substring(0,1)
.toString());
jml +=
Integer.valueOf(response.body().getSize().substring(0, 1));

//Tepi

if( response.body().getEdges().isEmpty()){
    System.out.println("empty");
} else{

textKeteranganTepi.setText(response.body().getEdges().substri
ng(3).trim());

textSkorTepi.setText(response.body().getEdges().substring(0,1)
.toString());
jml +=
Integer.valueOf(response.body().getEdges().substring(0,1));

//Tipe Nekrotik
if(
response.body().getNecrotic_type().isEmpty()){
    System.out.println("empty");
} else{

textKeteranganTipeNekrotik.setText(response.body().getNecroti
c_type().substring(3).trim());

textSkorTipeNekrotik.setText(response.body().getNecrotic_type()
.substring(0,1).toString());
jml+=
Integer.valueOf(response.body().getNecrotic_type().substring(
0,1));
}

//Jumlah Nekrotik
if(
response.body().getNecrotic_amount().isEmpty()){
    System.out.println("empty");
} else{

textKeteranganJumlahNekrotik.setText(response.body().getNecro
tic_amount().substring(3).trim());
```

```
textSkorJumlahNekrotik.setText(response.body().getNecrotic_amount().substring(0,1).toString());
jml+=
Integer.valueOf(response.body().getNecrotic_amount().substring(0,1));
}

//Warna Kulit Keliling Luka
if(
response.body().getSkinColor_surround().isEmpty()){
    System.out.println("empty");
} else{

textKeteranganWarnaKulit.setText(response.body().getSkinColor_surround().substring(3).trim());

textSkorWarnaKulit.setText(response.body().getSkinColor_surround().substring(0,1).toString());
jml+=
Integer.valueOf(response.body().getSkinColor_surround().substring(0,1));
}

//Granulasi
if(
response.body().getGranulation().isEmpty()){
    System.out.println("empty");
} else{

textKeteranganGranulasi.setText(response.body().getGranulation().substring(3).trim());

textSkorGranulasi.setText(response.body().getGranulation().substring(0,1).toString());
jml+=
Integer.valueOf(response.body().getGranulation().substring(0,1));
}

//Epitelisasi
if(
response.body().getEpithelization().isEmpty()){
    System.out.println("empty");
} else{

textKeteranganEpitelisasi.setText(response.body().getEpithelization().substring(3).trim());

textSkorEpitelisasi.setText(response.body().getEpithelization().substring(0,1).toString());
jml+=
Integer.valueOf(response.body().getEpithelization().substring(0,1));
}

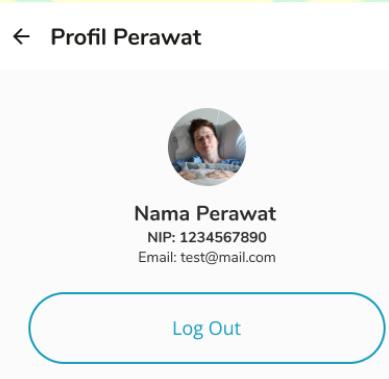
//jumlah skor
textSkorJumlah.setText(jml.toString());
```

```
        } else {
            Toast.makeText(getApplicationContext(),
            "gagal", Toast.LENGTH_LONG).show();
        }

    }

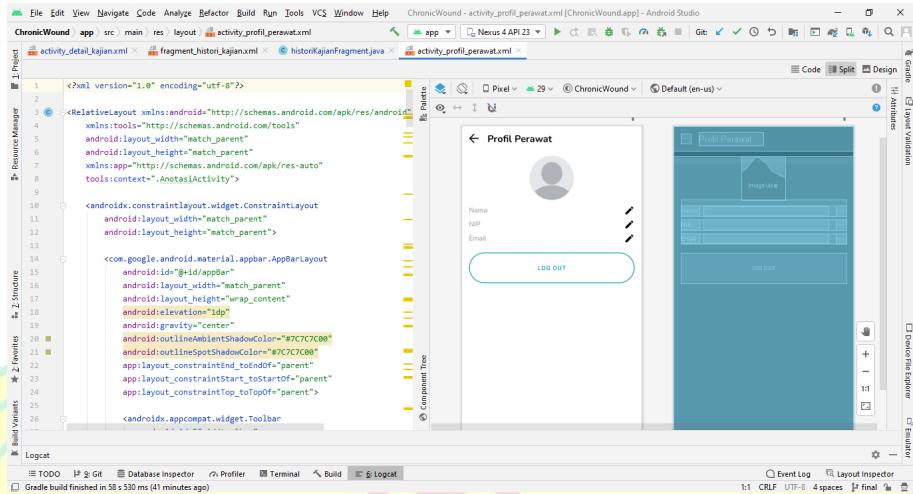
    @Override
    public void onFailure(Call<KajianResponse> call,
    Throwable t) {
        Toast.makeText(getApplicationContext(),
        t.getLocalizedMessage(), Toast.LENGTH_LONG).show();
    }
}
```

16) Membuat Mockup UI detail profil perawat



Gambar 4.37 Tampilan Profil Perawat pada XML

## 17) Implementasi Mockup UI detail profil perawat pada XML



Gambar 4.38 Tampilan Profil Perawat pada XML

Tampilan berbeda dengan *mockup* dikarenakan adanya saran oleh Scrum Master untuk menambahkan fitur edit profil perawat.

## 18) Implementasi fungsi untuk pemanggilan REST API data perawat

Fungsi untuk mendapatkan data perawat diimplementasikan dengan memanggil REST API cariIDPerawat dimana memerlukan input username atau NIP perawat. Sebelum REST API dapat dipanggil, REST API perlu didefinisikan terlebih dahulu pada *class controller* UserService.java seperti kode di bawah ini.

```
@GET("user/{username}")
Call<LoginResponse> cariIDPerawat(@Path("username")
String username);
```

Setelah itu dibuat fungsi cariPerawat dengan variabel masukan berupa NIP perawat.

```
public void cariPerawat(final String username) {
    Call<LoginResponse> loginResponseCall =
RetrofitClient.getService().cariIDPerawat(username);
    loginResponseCall.enqueue(new
Callback<LoginResponse>() {
        @Override
        public void onResponse(Call<LoginResponse>
call, Response<LoginResponse> response) {

            if (response.isSuccessful()) {
                //Login start main activity
                String nama =
```

```

response.body().getName();
String nip =
response.body().getUsername();
Integer id1 = response.body().get_id();
String id = id1.toString();

SharedPreferences preferences =
getSharedPreferences("preferences", MODE_PRIVATE);
SharedPreferences.Editor editor =
preferences.edit();
editor.putString("_id_perawat", id);
editor.commit();

textNamaPerawat.setText(nome);
textNIPPerawat.setText(nip);

textEmailPerawat.setText(response.body().getEmail());

if
(response.body().getProfile_image_url() != null){

Glide.with(getApplicationContext()).load(response.body()
).getProfile_image_url())
.centerCrop()
.thumbnail(0.05f)

.transition(DrawableTransitionOptions.withCrossFade())
.override( 200, 200 )
.into(fotoPerawat);}

} else {
Toast.makeText(getApplicationContext(),
"gagal", Toast.LENGTH_LONG).show();
}

}

@Override
public void onFailure(Call<LoginResponse> call,
Throwable t) {
Toast.makeText(getApplicationContext(),
t.getLocalizedMessage(), Toast.LENGTH_LONG).show();
}
);
}
}

```

19) Implementasi fungsi untuk menampilkan data perawat pada Front-End

Untuk menampilkan data perawat pada TextView digunakan .setText() yang terdapat pada Android sesuai dengan contoh kode di bawah. Sintaks tersebut digunakan pada saat REST API dipanggil.

```
textNamaPerawat.setText(nama);
textNIPPerawat.setText(nip);
```

20) Implementasi fungsi *log-out*

Tombol *log out* yang terdapat pada halaman Profil Perawat dibuat dengan mengganti status “Signed In” pada getSharedPreferences menjadi false. Kemudian pada saat pindah ke tampilan awal, diperlukan Intent.FLAG\_ACTIVITY\_CLEAR\_TOP dan Intent.FLAG\_ACTIVITY\_NEW\_TASK untuk menghentikan aktivitas yang sudah berjalan sebelumnya.

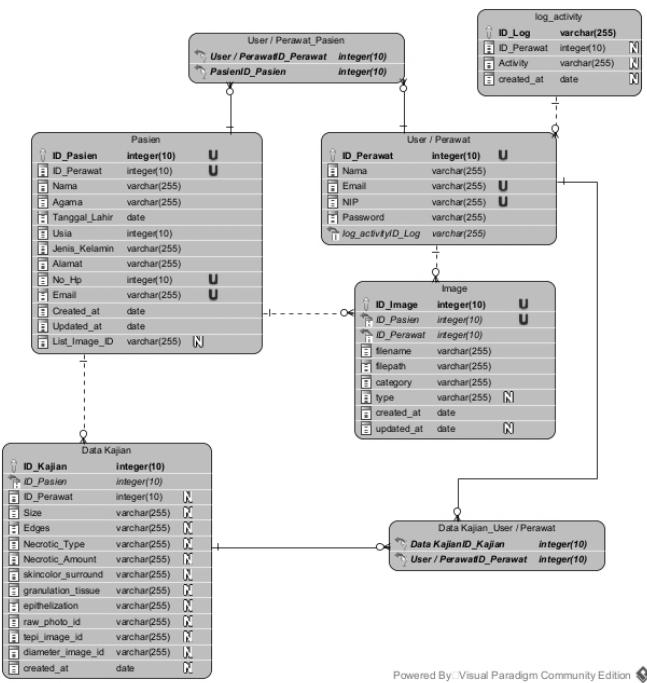
```
buttonLogout.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Set LoggedIn status to false

        SaveSharedPreference.setLoggedIn(getApplicationContext(),
            false);

        // Logout
        Intent intent = new
Intent(getApplicationContext(), FirstActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP
| Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
        Toast toast =
        Toast.makeText(getApplicationContext(), "Logout
berhasil", Toast.LENGTH_LONG);
        toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
        finish();
    }
});
```

21) Membuat database untuk *log activity*

Tabel *log\_activity* terdiri atas empat atribut diantaranya ID\_Log, ID\_Perawat, Activity, dan created\_at. Tabel log-activity memiliki relasi *one to many* dengan tabel Perawat dikarenakan satu perawat dapat memiliki banyak *log\_activity*.



## 22) Membuat routing table untuk log activity

Informasi mengenai jalur data yang akan digunakan melalui REST API dapat dilihat pada tabel di bawah ini.

Tabel 4.12 Routing Table Log Activity

Group	Name	API Endpoint	HTTP Verb	Keterangan
log_activity	INDEX	/get_logs	GET	menampilkan seluruh dokumen pada collection logging
	READ	/get_logs/<id_user>	GET	menampilkan seluruh dokumen pada collection logging yang dimiliki satu user
	CREATE	/insert_log	POST	menambahkan data log pada collection logging
	DESTROY	/delete_log/<id>	DELETE	menghapus data log_activity berdasarkan id logging

## 23) Membuat REST API untuk mencatat log activity user

```

def insert_log(data):
    collection = get_collection("logging")
    row = collection.insert_one(data)
    return row
  
```

Fungsi insert\_log terlebih dulu dibuat dengan python berisikan sintaks Pymongo yang bertujuan untuk menyimpan data ke database MongoDB. Kemudian dengan menggunakan Flask,

```

@bp.route('/insert_log', methods =['POST'])
def addLogging():
    try:
        data = {"_id" : str(uuid.uuid4().hex),
                "id_perawat" :
        request.form["id_perawat"],
                "activity" :
        request.form["activity"],
                "created_at" :
        time.strftime("%d/%m/%Y %H:%M:%S")
            }

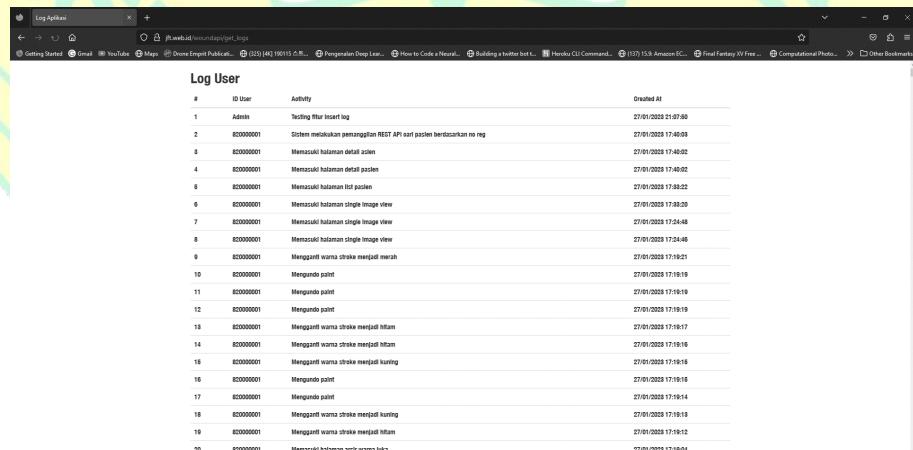
        cek = insert_log(data)
        return Response(response =
json.dumps({"message" : "true"}),
mimetype="application/json", status=200)

    except Exception as ex:
        print(ex)
        return Response(response =
json.dumps({"message" : "exe"}),
mimetype="application/json", status=500)

```

#### 24) REST API untuk melihat log activity semua user

REST API untuk melihat log activity semua user dapat diakses melalui URL [http://jft.web.id/woundapi/get\\_logs](http://jft.web.id/woundapi/get_logs). Dapat dilihat pada gambar berikut, bahwa ketika REST API dipanggil akan ditampilkan tabel informasi *log activity*.



#	ID User	Activity	Created At
1	Admin	Tutting thru insert log	27/01/2023 21:07:00
2	820000001	Sistem melakukan pemanggilan REST API pertama berdasarkan no reg	27/01/2023 17:00:00
3	820000001	Menzsaki halaman detail zulen	27/01/2023 17:46:02
4	820000001	Menzsaki halaman detail pasien	27/01/2023 17:46:02
5	820000001	Menzsaki halaman list pasien	27/01/2023 17:58:22
6	820000001	Menzsaki halaman single Image view	27/01/2023 17:58:20
7	820000001	Menzsaki halaman single Image view	27/01/2023 17:34:48
8	820000001	Menzsaki halaman single Image view	27/01/2023 17:34:46
9	820000001	Mengganti warna stroke menjadi merah	27/01/2023 17:19:21
10	820000001	Mengundo paint	27/01/2023 17:19:19
11	820000001	Mengundo paint	27/01/2023 17:19:19
12	820000001	Mengundo paint	27/01/2023 17:19:19
13	820000001	Mengganti warna stroke menjadi hitam	27/01/2023 17:19:17
14	820000001	Mengganti warna stroke menjadi hitam	27/01/2023 17:19:16
15	820000001	Mengganti warna stroke menjadi kuning	27/01/2023 17:19:16
16	820000001	Mengundo paint	27/01/2023 17:19:16
17	820000001	Mengundo paint	27/01/2023 17:19:14
18	820000001	Mengganti warna stroke menjadi kuning	27/01/2023 17:19:13
19	820000001	Mengganti warna stroke menjadi hitam	27/01/2023 17:19:12
20	820000001	Menzsaki halaman arsip werna luka	27/01/2023 17:19:04

Gambar 4.39 REST API Log Activity

### 25) REST API untuk melihat log activity 1 user berdasarkan id user

Dapat dilihat pada gambar 4.40, bahwa ketika REST API dipanggil akan ditampilkan tabel informasi *log activity* yang dimiliki oleh user tersebut.

The screenshot shows a Postman request for a REST API endpoint. The URL is `http://ft.web.id/woundapi/get_Logs/Admin`. The response status is 200 OK, time 124 ms, size 758 B. The response body is a JSON object:

```

1 {
2   "_id": "2cae63f0ad76402fb7c562c71e89bba",
3   "id_perawat": "Admin",
4   "activity": "Testing fitur insert log",
5   "created_at": "27/03/2023 21:07:50"
6 }
7
8

```

Gambar 4.40 REST API Log Activity Berdasarkan ID User

### 26) REST API untuk menghapus *log activity* 1 user berdasarkan id user

Dapat dilihat pada gambar 4.41, bahwa ketika REST API dipanggil akan ditampilkan tabel informasi *log activity* yang dimiliki oleh user tersebut.

The screenshot shows a Postman request for a REST API endpoint. The URL is `http://ft.web.id/woundapi/delete_log/2cae63f0ad76402fb7c562c71e89bba`. The response status is 200 OK, time 731 ms, size 750 B. The response body is a JSON object:

```

1 {
2   "_id": "2cae63f0ad76402fb7c562c71e89bba",
3   "id_perawat": "Admin",
4   "activity": "Testing fitur insert log",
5   "created_at": "27/03/2023 21:07:50"
6 }
7
8

```

Gambar 4.41 Menghapus Log Activity Berdasarkan ID Log

### 27) Implementasi fungsi API untuk mencatat log user pada Android

Class baru untuk mencatat log activity dibuat pada file `LogHelper`, class tersebut bersifat *public* sehingga dapat diakses pada class lain tanpa menuliskannya kembali.

```

public class LogHelper {

    public static void InsertLog( final String
        id_perawat, final String activity){
        Call<LoggingResponse> uploadRequestCall =
        RetrofitClient.getService().insertLog(id_perawat,activity
    );
}

```

```

        uploadRequestCall.enqueue(new
Callback<LoggingResponse>() {
    @Override
    public void onResponse(Call<LoggingResponse>
call, Response<LoggingResponse> response) {

        if(response.isSuccessful()){
            System.out.print("Success add log");

        }else {
            System.out.print("Something's
wrong");
        }
    }

    @Override
    public void onFailure(Call<LoggingResponse>
call, Throwable t) {
        System.out.print(t.toString());
    }
});
}
}

```

28) *Source code:*

- Android: <https://github.com/apraira/Chronic-Wound-Dataset/tree/sprint-9>
- Web Service: <https://github.com/apraira/woundapi/tree/sprint-9>

j. **Hasil sprint keseluruhan**

Bagian ini akan menjelaskan keseluruhan hasil dari iterasi *sprint* yang telah dikerjakan yaitu dari *sprint-1* hingga *sprint-9*.

1) Product Backlog Tested

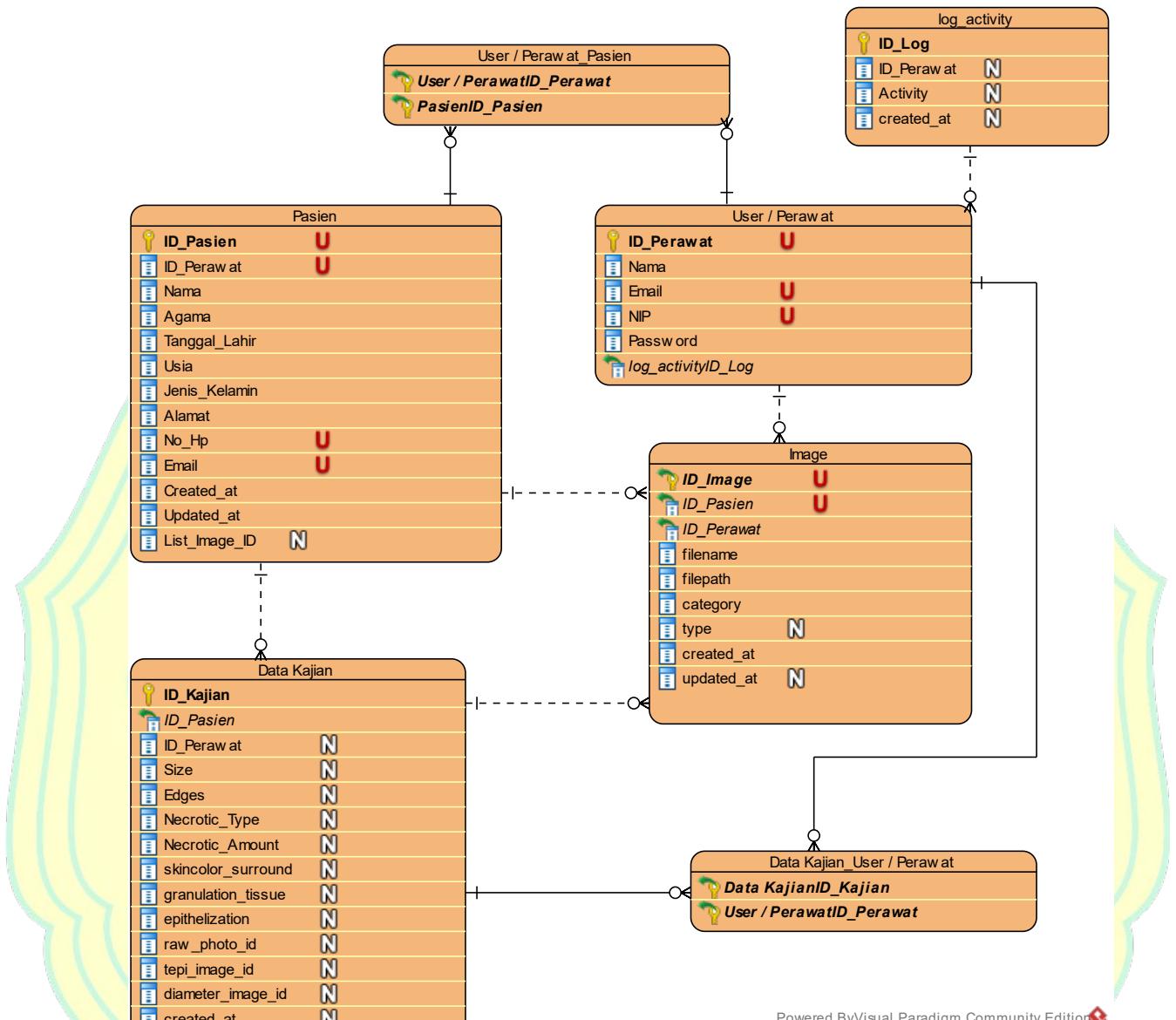
Tabel 4.13 Product Backlog Tested

No.	User Story	Priority	Sprint No.	Status
1	Konfigurasi Server	High	1	Tested
2	Konfigurasi Database	High	1	Tested
3	Membuat akun	Medium	1	Tested
4	Masuk ke akun yang sudah dibuat	Medium	1	Tested
5	Menambah pasien dan memilih pasien	High	2	Tested

No.	User Story	Priority	Sprint No.	Status
6	Manajemen fotografi luka	High	3-4	Tested
7	Melakukan penambahan pengkajian luka	High	4	Tested
8	Menganotasi tepi luka dan manganotasi orientasi luka	High	4-6	Tested
9	Mengunggah luka kronis yang akan dikaji	Medium	5	Tested
10	Mengarsir warna luka	High	7	Tested
11	Galeri luka kronis semua pasien	High	9	Tested
12	Unduh dataset luka milik user tersebut	High	9	Tested
13	Melihat histori kajian pasien dan status luka pasien	Medium	9	Tested
14	Melihat detail profil akun perawat atau pengguna	Low	9	Tested
15	Keluar dari akun	Low	9	Tested
16	Mencatat log activity user	Low	9	Tested

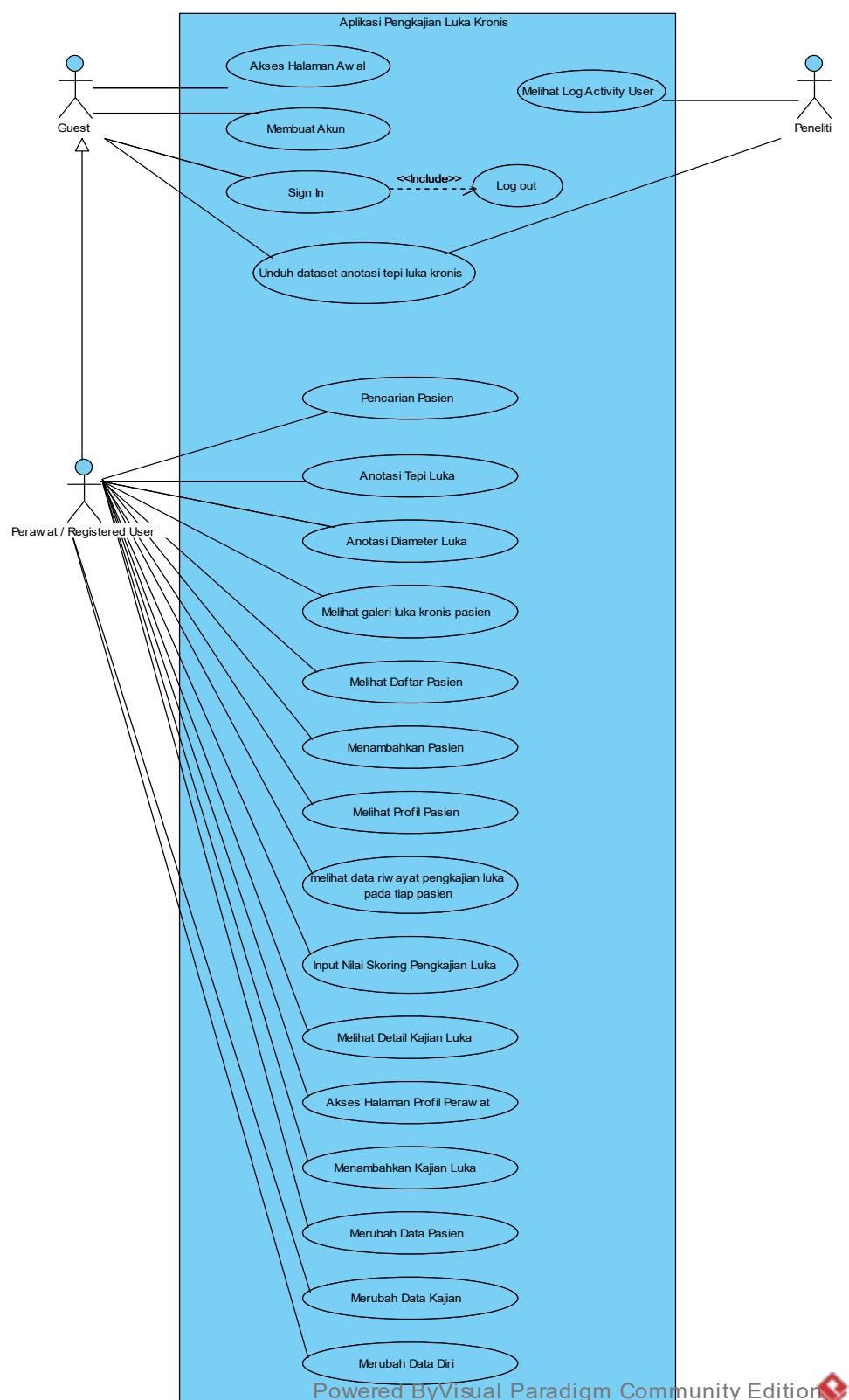


2) Desain database sistem



Gambar 4.42 Desain Database Sistem

### 3) Use Case Diagram



Gambar 4.43 Use Case Diagram

#### 4) Dokumentasi REST API/*Web Service*

Dokumentasi ini berisi penjelasan mengenai instruksi manual berupa contoh kode, tangkapan layar, dan lainnya yang membantu pembaca untuk lebih memahami bagaimana *Web Service* bekerja. Adapun penjelasannya adalah sebagai berikut:

- Meminta seluruh data user/perawat

REST API ini berfungsi untuk mengembalikan semua *document* perawat pada database.

- Request URL: <http://jft.web.id/woundapi/user>
- HTTP Method: GET
- Contoh hasil:

```
[
  {
    "_id": 820000001,
    "name": "Ns. Indah Pramesti, M.Kep.",
    "username": "002",
    "email": "indahpramesti@mail.com",
    "password": "ayamgeprek",
    "profile_image_url": "https://jft.web.id/woundapi/instance/userImage/820000001_16748872353061674887236.jpg"
  },
  {
    "_id": 820000002,
    "name": "eka",
    "username": "112233",
    "email": "ss@ss.ss",
    "password": "ssssss"
  }
]
```

- Deskripsi *field* pada hasil:

Tabel 4.14 Deskripsi Field GET All Perawat

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
_id	Integer	Id dari perawat tersebut
name	String	Nama perawat
username	String	NIP yang dimiliki perawat
email	String	Alamat surel perawat
password	String	Password yang digunakan oleh perawat

- Mendaftarkan perawat baru

REST API ini berfungsi untuk menambahkan user/perawat baru ke dalam database.

- Request URL:

[http://jft.web.id/woundapi/user/<Nama\\_Perawat>/<NI\\_P>/<email\\_perawat>/<password>](http://jft.web.id/woundapi/user/<Nama_Perawat>/<NI_P>/<email_perawat>/<password>)

- HTTP Method: POST

- Penjelasan parameter pada URL:

Tabel 4.15 Deskripsi Parameter URL Create Perawat

Field	Tipe	Deskripsi
Nama_Perawat	String	Nama perawat
NIP	String	Nomor induk yang dimiliki perawat
Email_perawat	String	Alamat surel perawat
password	String	Password yang digunakan oleh perawat

- Contoh hasil:

```
{
  "message": "true"
}
```

- Deskripsi *field* pada hasil:

Tabel 4.16 Deskripsi Field Create Perawat

Field	Tipe	Deskripsi
message	String	Jika true, maka data berhasil ditambahkan ke dalam database. Jika false maka penambahan data ke dalam database gagal.

- Validasi login

REST API ini digunakan untuk memvalidasi data perawat yang digunakan untuk fitur login.

- Request URL:

<http://jft.web.id/woundapi/user/find/<NIP>/<passw>>

- HTTP Method: GET
- Penjelasan parameter pada URL:

Tabel 4.17 Deskripsi Parameter URL Validasi Login

Field	Tipe	Deskripsi
NIP	String	Nomor induk yang dimiliki perawat
passw	String	Password yang digunakan oleh perawat

- Contoh hasil:

```
{
    "_id": 820000001,
    "name": "Ns. Indah Pramesti, M.Kep.",
    "username": "002",
    "email": "indahpramesti@mail.com",
    "password": "ayamgeprek",
    "profile_image_url": "https://jft.web.id/woundapi/instance/userImage/820000001_16748872353061674887236.jpg"
}
```

- Deskripsi *field* pada hasil:

Tabel 4.18 Deskripsi Field Validasi Login

Field	Tipe	Deskripsi
_id	Integer	Id dari perawat tersebut
name	String	Nama perawat
username	String	NIP yang dimiliki perawat
email	String	Alamat surel perawat
password	String	Password yang digunakan oleh perawat

- Memperbarui data perawat

REST API ini digunakan untuk memperbarui data perawat.

- Request URL: <http://jft.web.id/woundapi/user/update>
- HTTP Method: POST

➤ *Request Body:*

Tabel 4.19 Deskripsi *Request Body Update* Perawat

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
id_perawat	Integer	Id perawat.
jenis	String	Parameter <i>field</i> yang ingin diperbaharui
Isian	String	Isi dari parameter <i>field</i> yang ingin diperbaharui

➤ Contoh hasil:

```
{
    "message": "berhasil"
}
```

➤ Deskripsi *field* pada hasil:

Tabel 4.20 Deskripsi Field *Update* Perawat

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
message	String	Jika “berhasil” maka data user/perawat berhasil diperbaharui, jika “gagal” maka data user/perawat tidak diperbaharui.

- Menghapus data perawat

REST API ini berfungsi untuk menghapus data perawat yang terdapat pada database.

➤ *Request URL:*

<http://jft.web.id/woundapi/user/delete/<NIP>/<passw>>

➤ HTTP Method: DELETE

➤ Penjelasan parameter pada URL:

Tabel 4.21 Deskripsi Parameter URL Delete Perawat

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
NIP	String	Nomor induk yang dimiliki perawat
passw	String	Password yang digunakan oleh perawat

- Contoh hasil:

```
{
  "message": "data atas nama <NIP> sudah dihapus"
}
```

- Deskripsi *field* pada hasil:

Tabel 4.22 Deskripsi Field Hapus Perawat

Field	Tipe	Deskripsi
message	String	Pesan yang ditampilkan apakah sukses atau tidak.

- Mendapatkan seluruh data pasien

REST API ini berfungsi untuk mendapatkan seluruh data pasien yang terdapat di database.

- *Request URL*: <http://jft.web.id/woundapi/pasien>
- HTTP Method: GET
- Contoh hasil:

```
[
{
  "_id": "345679",
  "id_perawat": 820000001,
  "nama": "Diah Paramita",
  "agama": "Protestan",
  "born_date": "28-1-1999",
  "usia": "23",
  "kelamin": "Perempuan",
  "alamat": "Test",
  "no_hp": "0877776666677",
  "email": "diah@mail.com",
  "created_at": "28/01/2023 16:09:50",
  "updated_at": "28/01/2023 16:09:50",
  "list_image_id": [
    "90697a16a5e948eeba7571d2adb781a",
    "55c091988b0e4f7383fd46ba7a9baf4c",
    "8FB2D2E72A0D47E5948222AC0E369835",
    "1F3C3F12BA2648F786D89D616E5E0EC4",
    "A30A20B970C34DBF83BB37602D45096A",
    "172C3DFED49F43CDA94D9078D36CDE64",
    "9679B904133948CC91FB195AE8085115"
  ]
}
```

- Deskripsi *field* pada hasil:

Tabel 4.23 Deskripsi Field Pasien

Field	Tipe	Deskripsi
_id	String	Nomor rekam medis pasien.
id_perawat	Integer	Id perawat yang membuat data pasien baru.
nama	String	Nama pasien.
agama	String	Agama pasien.
born_date	String	Tanggal lahir pasien
usia	String	Usia pasien.
kelamin	String	Jenis kelamin pasien.
alamat	String	Alamat tempat tinggal pasien.
no_hp	String	Nomor ponsel pasien.
email	String	Alamat surel pasien.
created_at	String	Tanggal data pasien dibuat
Updated_at	String	Tanggal terakhir kali data pasien diperbarui.
List_image_id	List	Daftar id_image yang dimiliki oleh pasien tersebut.

- Mendapatkan satu data pasien

REST API ini berfungsi untuk mendapatkan satu data pasien yang terdapat di database.

- *Request URL:*

<http://jft.web.id/woundapi/pasien/<NRM>>

- HTTP Method: GET

- Penjelasan parameter pada URL:

Tabel 4.24 Deskripsi Parameter URL Get 1 Pasien

Field	Tipe	Deskripsi
NRM	String	Nomor Rekam Medis pasien

➤ Contoh hasil:

```
{
    "_id": "345679",
    "id_perawat": 820000001,
    "nama": "Diah Paramita",
    "agama": "Protestan",
    "born_date": "28-1-1999",
    "usia": "23",
    "kelamin": "Perempuan",
    "alamat": "Test",
    "no_hp": "0877776666677",
    "email": "diah@mail.com",
    "created_at": "28/01/2023 16:09:50",
    "updated_at": "28/01/2023 16:09:50",
    "list_image_id": [
        "90697a16a5e948eeba7571d2aadb781a",
        "55c091988b0e4f7383fd46ba7a9baf4c",
        "8FB2D2E72A0D47E5948222AC0E369835"
    ]
}
```

➤ Deskripsi *field* pada hasil:

Tabel 4.25 Deskripsi Field Pasien

Field	Tipe	Deskripsi
_id	String	Nomor rekam medis pasien.
id_perawat	Integer	Id perawat yang membuat data pasien baru.
nama	String	Nama pasien.
agama	String	Agama pasien.
born_date	String	Tanggal lahir pasien
usia	String	Usia pasien.
kelamin	String	Jenis kelamin pasien.
alamat	String	Alamat tempat tinggal pasien.
no_hp	String	Nomor ponsel pasien.
email	String	Alamat surel pasien.
created_at	String	Tanggal data pasien dibuat
Updated_at	String	Tanggal terakhir kali data pasien diperbarui.
List_image_id	List	Daftar id_image yang dimiliki oleh pasien tersebut.

- Membuat pasien baru

REST API ini berfungsi untuk menambahkan pasien baru ke dalam database.

- Request URL: <http://jft.web.id/woundapi/pasien>
- HTTP Method: POST
- Request Body:

Tabel 4.26 Request Body Create Pasien

Field	Tipe	Deskripsi
_id	String	Nomor rekam medis pasien.
id_perawat	Integer	Id perawat yang membuat data pasien baru.
nama	String	Nama pasien.
agama	String	Agama pasien.
born_date	String	Tanggal lahir pasien
usia	String	Usia pasien.
kelamin	String	Jenis kelamin pasien.
alamat	String	Alamat tempat tinggal pasien.
no_hp	String	Nomor ponsel pasien.
email	String	Alamat surel pasien.

- Contoh hasil:

```
{
    "message": "true"
}
```

- Deskripsi field pada hasil:

Tabel 4.27 Deskripsi Field Create Perawat

Field	Tipe	Deskripsi
message	String	Jika true, maka data berhasil ditambahkan ke dalam database. Jika false maka penambahan data ke dalam database gagal.

- Menghapus data pasien

REST API ini berfungsi untuk menghapus data pasien yang terdapat pada database.

- Request URL:

`http://jft.web.id/woundapi/pasien/<nomor_rekam_medis>`

- HTTP Method: DELETE

- Penjelasan parameter pada URL:

Tabel 4.28 Deskripsi Parameter URL Delete Perawat

Field	Tipe	Deskripsi
nomor_rekam_medis	String	Nomor rekam medis pasien yang ingin dihapus

- Contoh hasil:

Hasil yang ditampilkan merupakan data pasien yang baru saja dihapus.

```
{
  "_id": "32323",
  "id_perawat": 12323,
  "nama": "2323",
  "agama": "2323",
  "born_date": "23233",
  "usia": "2323",
  "kelamin": "2323",
  "alamat": "sdadas",
  "no_hp": "sadds",
  "email": "sadasd",
  "created_at": "04/02/2023 12:39:19",
  "updated_at": "04/02/2023 12:39:19",
  "list_image_id": []
}
```

- Mendapatkan seluruh data *image*

REST API ini berfungsi untuk mendapatkan seluruh data *image* yang terdapat di database.

- Request URL: `http://jft.web.id/woundapi/get_images`

- HTTP Method: GET

- Contoh hasil:

```
[
  {
    "_id": "05865ACBB26C41FD8127B78A5CEE6A21",
    "id_pasien": "all",
    "id_perawat": "artist",
```

```

        "filename": "IMG JPG WARNA 16739383886771673938390.
jpg",
        "filepath": "/var/www/html/woundapi/instance/upload
s",
        "type": "Jpg",
        "category": "Arsir Warna",
        "created_at": "17/01/2023 13:53:10",
        "updated_at": "17/01/2023 13:53:10"
    }
]

```

- Deskripsi *field* pada hasil:

Tabel 4.29 Deskripsi *Field Get All Image*

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
_id	String	Id dari <i>image</i> tersebut
Id_pasien	String	Nomor rekam medis pasien.
id_perawat	Integer	Id perawat yang mengambil <i>image</i>
Filename	String	Nama file <i>image</i>
Filepath	String	Lokasi dimana file <i>image</i> tersimpan di server
Type	String	Jenis ekstensi <i>image</i> , dapat berupa .jpg, .svg. atau .png
Category	String	Kategori <i>image</i> tersebut. Terdapat 4 kategori: Raw, Anotasi Tepi, Anotasi Diameter, dan Arsir Warna
created_at	String	Tanggal data <i>image</i> dibuat
Updated_at	String	Tanggal terakhir kali data diperbarui.

- Mendapatkan satu data *image*  
REST API ini berfungsi untuk mendapatkan satu data *image* yang terdapat di database berdasarkan id dari *image* tersebut.

- Request URL:

[http://jft.web.id/woundapi/get\\_images/<id\\_image>](http://jft.web.id/woundapi/get_images/<id_image>)

- HTTP Method: GET

- Penjelasan parameter pada URL:

Tabel 4.30 Deskripsi Parameter URL *Get 1 Image*

Field	Tipe	Deskripsi
id_image	String	Id dari <i>image</i> yang ingin dilihat datanya

- Contoh hasil:

```
{
    "_id": "05865ACBB26C41FD8127B78A5CEE6A21",
    "id_pasien": "all",
    "id_perawat": "artist",
    "filename": "IMG_JPG_WARNA_16739383886771673938390.jpg",
    "filepath": "/var/www/html/woundapi/instance/upload",
    "type": "Jpg",
    "category": "Arsir Warna",
    "created_at": "17/01/2023 13:53:10",
    "updated_at": "17/01/2023 13:53:10"
}
```

- Deskripsi *field* pada hasil:

Tabel 4.31 Deskripsi *Field Get 1 Image*

Field	Tipe	Deskripsi
_id	String	Id dari <i>image</i> tersebut
Id_pasien	String	Nomor rekam medis pasien.
id_perawat	Integer	Id perawat yang mengambil <i>image</i>
Filename	String	Nama file <i>image</i>
Filepath	String	Lokasi dimana file <i>image</i> tersimpan di server
Type	String	Jenis ekstensi <i>image</i> , dapat berupa .jpg, .svg. atau .png
Category	String	Kategori <i>image</i> tersebut. Terdapat 4 kategori: Raw, Anotasi Tepi, Anotasi Diameter, dan Arsir Warna
created_at	String	Tanggal data <i>image</i> dibuat
Updated_at	String	Tanggal terakhir kali data diperbarui.

- Mengunggah file *image* ke server

REST API ini berfungsi untuk mengunggah file *image* luka kronis ke server dan memasukkan datanya ke dalam database.

- *Request URL:* <http://jft.web.id/woundapi/upload>
- *HTTP Method:* POST
- *Request Body:*

Tabel 4.32 *Request Body Upload Image*

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
image	File	File foto yang ingin diunggah
id	String	Id image
Id_pasien	String	Nomor rekam medis pasien.
Id_perawat	String	Id perawat yang mengambil foto
type	String	Tipe foto yang diunggah. (JPG / PNG / Vector)
Category	String	Kategori <i>image</i> tersebut. Terdapat 4 kategori: Raw, Anotasi Tepi, Anotasi Diameter, dan Arsir Warna

- Contoh hasil:

```
{
  "message": "true"
}
```

- Deskripsi *field* pada hasil:

Tabel 4.33 Deskripsi Field *Upload Image*

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
message	String	Jika true, maka data berhasil ditambahkan ke dalam database. Jika false maka penambahan data ke dalam database gagal.

- Mendapatkan seluruh data *image* berdasarkan nomor rekam medis pasien

REST API ini berfungsi untuk mengunggah file *image* luka kronis ke server dan memasukkan datanya ke dalam database.

- *Request URL:*

[http://jft.web.id/woundapi/image/find/<id\\_pasien>](http://jft.web.id/woundapi/image/find/<id_pasien>)

- HTTP Method: POST

- Penjelasan parameter pada URL:

Tabel 4.34 Deskripsi Parameter URL *Get 1 Image*

Field	Tipe	Deskripsi
Id_pasien	String	Nomor rekam medis pasien.

- Contoh hasil:

```
[  
  {  
    "_id": "1221",  
    "id_pasien": "345679",  
    "id_perawat": "820000009",  
    "filename": "8b12bd7a9827269726fc5e7af8f20bdc167550  
4886.jpg",  
    "filepath": "/var/www/html/woundapi/instance/upload  
s",  
    "type": "u",  
    "category": "u",  
    "created_at": "04/02/2023 17:01:26",  
    "updated_at": "04/02/2023 17:01:26"  
  }  
]
```

- Deskripsi *field* pada hasil:

Tabel 4.35 Deskripsi Field *Get 1 Image*

Field	Tipe	Deskripsi
_id	String	Id dari <i>image</i> tersebut
Id_pasien	String	Nomor rekam medis pasien.
id_perawat	Integer	Id perawat yang mengambil <i>image</i>
Filename	String	Nama file <i>image</i>
Filepath	String	Lokasi dimana file <i>image</i> tersimpan di server

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
Type	String	Jenis ekstensi <i>image</i> , dapat berupa .jpg, .svg atau .png
Category	String	Kategori <i>image</i> tersebut. Terdapat 4 kategori: Raw, Anotasi Tepi, Anotasi Diameter, dan Arsir Warna
created_at	String	Tanggal data <i>image</i> dibuat
Updated_at	String	Tanggal terakhir kali data diperbarui.

- Menghapus data *image*

REST API ini berfungsi untuk menghapus data *image* pada database dan *image* yang terdapat pada server.

➤ *Request URL:*

[http://jft.web.id/woundapi/delete\\_image/<id>](http://jft.web.id/woundapi/delete_image/<id>)

➤ *HTTP Method:* DELETE

➤ Penjelasan parameter pada URL:

Tabel 4.36 Deskripsi Parameter URL *Delete Image*

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
id	String	Id dari <i>image</i> yang ingin dihapus

➤ Contoh hasil:

Hasil yang ditampilkan merupakan data pasien yang baru saja dihapus.

```
{
  "_id": "1221",
  "id_pasien": "345679",
  "id_perawat": "820000009",
  "filename": "8b12bd7a9827269726fc5e7af8f20bdc1675504886.jpg",
  "filepath": "/var/www/html/woundapi/instance/uploads",
  "type": "u",
  "category": "u",
  "created_at": "04/02/2023 17:01:26",
  "updated_at": "04/02/2023 17:01:26"
}
```

- Mendapat seluruh data kajian

REST API ini berfungsi untuk mendapatkan seluruh data kajian yang terdapat di database.

- Request URL: [http://jft.web.id/woundapi/get\\_kajians](http://jft.web.id/woundapi/get_kajians)
- HTTP Method: GET
- Contoh hasil:

```
[
  {
    "_id": "f2f8dcfb7d243e7a4c83d2208965b1a",
    "id_pasien": "004",
    "id_perawat": "820000001",
    "size": "1 = kurang dari 4cm²",
    "edges": "1 = Tidak terlihat jelas",
    "necrotic_type": "4 = Adherent/menempel, soft, \nne\nr dapat black eschar",
    "necrotic_amount": "3 = 25% sampai 50%\nof wound co\nvered",
    "skincolor_surround": "2 = Merah terang,\npu\nca\nka disentuh",
    "granulation": "2 = Cerah, merah daging;\n75% sampa\nai 100% luka terisi\n&/atau jaringan tumbuh berlebih",
    "epithelialization": "1 = 100% wound covered,\npermuka\nan utuh",
    "raw_photo_id": "641B5A5F98DB458F86620C31F8672373",
    "tepi_image_id": "510AE4C54689488F8C29FAED0F087256"
  ,
    "diameter_image_id": "92E0EBC2FF3D4A7C8925D9BFACA5E\nF26",
    "created_at": "17/01/2023 13:56:36"
  }
]
```

- Deskripsi *field* pada hasil:

Tabel 4.37 Deskripsi *Field Get All Image*

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
_id	String	Id data kajian
Id_pasien	String	Id pasien
id_perawat	Integer	Id perawat
Size	String	Luas luka pasien
Edges	String	Kondisi tepi luka pasien
Necrotic_type	String	Tipe jaringan nekrotik

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
Necrotic_amount	String	Jumlah jaringan nekrotik pada luka pasien
Skincolor_surround	String	Warna kulit pada sekitar luka pasien
Granulation	String	Kondisi jaringan granulasi pada luka pasien
Epithelization	String	Kondisi jaringan epitel pada luka pasien
Raw_photo_id	String	Id foto luka mentah tanpa adanya anotasi
Tepi_image_id	String	Id foto anotasi tepi luka pasien
Diameter_image_id	String	Id foto anotasi diameter luka pasien
Created_at	String	Tanggal data kajian dibuat

- Mendapatkan satu data kajian

REST API ini berfungsi untuk mendapatkan satu data kajian yang terdapat di database.

➤ *Request URL:*

[http://jft.web.id/woundapi/get\\_kajian/<id>](http://jft.web.id/woundapi/get_kajian/<id>)

➤ *HTTP Method:* GET

➤ *Penjelasan parameter pada URL:*

Tabel 4.38 Deskripsi Parameter URL *Delete Image*

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
id	String	Id data kajian

➤ Contoh hasil:

```
{
    "_id": "f2f8dcfb7d243e7a4c83d2208965b1a",
    "id_pasien": "004",
    "id_perawat": "820000001",
    "size": "1 = kurang dari 4cm²",
    "edges": "1 = Tidak terlihat jelas",
    "necrotic_type": "4 = Adherent/menempel, soft, \n terdapat black eschar",
    "necrotic_amount": "3 = 25% sampai 50%\nof wound covered",
```

```

    "skincolor_surround": "2 = Merah terang,\n pucat jika disentuh",
    "granulation": "2 = Cerah, merah daging;\n75% sampai 100% luka terisi\n&/atau jaringan tumbuh berlebih",
    "epithelization": "1 = 100% wound covered,\npermukaan utuh",
    "raw_photo_id": "641B5A5F98DB458F86620C31F8672373",
    "tepi_image_id": "510AE4C54689488F8C29FAED0F087256"
  ,
  "diameter_image_id": "92E0EBC2FF3D4A7C8925D9BFACA5EF26",
  "created_at": "17/01/2023 13:56:36"
}

```

➤ Deskripsi *field* pada hasil:

Tabel 4.39 Deskripsi *Field Get All Image*

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
_id	String	Id data kajian
Id_pasien	String	Id pasien
id_perawat	Integer	Id perawat
Size	String	Luas luka pasien
Edges	String	Kondisi tepi luka pasien
Necrotic_type	String	Tipe jaringan nekrotik
Necrotic_amount	String	Jumlah jaringan nekrotik pada luka pasien
Skincolor_surround	String	Warna kulit pada sekitar luka pasien
Granulation	String	Kondisi jaringan granulasi pada luka pasien
Epithelization	String	Kondisi jaringan epitel pada luka pasien
Raw_photo_id	String	Id foto luka mentah tanpa adanya anotasi
Tepi_image_id	String	Id foto anotasi tepi luka pasien
Diameter_image_id	String	Id foto anotasi diameter luka pasien
Created_at	String	Tanggal data kajian dibuat

- Membuat data kajian baru

REST API ini berfungsi untuk membuat data kajian baru ke dalam database.

- Request URL: [http://jft.web.id/woundapi/insert\\_kajian](http://jft.web.id/woundapi/insert_kajian)
- HTTP Method: POST
- Request Body:

Tabel 4.40 Request Body Membuat Kajian Baru

Field	Tipe	Deskripsi
Id_pasien	String	Id pasien
id_perawat	Integer	Id perawat
Size	String	Luas luka pasien
Edges	String	Kondisi tepi luka pasien
Necrotic_type	String	Tipe jaringan nekrotik
Necrotic_amount	String	Jumlah jaringan nekrotik pada luka pasien
Skincolor_surround	String	Warna kulit pada sekitar luka pasien
Granulation	String	Kondisi jaringan granulasi pada luka pasien
Epithelization	String	Kondisi jaringan epitel pada luka pasien
Raw_photo_id	String	Id foto luka mentah tanpa adanya anotasi
Tepi_image_id	String	Id foto anotasi tepi luka pasien
Diameter_image_id	String	Id foto anotasi diameter luka pasien

- Contoh hasil:

```
{
  "_id": "5e0c953f3e55466688d9f32741d34698",
  "id_pasien": "q",
  "id_perawat": "q",
  "size": "w",
  "edges": "e",
  "necrotic_type": "d",
  "necrotic_amount": "f",
  "skincolor_surround": "g",
  "granulation": "g",
  "epithelization": "v",
}
```

```

    "raw photo id": "v",
    "tepi image id": "g",
    "diameter image id": "g",
    "created at": "04/02/2023 17:38:14"
}

```

- Deskripsi *field* pada hasil:

Tabel 4.41 Deskripsi Field Membuat Kajian Baru

Field	Tipe	Deskripsi
_id	String	Id data kajian
Id_pasien	String	Id pasien
id_perawat	Integer	Id perawat
Size	String	Luas luka pasien
Edges	String	Kondisi tepi luka pasien
Necrotic_type	String	Tipe jaringan nekrotik
Necrotic_amount	String	Jumlah jaringan nekrotik pada luka pasien
Skincolor_surround	String	Warna kulit pada sekitar luka pasien
Granulation	String	Kondisi jaringan granulasi pada luka pasien
Epithelization	String	Kondisi jaringan epitel pada luka pasien
Raw_photo_id	String	Id foto luka mentah tanpa adanya anotasi
Tepi_image_id	String	Id foto anotasi tepi luka pasien
Diameter_image_id	String	Id foto anotasi diameter luka pasien
Created_at	String	Tanggal data kajian dibuat

- Mendapatkan data kajian berdasarkan nomor rekam medis pasien

REST API ini berfungsi untuk mendapatkan seluruh data kajian yang dimiliki oleh pasien.

- Request URL:

[http://jft.web.id/woundapi/get\\_kajian/pasien/<nomor\\_rekam\\_medis>](http://jft.web.id/woundapi/get_kajian/pasien/<nomor_rekam_medis>)

- HTTP Method: GET
- Penjelasan parameter pada URL:

Tabel 4.42 Deskripsi Parameter URL Hapus Data Kajian

Field	Tipe	Deskripsi
Nomor_rekam_medis	String	Nomor rekam medis pasien

- Contoh hasil:

```
[
{
    "_id": "f2f8dcfbb7d243e7a4c83d2208965b1a",
    "id_pasien": "004",
    "id_perawat": "820000001",
    "size": "1 = kurang dari 4cm²",
    "edges": "1 = Tidak terlihat jelas",
    "necrotic_type": "4 = Adherent/menempel, soft, \n tetapi dapat black eschar",
    "necrotic_amount": "3 = 25% sampai 50%\nof wound covered",
    "skincolor_surround": "2 = Merah terang,\n pucat jika disentuh",
    "granulation": "2 = Cerah, merah daging;\n75% sampai 100% luka terisi\n&/atau jaringan tumbuh berlebih",
    "epithelialization": "1 = 100% wound covered,\npermukaan utuh",
    "raw_photo_id": "641B5A5F98DB458F86620C31F8672373",
    "tepi_image_id": "510AE4C54689488F8C29FAED0F087256"
    ,
    "diameter_image_id": "92E0EBC2FF3D4A7C8925D9BFACA5EF26",
    "created_at": "17/01/2023 13:56:36"
}
]
```

- Deskripsi *field* pada hasil:

Tabel 4.43 Deskripsi Field Membuat Kajian Baru

Field	Tipe	Deskripsi
_id	String	Id data kajian
Id_pasien	String	Id pasien
id_perawat	Integer	Id perawat
Size	String	Luas luka pasien
Edges	String	Kondisi tepi luka pasien
Necrotic_type	String	Tipe jaringan nekrotik

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
Necrotic_amount	String	Jumlah jaringan nekrotik pada luka pasien
Skincolor_surround	String	Warna kulit pada sekitar luka pasien
Granulation	String	Kondisi jaringan granulasi pada luka pasien
Epithelization	String	Kondisi jaringan epitel pada luka pasien
Raw_photo_id	String	Id foto luka mentah tanpa adanya anotasi
Tepi_image_id	String	Id foto anotasi tepi luka pasien
Diameter_image_id	String	Id foto anotasi diameter luka pasien
Created_at	String	Tanggal data kajian dibuat

- Menghapus data kajian

REST API ini berfungsi untuk mendapatkan seluruh data kajian yang dimiliki oleh pasien.

➤ Request URL:

[http://jft.web.id/woundapi/delete\\_kajian/<id>](http://jft.web.id/woundapi/delete_kajian/<id>)

➤ HTTP Method: DELETE

➤ Penjelasan parameter pada URL:

Tabel 4.44 Deskripsi Parameter URL Hapus Data Kajian

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
id	String	Id data kajian

➤ Contoh hasil:

```
{
  "message": "1 kajian deleted"
}
```

➤ Deskripsi *field* pada hasil:

Tabel 4.45 Deskripsi Field Hapus Data Kajian

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
message	String	Pesan status apakah data

Field	Tipe	Deskripsi
		berhasil dihapus atau tidak.

- Mendapatkan seluruh *log activity*

REST API ini berfungsi untuk mendapatkan seluruh data *log activity* yang terdapat di database.

- Request URL: [http://jft.web.id/woundapi/get\\_logs](http://jft.web.id/woundapi/get_logs)
- HTTP Method: GET
- Contoh hasil:

#	ID Log	ID User	Activity	Created At
1	66129573990942190e4d2871e81f6d5	820000006	Menari id berdasarkan NIP	31/01/2023 09:12:26
2	e700b0864846793b9e6208fb0f22329f	820000006	Menari id berdasarkan NIP	31/01/2023 09:12:26
3	498360ca02294be83799ff4ef9aa3a	Guest	Data guest fedap pada database, akan diarahkan ke halaman utama	31/01/2023 09:12:26
4	9971de0888494840860f0f48277166d	Guest	Data akun deek ke database	31/01/2023 09:12:26
5	36577660aa0494985cdab5a607171df	Guest	Meneleks tombol login pada halaman login	31/01/2023 09:12:26
6	3f48b660b8439996e7d1738a7a07	Sistem	Validasi istan form login	31/01/2023 09:12:26
7	d659172910340778b2620647248fb0f	Guest	Memasuki halaman Login	31/01/2023 09:12:17
8	be44839e7f544142ba4e0b797338e56	Guest	Meneleks tombol login	31/01/2023 09:12:16
9	0e9960f08f7472a23a4941e0e0116161	Guest	Memasuki halaman Pertama	31/01/2023 09:12:16
10	0210a01093774868bd401e92b01ae93	Guest	Memasuki halaman Pertama	31/01/2023 09:12:15
11	60f184916174941909569d1283403b	820000006	Meneleks tombol menuju detail profil perawat	31/01/2023 09:12:12
12	0x7ab0b6d9259a7e7aef11762f256824	820000006	Menari id berdasarkan NIP	31/01/2023 09:12:11
13	6f6282f05f454829823a3a49b004a	820000006	Menari id berdasarkan NIP	31/01/2023 09:12:11
14	ab0d444a47940111baaf588f538fb	820000006	Memasuki halaman utama	31/01/2023 09:12:11
15	b011679111446bb9a464660113f9a1	820000006	Memasuki halaman list pasien	31/01/2023 09:12:02
16	d3042717ea94ad886841243a49728e	820000006	Melakukan pencarian id perawat untuk disimpan sebagai sharedPreferences	31/01/2023 09:12:02
17	87d9d19dd088441aead9d8d0a0836e77	820000006	Meneleks tombol untuk menju list pasien	31/01/2023 09:12:02
18	038cf796d04601bd001e460210775	820000006	Menari id berdasarkan NIP	31/01/2023 09:12:01
19	0a671bd00690489998bf04a3e171d0	820000006	Menari id berdasarkan NIP	31/01/2023 09:11:33
20	j768477867474068a2cb29e976963	820000006	Menari id berdasarkan NIP	31/01/2023 09:11:33
21	69e90307b0a4411a8e497aet19069e6	820000001	Memasuki halaman utama	31/01/2023 09:11:33
22	f0e879306414f88747d0baaa1d8	Guest	Data guest fedap pada database, akan diarahkan ke halaman utama	31/01/2023 09:11:33
99	a77da504538a6793977a6995115310	Sistem	Validasi istan form login	31/01/2023 09:11:33

Gambar 4.44 Hasil REST API Log Activity

- Deskripsi *field* pada hasil:

Tabel 4.46 Deskripsi *Field* Seluruh Log Activity

Field	Tipe	Deskripsi
_id	String	Id <i>log activity</i>
id_perawat	Integer	Id perawat
Activity	String	Aktivitas yang dilakukan perawat

- Mendapatkan seluruh *log activity* berdasarkan id perawat  
REST API ini berfungsi untuk mendapatkan seluruh data *log activity* yang dimiliki oleh perawat.

➤ Request URL:

[http://jft.web.id/woundapi/get\\_logs/<id\\_perawat>](http://jft.web.id/woundapi/get_logs/<id_perawat>)

➤ HTTP Method: GET

➤ Penjelasan parameter pada URL:

Tabel 4.47 Deskripsi Parameter URL *Log Activity Perawat*

Field	Tipe	Deskripsi
Id_perawat	String	Id perawat

➤ Contoh hasil:

```
[  
  {  
    "_id": "986ac30b45a24223b0e665e442a71b21",  
    "id_perawat": "820000001",  
    "activity": "Mencari id berdasarkan NIP",  
    "created_at": "17/01/2023 13:14:07"  
  },  
  {  
    "_id": "155611d080414a43bc408ef21bd317db",  
    "id_perawat": "820000001",  
    "activity": "Menekan tombol untuk menuju list pasien",  
    "created_at": "17/01/2023 13:14:09"  
  }  
]
```

➤ Deskripsi *field* pada hasil:

Tabel 4.48 Deskripsi Field *Log Activity Perawat*

Field	Tipe	Deskripsi
_id	String	Id <i>log activity</i>
id_perawat	Integer	Id perawat
Activity	String	Aktivitas yang dilakukan perawat
Created_at	String	Tanggal <i>log activity</i> dibuat

- Membuat *log activity* baru

REST API ini berfungsi untuk membuat *log activity* baru ke dalam database.

- Request URL: [http://jft.web.id/woundapi/insert\\_log](http://jft.web.id/woundapi/insert_log)
- HTTP Method: POST
- Request Body:

Tabel 4.49 Request Body Membuat Log Activity Baru

Field	Tipe	Deskripsi
id_perawat	Integer	Id perawat
Activity	String	Aktivitas yang dilakukan perawat ketika menjalankan aplikasi

- Contoh hasil:

```
{
  "message": "true"
}
```

- Deskripsi field pada hasil:

Tabel 4.50 Deskripsi Field Membuat Log Activity Baru

Field	Tipe	Deskripsi
message	String	Jika true, maka data berhasil ditambahkan ke dalam database. Jika false maka penambahan data ke dalam database gagal.

- Menghapus log berdasarkan id *log activity*

REST API ini berfungsi untuk menghapus satu *log activity* yang terdapat ada database.

- Request URL:  
[http://jft.web.id/woundapi/delete\\_log/<id>](http://jft.web.id/woundapi/delete_log/<id>)
- HTTP Method: DELETE

- Penjelasan parameter pada URL:

Tabel 4.51 Deskripsi Parameter URL Hapus *Log Activity*

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
id	String	Id <i>log activity</i>

- Contoh hasil:

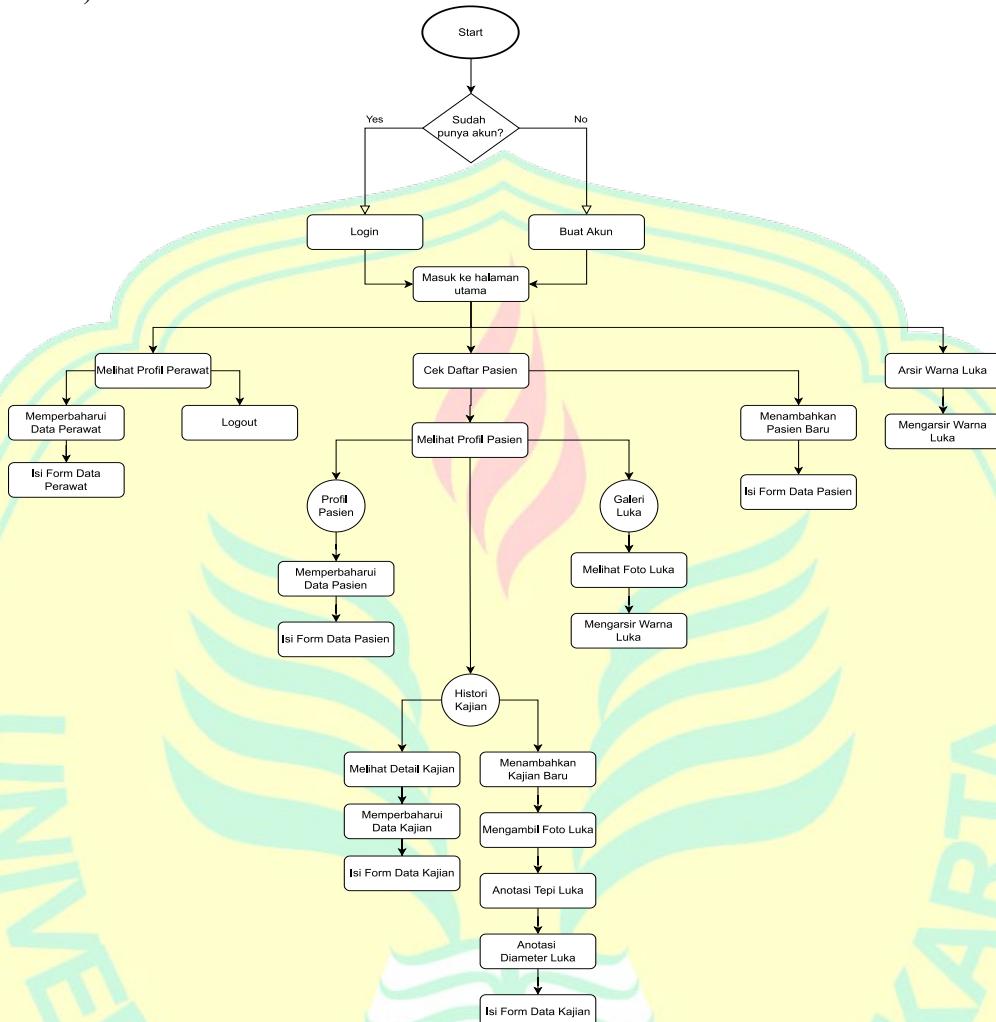
```
{
  "_id": "7d28449f7e4d498bbd6d8dc5fbd26b12",
  "id_perawat": "a",
  "activity": "s",
  "created_at": "04/02/2023 18:11:27"
}
```

- Deskripsi *field* pada hasil:

Tabel 4.52 Deskripsi Field Hapus *Log Activity*

<b>Field</b>	<b>Tipe</b>	<b>Deskripsi</b>
_id	String	Id <i>log activity</i>
id_perawat	Integer	Id perawat
Activity	String	Aktivitas yang dilakukan perawat
Created_at	String	Tanggal <i>log activity</i> dibuat

### 5) Flowchart Keseluruhan



Gambar 4.45 Flowchart Aplikasi

### 6) Source code Keseluruhan

- Android: <https://github.com/apraira/Chronic-Wound-Dataset/tree/final>
- Web Service: <https://github.com/apraira/woundapi/tree/final>

## B. Uji Aplikasi

Pengujian aplikasi dilakukan menggunakan dua tipe yaitu *Unit Testing* dan *User Acceptance Test* (UAT). Pengujian dilaksanakan pada saat seluruh *User Story* pada *Product Backlog* telah diimplementasikan. *Unit testing* aplikasi dilakukan terhadap satu *internal developer*, sedangkan pengujian *User Acceptance Test* dilakukan terhadap satu penguji ahli atau perawat.

### a. Hasil *Unit Testing*

*Unit testing* terhadap satu *internal developer* dilaksanakan pada tanggal 26 Januari 2023 secara daring melalui WhatsApp. Adapun hasil dari *unit testing* yang telah dilaksanakan dapat dilihat pada tabel di bawah ini:

Tabel 4.53 Pengujian Halaman Awal

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat aplikasi dibuka akan muncul <i>permission request</i> untuk menggunakan kamera dan akses ke storage	✓		Diterima
2	Saat <i>permission</i> disetujui maka akan masuk ke tampilan awal, jika tidak, maka tidak dapat masuk ke aplikasi	✓		Diterima
3	Jika tombol "LOG IN" ditekan, guest akan masuk ke halaman Login	✓		Diterima
4	Jika tombol "Buat Akun" ditekan maka akan masuk ke halaman Buat Akun	✓		Diterima

Tabel 4.54 Pengujian Registrasi User/Perawat

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Ketika mengisi form registrasi dengan lengkap kemudian submit, maka akan masuk ke halaman beranda	✓		Diterima
2	Ketika mengisi form registrasi tidak lengkap kemudian submit, maka akan menampilkan pesan kesalahan	✓		Diterima

Tabel 4.55 Pengujian Login User

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Ketika mengisi form login dengan data yang sesuai kemudian klik submit, maka akan masuk ke halaman beranda	✓		Diterima
2	Ketika mengisi form login dengan data yang tidak sesuai kemudian klik submit, maka akan menampilkan pesan kesalahan	✓		Diterima

Tabel 4.56 Pengujian Halaman Beranda

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat ikon profil pada kanan atas di-klik maka akan muncul halaman profil perawat	✓		Diterima
2	Saat tombol daftar pasien di-klik maka akan muncul halaman daftar pasien	✓		Diterima
3	Saat tombol galeri foto luka di-klik maka akan muncul halaman galeri foto luka	✓		Diterima
4	Saat tombol arsir warna luka di-klik maka akan muncul halaman arsir warna luka	✓		Diterima

Tabel 4.57 Pengujian Halaman Daftar Pasien

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat ikon panah pada kiri atas diklik akan kembali ke halaman beranda	✓		Diterima
2	Saat ikon lup pada kanan atas diklik maka akan muncul isian untuk mencari pasien	✓		Diterima
3	Saat mencari pasien dengan <i>input</i> nama yang terdapat pada database, maka akan muncul nama pasien tersebut	✓		Diterima
4	Saat mencari pasien dengan <i>input</i> nama yang tidak terdapat pada database, maka akan muncul pesan bahwa tidak	✓		Diterima

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
	ditemukan			
5	Saat mencari pasien dengan <i>input</i> nomor rekam medis pasien yang terdapat pada database, maka akan muncul nama pasien sesuai dengan nomor registrasi yang telah di- <i>input</i>	✓		Diterima
6	Saat mencari pasien dengan <i>input</i> nomor rekam medis pasien yang tidak terdapat pada database, maka akan muncul pesan bahwa tidak ditemukan	✓		Diterima
7	Saat menekan tombol Laki-laki maka akan muncul seluruh pasien laki-laki	✓		Diterima
8	Saat menekan tombol Perempuan maka akan muncul seluruh pasien perempuan	✓		Diterima
9	Saat menekan tombol tambah pasien baru pada kanan bawah, maka akan muncul halaman penambahan pasien baru	✓		Diterima
10	Saat klik nama pasien, maka akan muncul halaman detail pasien	✓		Diterima

Tabel 4.58 Pengujian Tambah Pasien Baru

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Ketika mengisi form pasien dengan lengkap kemudian klik submit, maka akan masuk ke halaman detail pasien yang baru saja diinput	✓		Diterima
2	Ketika mengisi form pasien tidak lengkap kemudian submit, maka akan menampilkan pesan kesalahan	✓		Diterima

Tabel 4.59 Pengujian Halaman Detail Pasien

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat masuk pertama kali ke halaman detail pasien, maka informasi umum pasien	✓		Diterima

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
	terlihat			
2	Saat klik ikon pensil maka akan masuk ke halaman <i>edit</i> data pasien	✓		Diterima
3	Saat <i>tab</i> Histori Kajian diklik maka akan muncul histori kajian pasien	✓		Diterima
4	Saat <i>tab</i> Galeri Luka diklik maka akan muncul seluruh foto luka milik pasien	✓		Diterima
5	Saat tombol Raw pada <i>tab</i> Galeri Luka diklik, maka akan muncul semua foto luka tanpa anotasi dengan format JPG.	✓		Diterima
6	Saat tombol Anotasi Tepi pada <i>tab</i> Galeri Luka diklik, maka akan muncul semua foto luka beranotasi tepi dengan format JPG.	✓		Diterima
7	Saat tombol Anotasi Diameter pada <i>tab</i> Galeri Luka diklik, maka akan muncul semua foto luka beranotasi diameter dengan format JPG.	✓		Diterima
8	Saat klik tombol "Tambah Hasil Kajian" pada <i>tab</i> Histori Kajian, maka akan diarahkan ke halaman tambah hasil kajian.	✓		Diterima
9	Saat klik tombol tanggal yang terdapat pada <i>tab</i> Histori Kajian, maka akan diarahkan menuju halaman detail kajian luka sesuai dengan tanggal yang dipilih.	✓		Diterima
10	Saat klik ikon pensil pada sebelah kanan teks "Informasi Umum Pasien" maka akan diarahkan menuju halaman Edit Data Pasien	✓		Diterima

Tabel 4.60 Pengujian Halaman Edit Detail Pasien

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat pertama kali masuk ke halaman Edit Data Pasien maka akan ditampilkan kembali informasi umum yang telah ada serta ikon pensil di sebelah kanannya	✓		Diterima
2	Saat klik ikon pensil, maka akan ditampilkan pesan perintah, kolom isian, dan tombol <i>submit</i>	✓		Diterima

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
3	Ketika tidak mengisi kolom kemudian klik submit, maka akan ditampilkan pesan error	✓		Diterima
4	Ketika mengisi kolom dengan lengkap kemudian klik submit, maka akan masuk ke halaman Edit Data Pasien dengan data yang sudah diperbarui	✓		Diterima

Tabel 4.61 Pengujian Tambah Hasil Kajian

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat pertama kali masuk ke halaman tambah hasil kajian baru hanya terdapat nama pasien, nomor rekam medis, jenis kelamin, usia, dan tombol kamera.	✓		Diterima
2	Ketika pengguna belum mengambil foto luka maka tidak akan muncul form kajian luka.	✓		Diterima
3	Ketika pengguna mengklik tombol kamera, maka akan masuk ke halaman pengambilan foto luka.	✓		Diterima
4	Ketika pengguna selesai mengambil foto luka, maka akan diarahkan ke halaman anotasi tepi luka.	✓		Diterima
5	Ketika pengguna selesai menganotasi tepi luka dan klik tombol "SAVE" maka akan diarahkan untuk manganotasi diameter koordinat X.	✓		Diterima
6	Ketika pengguna selesai manganotasi diameter luka koordinat X dan klik tombol "SAVE" maka diarahkan untuk manganotasi diameter koordinat Y.	✓		Diterima
7	Ketika pengguna selesai manganotasi diameter luka koordinat X dan klik tombol "SAVE" maka diarahkan untuk manganotasi diameter koordinat Y.	✓		Diterima
8	Ketika pengguna selesai manganotasi diameter luka koordinat Y dan klik tombol "SAVE" maka diarahkan untuk mengisi form kajian luka.	✓		Diterima

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
9	Ketika mengisi form kajian dengan lengkap kemudian klik submit, maka akan masuk ke halaman detail pasien yang baru saja diinput.	✓		Diterima
10	Ketika mengisi form kajian secara tidak lengkap kemudian klik submit, maka akan masuk ke halaman detail pasien yang baru saja diinput.	✓		Diterima

Tabel 4.62 Pengujian Halaman Detail Kajian

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat masuk pertama kali ke halaman detail kajian, maka akan ditampilkan informasi tanggal kajian, nama pasien, perawat yang menangani, NIP perawat, foto luka, anotasi tepi luka, anotasi diameter luka, skoring kajian luka, dan ikon pensil untuk mengedit data	✓		Diterima
2	Saat klik ikon pensil di atas foto luka atau foto anotasi, maka akan masuk ke halaman untuk melakukan foto ulang atau anotasi ulang	✓		Diterima
3	Saat klik ikon pensil pada sisi kanan tulisan “Skoring Kajian Luka” maka akan masuk ke halaman Edit Skoring Kajian	✓		Diterima
4	Saat klik ikon panah pada kiri atas, maka akan kembali ke halaman detail pasien.	✓		Diterima

Tabel 4.63 Pengujian Edit Skoring Kajian

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat pertama kali masuk ke halaman Edit Skoring Kajian maka akan ditampilkan kembali informasi skoring yang telah ada serta ikon pensil di sebelah kanannya	✓		Diterima
2	Saat klik ikon pensil, maka akan ditampilkan pesan perintah, kolom isian, dan tombol <i>submit</i>	✓		Diterima
3	Ketika tidak mengisi kolom kemudian klik	✓		Diterima

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
	submit, maka akan ditampilkan pesan error			
4	Ketika mengisi kolom dengan lengkap kemudian klik submit, maka akan masuk ke ke halaman Edit Skoring Kajian dengan data yang sudah diperbarui	✓		Diterima

Tabel 4.64 Pengujian Halaman Detail Foto

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat pertama kali masuk ke halaman detail foto, maka akan ditampilkan foto tersebut (kecuali yang berformat SVG), informasi filename, informasi nomor rekam medis pasien, informasi NIP perawat yang mengambil foto, tanggal unggah, dan kategori.	✓		Diterima

Tabel 4.65 Pengujian Arsir Warna Luka

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat klik ikon galeri biru yang terdapat saat pertama kali membuka fitur arsir warna luka, maka pengguna diarahkan untuk mengunggah foto dari galeri.	✓		Diterima
2	Ketika pengguna klik tombol panah pada kiri atas, maka akan diarahkan kembali ke halaman sebelumnya.	✓		Diterima
3	Pada saat pengguna menggambar, warna yang pertama kali dipakai adalah warna hitam.	✓		Diterima
4	Ketika tombol "Red" diklik maka warna <i>stroke</i> akan berubah menjadi warna merah.	✓		Diterima
5	Ketika tombol "Yellow" diklik maka warna <i>stroke</i> akan berubah menjadi warna kuning.	✓		Diterima
6	Ketika tombol "Black" diklik maka warna <i>stroke</i> akan berubah menjadi warna hitam.	✓		Diterima
7	Ketika tombol "Undo" diklik maka stroke yang baru saja digambar akan hilang.	✓		Diterima

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
8	Ketika ikon bulat biru digeser ke kanan, maka <i>stroke</i> akan semakin lebar.	✓		Diterima
9	Ketika selesai mengarsir warna luka dan klik tombol save, maka foto akan diunggah ke server lalu kembali ke halaman beranda.	✓		Diterima

Tabel 4.66 Pengujian Halaman Detail Perawat

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat pertama kali masuk ke halaman Profil Perawat maka akan ditampilkan nama perawat, NIP perawat, email perawat, tombol <i>log out</i> , dan ikon pensil	✓		Diterima
2	Ketika tombol "LOG OUT" diklik, maka pengguna akan kembali ke halaman awal dan tidak bisa mengakses fitur pada aplikasi.	✓		Diterima
3	Ketika ikon pensil diklik maka akan diarahkan ke halaman edit data perawat.	✓		Diterima

Tabel 4.67 Pengujian Fitur Edit Perawat

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat pertama kali masuk ke halaman Edit Profil Perawat maka akan ditampilkan kalimat perintah, kolom teks untuk mengedit, dan tombol <i>submit</i> .	✓		Diterima
2	Ketika mengisi kolom dengan lengkap kemudian klik submit, maka akan masuk ke halaman Profil Pasien	✓		Diterima
3	Ketika tidak mengisi kolom kemudian klik submit, maka akan ditampilkan pesan error	✓		Diterima

Tabel 4.68 Pengujian Fitur Log Activity User

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat mengunjungi link <a href="http://jft.web.id/woundapi/get_logs">http://jft.web.id/woundapi/get_logs</a> maka akan ditampilkan histori logging seluruh pengguna.	✓		Diterima
2	Ketika tombol "LOG OUT" diklik, maka pengguna akan kembali ke halaman awal dan tidak bisa mengakses fitur pada aplikasi.	✓		Diterima

Tabel 4.69 Pengujian Fitur Unduh Dataset Luka

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat mengunjungi link <a href="http://jft.web.id/woundapi/">http://jft.web.id/woundapi/</a> maka akan ditampilkan halaman untuk mengunduh dataset foto luka kronis.	✓		Diterima
2	Saat tombol "Unduh Semua Foto" diklik, maka akan mengunduh semua foto luka kronis tanpa terkecuali.	✓		Diterima
3	Saat tombol "Unduh" pada card Raw Image diklik, maka akan mengunduh semua foto luka kronis tanpa anotasi.	✓		Diterima
4	Saat tombol "Unduh" pada card Anotasi Tepi diklik, maka akan mengunduh semua foto anotasi tepi luka kronis.	✓		Diterima
5	Saat tombol "Unduh" pada card Anotasi Diameter diklik, maka akan mengunduh semua foto anotasi diameter luka kronis.	✓		Diterima
6	Setelah mengisikan nomor rekam medis pada form yang tersedia lalu menekan tombol "Unduh", maka akan mengunduh semua foto pasien berdasarkan nomor rekam medis yang telah di-input.	✓		Diterima

### b. Hasil UAT (*User Acceptance Test*)

*User Acceptance Test* terhadap perawat dilaksanakan pada tanggal 27 Januari 2023 secara luring bertempat di Politeknik Kesehatan Jakarta I. Adapun hasil dari UAT yang telah dilaksanakan dapat dilihat pada tabel di bawah ini:

Tabel 4.70 Pengujian Fitur Aplikasi pada Perawat

No.	Daftar Pertanyaan	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Tampilan aplikasi nyaman dan sudah sesuai dengan kebutuhan perawat	✓		Diterima
2	Fitur login sudah sesuai dengan kebutuhan perawat.	✓		Diterima
3	Fitur registrasi / buat akun sudah sesuai dengan kebutuhan perawat.	✓		Diterima
4	Fitur lihat daftar pasien sudah sesuai dengan kebutuhan perawat.	✓		Diterima
5	Fitur lihat detail pasien sudah sesuai dengan kebutuhan perawat.	✓		Diterima
6	Fitur pencarian pasien sudah sesuai dengan kebutuhan perawat.	✓		Diterima
7	Fitur penambahan pasien baru sudah sesuai dengan kebutuhan perawat.	✓		Diterima
8	Fitur lihat histori kajian pasien sudah sesuai dengan kebutuhan perawat.	✓		Diterima
9	Fitur lihat detail kajian pasien sudah sesuai dengan kebutuhan perawat.		✓	Butuh Revisi
10	Fitur tambah hasil kajian baru sudah sesuai dengan kebutuhan perawat		✓	Butuh Revisi
11	Fitur galeri luka pasien sudah sesuai dengan kebutuhan perawat.		✓	Butuh Revisi
12	Fitur profil perawat sudah sesuai dengan kebutuhan perawat.	✓		Diterima
13	Fitur log out sudah sesuai dengan kebutuhan perawat.	✓		Diterima

Berdasarkan tabel yang telah di atas, terdapat tiga fitur yang memerlukan revisi diantaranya:

- Pada halaman detail kajian pasien sebaiknya ditambahkan kolom jenis pembayaran dan tampilkan foto luka sebelumnya agar lebih informatif.
- Pada halaman tambah hasil kajian, perlu difasilitasi fitur untuk menghapus foto anotasi luka dan penyesuaian bahasa kajian yang lebih familiar dengan perawat.
- Pada halaman galeri luka pasien dan halaman detail kajian, foto anotasi tidak perlu ditampilkan, hanya foto luka saja.

### c. Kesimpulan Pengujian

Pengujian aplikasi dilaksanakan dengan dua metode yaitu *unit testing* dan *User Acceptance Test (UAT)*. Berdasarkan uraian di atas, seluruh skenario pada *unit testing* terhadap satu *internal developer* berjalan dengan baik. Namun, berdasarkan hasil UAT terhadap perawat terdapat beberapa masukan seperti penambahan fasilitas hapus foto pada fitur tambah hasil kajian, penambahan kolom jenis pembayaran pada detail pasien, dan menghapus foto anotasi luka pada halaman galeri luka pasien. Dengan pengujian yang telah dilakukan maka dapat disimpulkan bahwa sistem yang dirancang telah lulus uji. Untuk bukti pengujian dilampirkan pada Lampiran 3 dan Lampiran 4.

## BAB V

### KESIMPULAN DAN SARAN

#### A. Kesimpulan

Berdasarkan hasil implementasi dan pengujian aplikasi pada penelitian ini, maka didapat kesimpulan sebagai berikut:

1. Terciptanya prototipe aplikasi pengkajian luka kronis versi pertama berbasis Android yang sudah mengintegrasikan fitur-fitur pada Product Backlog. Adapun perancangannya dilakukan dengan metode Scrum dengan tahapan penyusunan Product Backlog, Sprint Backlog, dan dikerjakan dalam sembilan Sprint.
2. Terimplementasikannya Web Service yang berfungsi sebagai Back-End aplikasi pengkajian luka kronis berbasis Android.
3. Berdasarkan hasil pengujian yang dilakukan terhadap satu anggota tim penelitian, didapatkan bahwa fitur-fitur yang terdapat pada aplikasi berjalan dengan baik dan lulus uji fungsional. Sedangkan berdasarkan hasil *User Acceptance Test* terhadap satu perawat, didapatkan bahwa masih adanya fitur yang belum sesuai dengan kebutuhan perawat dan belum siap untuk tahap pengujian lebih lanjut terhadap lebih dari satu perawat.

#### B. Saran

Adapun beberapa saran untuk penelitian selanjutnya yaitu:

1. Berdasarkan saran perawat saat UAT, sebaiknya ditambahkan perbandingan foto kajian luka saat ini dan foto kajian sebelumnya pada halaman detail kajian.
2. Berdasarkan saran perawat saat UAT, pada halaman tambah hasil kajian, perlu difasilitasi fitur untuk menghapus foto anotasi luka dan penyesuaian bahasa kajian yang lebih familiar dengan perawat.
3. Berdasarkan saran perawat saat UAT, sebaiknya pengujian aplikasi dilakukan secara masif tidak hanya satu perawat saja dan dilakukan pengujian kepuasan user menggunakan skala likert.

4. Berdasarkan saran oleh Scrum Master, sebaiknya terdapat fitur zoom pada saat melakukan anotasi warna luka ataupun anotasi tepi luka.
5. Berdasarkan saran perawat saat UAT, Sebaiknya aplikasi diuji dengan data yang sebenarnya oleh perawat luka di klinik ataupun di rumah sakit.



## DAFTAR PUSTAKA

- Ambler, S. W. (2005). *The Elements of UML 2.0 Style* (1st ed.). Cambridge University Press.
- Aminuddin, M., Sholichin, Sukmana, M., & Nopriyanto, D. (2020). *Modul Perawatan luka* (I. Samsugito (ed.); 1st ed.). CV Gunawana Lestari.
- Arif, T., Sudjarwo, E., Kesehatan, P., & Malang, K. (2020). Hubungan Kadar Glukosa Dengan Tingkat Diabetic Foot Ulcers Berdasarkan Bates-Jensen Wound Assessment-Tools. *Jurnal Ilmiah Media Husada*, 9(2), 60–66. <https://ojs.widyagamahusada.ac.id>
- Bates-Jensen, B. M., McCreath, H., Patlan, A., & Harputlu, D. (2019). Reliability of the Bates-Jensen Wound Assessment Tool (BWAT) for Pressure Injury Assessment: The Pressure Ulcer Detection Study. *HHS Public Access*, 27(4), 386–395. <https://doi.org/10.1111/wrr.12714.Reliability>
- Black, R. (2002). *Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing*. Wiley.
- Cholifah, W., Yulianingsih, Y., & Sagita, S. (2018). Pengujian Black Box Testing Pada Aplikasi Action & Strategy Berbasis Android Dengan Teknologi Phonegap. *Jurnal String*, 3(2), 206-210.
- Chouhan, C., Shrivastava, V., & S Sodhi, P. (2012). Test Case Generation based on Activity Diagram for Mobile Application. *International Journal of Computer Applications*, 57(23), 4–9. <https://doi.org/10.5120/9436-3563>
- Dharwiyanti, S., & Wahono, R. S. (2003). Pengantar Unified Modeling Language (UML). *Kuliah Umum IlmuKomputer.Com*. [https://www.academia.edu/download/50995520/Modul\\_UML.pdf](https://www.academia.edu/download/50995520/Modul_UML.pdf)
- Dienillah, A. F., & Dewi, A. O. . (2018). Upaya Penyelamatan Informasi Melalui Proses Digitalisasi Arsip Akta Kelahiran Di Dinas Kependudukan Dan Pencatatan Sipil Kota Pekalongan. *Jurnal Ilmu Perpustakaan*, 7(3), 131–140. <https://ejournal3.undip.ac.id/index.php/jip/article/view/22926>
- Flask-RESTful. (2020). *Flask-RESTful 0.3.8 documentation*. FlaskRESTful. <https://flask-restful.readthedocs.io/en/latest/>
- Fowler, M., & Scott, K. (2000). UML Distilled Second Edition A Brief Guide to the Standard Object Modeling Language. In *Addison Wesley* (2nd ed.). Addison Wesley. <https://doi.org/10.1128/jcm.31.5.1354-1357.1993>
- Greatrex-White, S., & Moxey, H. (2015). Wound assessment tools and nurses'

- needs: An evaluation study. *International Wound Journal*, 12(3), 293–301. <https://doi.org/10.1111/iwj.12100>
- Hady, E. L., Haryono, K., & Rahayu, N. W. (2020). User Acceptance Testing (UAT) pada purwarupa sistem tabungan santri (studi kasus: Pondok Pesantren Al-Mawaddah). *Jurnal Ilmiah Multimedia dan Komunikasi*, 5(1).
- Hamilton, T. (2022, December 31). *Unit Testing Tutorial – What is, Types & Test Example*. Guru99. <https://www.guru99.com/unit-testing-guide.html>
- Hendy. (2019). Pemodelan Sistem Menggunakan UML (Unified Modelling Language). *ResearchgateGate*.
- Huang, Q., & Dom, B. (1995). Quantitative methods of evaluating image segmentation. Proceedings., International Conference On Image Processing. doi: 10.1109/icip.1995.537578
- Kartika, R. W., Bedah, B., Paru, J., & Luka, A. P. (2015). Perawatan Luka Kronis dengan Modern Dressing. *Perawatan Luka Kronis Dengan Modern Dressing*, 42(7), 546–550.
- KOCAKOVUN, Ş. (2017). Developing of Android Mobile Application Using Java and Eclipse: An Application. *International Journal of Electronics, Mechanical and Mechatronics Engineering*, 7(1), 1335–1354. <https://doi.org/10.17932/iau.ijemme.21460604.2017.7/1.1335-1354>
- Ningrum, S. K. (2019). Implementasi Scrum pada Manajemen Proyek Pengembangan Perangkat Lunak Pemesan Undangan (Studi Kasus: Paperlust). *Automata*. <https://dspace.uii.ac.id/handle/123456789/20869%0Ahttps://journal.uii.ac.id/AUTOMATA/article/view/13905>
- Otair, M. A., & Odat, A. M. (2015). Enhancing an end user development in database design using entity relationship diagram Mapper. *Journal of Theoretical and Applied Information Technology*, 77(2), 218–228.
- Ozierańska, A., Skomra, A., Kuchta, D., & Rola, P. (2016). The critical factors of Scrum implementation in IT project– the case study. *Journal of Economics and Management*, 25(3), 79–96. <https://doi.org/10.22367/jem.2016.25.06>
- Perry, William E. (2006). *Effective Methods for Software Testing 3rd*. Indianapolis, Indiana.

- Pratama, O. N. P. (2020). *SISTEM PAKAR PENGAJIAN LUCA PENYAKIT DIABETES MELLITUS BERDASARKAN INSTRUMEN BATES-JENSEN WOUND ASSESSMENT TOOLS (BJWAT)*) [Universitas Bina Sarana Informatika]. <https://repository.bsi.ac.id/index.php/repo/viewitem/22476>
- Purnama, S. (2013). METODE PENELITIAN DAN PENGEMBANGAN (Pengenalan untuk Mengembangkan Produk Pembelajaran Bahasa Arab). *LITERASI*, 4(1), 19–32.
- Putra, I., Dawood, R., & Roslidar. (2017). Rancang Bangun Layanan Web (Web Service) Untuk Aplikasi Rekam Medis Praktik Pribadi Dokter. *KITEKTRO: Jurnal Online Teknik Elektro*, 2(1), 1–8.
- Pressman, Roger S. (2012). Rekayasa Perangkat Lunak Pendekatan Praktisi (Buku 1). (Alih bahasa : Adi Nugroho). Yogyakarta : Andi Offset
- Rubin, K. S. (2013). *Praktyczny przewodnik po najpopularniejszej metodyce Agile* (1st ed.). Addison Wesley.
- Schwaber, K., & Sutherland, J. (2020). Guia do SCRUM. *Harvard Business Review*, IV, 163–179. [https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum\\_Guide.pdf](https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf)
- Shaikh, A., Hafeez, A., Wagan, A. A., Alrizq, M., Alghamdi, A., & Reshan, M. S. Al. (2021). More Than Two Decades of Research on Verification of UML Class Models: A Systematic Literature Review. *IEEE Access*, 9, 142461–142474. <https://doi.org/10.1109/ACCESS.2021.3121222>
- Siregar, Y. B. (2019). Digitalisasi Arsip Untuk Efisiensi Penyimpanan dan Aksesibilitas. *Jurnal Administrasi Dan Kesekretariatan*, 4(1), 1–19.
- S, Rosa A., & Shalahuddin, M. (2013). Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek. Bandung : Informatika.
- Urva, G., & Siregar, H. F. (2015). Pemodelan UML E- Marketing Minyak Goreng. *Open Access Journal of Information System (OAJIS)*, 2, 92–101.
- York, S. C. of the S. of N. (2018). *SM KIDS against Google LLC* (Issue 1).

## LAMPIRAN

### Lampiran 1 Transkrip Wawancara Pertama

Hari: Kamis

Tanggal: 3 Maret 2022

Lokasi: Daring (Zoom Meeting)

Narasumber: Ns. Ratna Aryani, Dosen di Poltekkes Kemenkes Jakarta I

Peneliti	Selamat siang Bu, terima kasih telah menyempatkan waktunya, saya izin mengajukan beberapa pertanyaan ya Bu.
Narasumber	Ya, silakan.
Peneliti	Mengapa pengkajian warna luka perlu dilakukan?
Narasumber	Pengkajian warna luka itu penting dilakukan untuk menjadi modal dasar perawat menentukan status luka bagaimana serta untuk mengetahui <i>progress</i> penyembuhan luka.
Peneliti	Siapa yang berperan untuk melakukan pengkajian warna luka?
Narasumber	Perawat yang bertanggung jawab terhadap pasien luka. Umumnya dalam satu ruangan terdapat kondisi kasus luka seperti luka tekan, luka kanker, luka akut, dan luka diabet. Perawat yang berada di ruangan tersebut bertanggung jawab untuk melihat kondisi luka sekaligus melakukan perawatan yang dimulai dari pengkajian.
Peneliti	Kapan pengkajian warna luka harus dilakukan?
Narasumber	Pada saat awal pengecekan kondisi luka dan pada saat penggantian balutan.
Peneliti	Bagaimana alur pengkajian warna luka dilakukan?
Narasumber	Balutan luka dibuka, dicuci lukanya agar kotorannya lepas jadi tidak bias, kalau sudah dibersihkan dikeringkan, kemudian

	dilihat warna dasarnya, berapa persen merah, berapa persen kuning berapa persen hitam.
Peneliti	Apa saja yang perlu diperhatikan pada saat pengkajian warna luka?
Narasumber	Yang perlu diperhatikan adalah pencahayaan dan kebersihan luka.
Peneliti	Sistem apa yang saat ini digunakan untuk pengkajian warna luka?
Narasumber	Selama ini masih manual berdasarkan visual perawatnya. Namun saya pernah membuat alat memakai <i>Android</i> yang dapat mengenali warna luka yang kemudian dibandingkan dengan hasil perawatnya.
Peneliti	Apakah saja kendala yang terjadi pada sistem yang telah dipakai saat ini?
Narasumber	Terdapat subjektivitas perawat, hasil dari satu perawat dan perawat lainnya presentasenya berbeda-beda.
Peneliti	Apa pengembangan yang bisa dilakukan dari sistem yang telah Ibu buat?
Narasumber	Pengembangannya lebih ke arah ke peningkatan akuratan, sensitivitas warna dan penambahan <i>item-item</i> pengkajian selain warna dasar luka.

## Lampiran 2 Transkrip Wawancara Kedua

Hari: Sabtu

Tanggal: 19 Maret

2022 Lokasi: Daring

(Zoom Meeting)

Narasumber: Ns. Ratna Aryani, Dosen di Poltekkes Kemenkes Jakarta I

Scrum Master: Muhammad Eka Suryana, M. Kom.

<i>Scrum Master</i>	Assalamualaikum Bu, tampaknya kemarin ada masalah sedikit pada wawancara sebelumnya, saya izin mengonfirmasi beberapa hal ya Bu untuk menentukan arah penelitian kami.
Narasumber	Oh iya-iya boleh Pak Eka.
<i>Scrum Master</i>	Sebelumnya kita sudah membuat aplikasi untuk pengkajian warna luka, nah itu bagaimana cara perawat untuk menentukan tingkat akurasinya?
Narasumber	Untuk menentukan tingkat akurasinya kita membandingkan hasil presentase dari pengkajian menggunakan aplikasi dengan presentase 2 perawat yang tersertifikasi.
<i>Scrum Master</i>	Berapa jumlah data yang diujikan?
Narasumber	Jumlah data yang diujikan sebanyak 100 dengan masing-masing data luka campur 54 data, luka hitam 41 data, luka kuning 42 data dan luka merah 44 data.
<i>Scrum Master</i>	Lalu bagaimana cara perawat menentukan rasio presentase warna luka?
Narasumber	Perawat menentukan dengan cara menduga rasio presentase warna luka dengan mata telanjang, kemudian dibandingkan dari hasil program.

<i>Scrum Master</i>	Jadi begini, sebelumnya penelitian kami ditujukan untuk pengumpulan data anotasi warna luka yang nantinya akan dipakai untuk meningkatkan tingkat akurasi algoritma pengkajian warna luka yang sebelumnya sudah dibuat.
	Namun, apakah mungkin jika perawat dimintai tolong untuk mengarsir atau anotasi warna luka pada aplikasi?
Narasumber	Tidak mungkin, karena pada aslinya warna luka itu tercampur, jadi sulit sekali untuk dibedakan.
<i>Scrum Master</i>	Apa manfaat dari perawat menebak persentase warna luka yang diberikan oleh perawat?
Narasumber	Untuk mengetahui progress perkembangan luka, jadi semakin besar warna merahnya, maka luka mengalami perbaikan, walaupun <i>item</i> warna dasar luka bukan hanya satu-satunya <i>item</i> yang digunakan pada saat pengkajian luka. Jadi nanti juga dapat dilihat dari jumlah eksudatnya, kondisi tepi luka, ada atau tidaknya infeksi, ukuran luka, stadium luka, bau luka, epitelisasi luka, kondisi sekitar luka, dan ada nyeri atau tidak.
<i>Scrum Master</i>	Dari semua item pengkajian, apakah ada data yang dapat diambil berdasarkan gambar? Yang difoto oleh perawat?
Narasumber	Pada kondisi <i>real</i> di lapangan, kita baru memfoto luka setelah luka sudah dicuci, biasanya eksudat yang bersifat <i>purulent</i> sudah bersih, namun tetap ada eksudatnya.
<i>Scrum Master</i>	Kalau dari item pengkajian tepi luka, yang dilihat apanya?
Narasumber	Tepi luka yang normal itu adalah yang halus, bersih. Namun, ada beberapa pengkajian itu menggunakan <i>scoring</i> , jadi misalkan jika tepi lukanya halus dan bersih nanti nilainya 0, kalau tepi lukanya kasar atau kotor nanti nilainya 4.
<i>Scrum Master</i>	Pada <i>item</i> epitelisasi luka, bagaimana cara perawat mencatat progress epitelisasinya?

Narasumber	Kalau beberapa rumah sakit, mereka punya pengkajian luka <i>scoring</i> , jadi tinggal centang-cetang. Jika <i>scoring</i> -nya tinggi maka bisa disimpulkan dia lukanya membaik.
<i>Scrum Master</i>	Saya ingin konfirmasi, biasanya perawat memberi penyimpanan memberi catatan untuk proses pengkajian luka itu melalui aplikasi khusus atau catatan kertas?
Narasumber	Manual menggunakan catatan kertas.
<i>Scrum Master</i>	Kira-kira dibutuhkan atau tidak kalau nantinya ada aplikasi Android yang bisa <i>me-record</i> semua data itu nanti?
Narasumber	Iya, bagus itu.
<i>Scrum Master</i>	Baik, jadi nanti akan dibuat aplikasi yang dapat menyimpan data <i>scoring</i> luka sehingga nanti semua datanya bisa dilihat dari aplikasi itu. Setelah aplikasi itu ada, nanti akan diekstensi dengan penambahan algoritma penelitian sebelumnya. Mungkin untuk awalan tidak semua kategori dimasukkan ke dalam aplikasinya hanya beberapa saja terutama yang memiliki data gambar.
Narasumber	Baik tidak apa-apa. Nanti dalam pembuatannya tolong saya ikut dilibatkan ya.
<i>Scrum Master</i>	Pasti dilibatkan. Terima kasih Bu sudah berkenan memberikan jawaban dan waktunya, Wassalamualaikum wr.wb.
Narasumber	Baik saya mohon undur diri ya.

### Lampiran 3 User Acceptance Test Internal Developer

#### Formulir User Acceptance Test (UAT)

#### Internal Developer

##### A. Pengujian Halaman Awal

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat aplikasi dibuka akan muncul <i>permission request</i> untuk menggunakan kamera dan akses ke storage	✓		
2	Saat <i>permission</i> disetujui maka akan masuk ke tampilan awal, jika tidak, maka tidak dapat masuk ke aplikasi	✓		
3	Jika tombol "LOG IN" ditekan, guest akan masuk ke halaman Login	✓		
4	Jika tombol "Buat Akun" ditekan maka akan masuk ke halaman Buat Akun	✓		

##### B. Pengujian Registrasi User/Perawat

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Ketika mengisi form registrasi dengan lengkap kemudian submit, maka akan masuk ke halaman beranda	✓		
2	Ketika mengisi form registrasi tidak lengkap kemudian submit, maka akan menampilkan pesan kesalahan	✓		

##### C. Pengujian Login User

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Ketika mengisi form login dengan data yang sesuai kemudian klik submit, maka akan masuk ke halaman beranda	✓		
2	Ketika mengisi form login dengan data yang tidak sesuai kemudian klik submit, maka akan menampilkan pesan kesalahan	✓		

D. Pengujian Halaman Beranda

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat ikon profil pada kanan atas di-klik maka akan muncul halaman profil perawat	✓		
2	Saat tombol daftar pasien di-klik maka akan muncul halaman daftar pasien	✓		
3	Saat tombol galeri foto luka di-klik maka akan muncul halaman galeri foto luka	✓		
4	Saat tombol arsir warna luka di-klik maka akan muncul halaman arsir warna luka	✓		

E. Pengujian Halaman Daftar Pasien

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat ikon panah pada kiri atas diklik akan kembali ke halaman beranda	✓		
2	Saat ikon lup pada kanan atas diklik maka akan muncul isian untuk mencari pasien	✓		
3	Saat mencari pasien dengan <i>input</i> nama yang terdapat pada database, maka akan muncul nama pasien tersebut	✓		
4	Saat mencari pasien dengan <i>input</i> nama yang tidak terdapat pada database, maka akan muncul pesan bahwa tidak ditemukan	✓		
5	Saat mencari pasien dengan <i>input</i> nomor rekam medis pasien yang terdapat pada database, maka akan muncul nama pasien sesuai dengan nomor registrasi yang telah di- <i>input</i>	✓		
6	Saat mencari pasien dengan <i>input</i> nomor rekam medis pasien yang tidak terdapat pada database, maka akan muncul pesan bahwa tidak ditemukan	✓		
7	Saat menekan tombol Laki-laki maka akan muncul seluruh pasien laki-laki	✓		
8	Saat menekan tombol Perempuan maka akan muncul seluruh pasien perempuan	✓		
9	Saat menekan tombol tambah pasien baru pada kanan bawah, maka akan muncul halaman penambahan pasien baru	✓		
10	Saat klik nama pasien, maka akan muncul halaman detail pasien	✓		

F. Pengujian Tambah Pasien Baru

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Ketika mengisi form pasien dengan lengkap kemudian klik submit, maka akan masuk ke halaman detail pasien yang baru saja diinput	✓		
2	Ketika mengisi form pasien tidak lengkap kemudian submit, maka akan menampilkan pesan kesalahan	✓		

G. Pengujian Halaman Detail Pasien

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat masuk pertama kali ke halaman detail pasien, maka informasi umum pasien terlihat	✓		
2	Saat klik ikon pensil maka akan masuk ke halaman <i>edit</i> data pasien	✓		
3	Saat <i>tab</i> Histori Kajian diklik maka akan muncul histori kajian pasien	✓		
4	Saat <i>tab</i> Galeri Luka diklik maka akan muncul seluruh foto luka milik pasien	✓		
5	Saat tombol Raw pada <i>tab</i> Galeri Luka diklik, maka akan muncul semua foto luka tanpa anotasi dengan format JPG.	✓		
6	Saat tombol Anotasi Tepi pada <i>tab</i> Galeri Luka diklik, maka akan muncul semua foto luka beranotasi tepi dengan format JPG.	✓		
7	Saat tombol Anotasi Diameter pada <i>tab</i> Galeri Luka diklik, maka akan muncul semua foto luka beranotasi diameter dengan format JPG.	✓		
8	Saat klik tombol "Tambah Hasil Kajian" pada <i>tab</i> Histori Kajian, maka akan diarahkan ke halaman tambah hasil kajian.	✓		
9	Saat klik tombol tanggal yang terdapat pada <i>tab</i> Histori Kajian, maka akan diarahkan menuju halaman detail kajian luka sesuai dengan tanggal yang dipilih.	✓		
10	Saat klik ikon pensil pada sebelah kanan teks "Informasi Umum Pasien" maka akan diarahkan menuju halaman Edit Data Pasien	✓		

#### H. Pengujian Halaman Edit Data Pasien

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat pertama kali masuk ke halaman Edit Data Pasien maka akan ditampilkan kembali informasi umum yang telah ada serta ikon pensil di sebelah kanannya	✓		
2	Saat klik ikon pensil, maka akan ditampilkan pesan perintah, kolom isian, dan tombol <i>submit</i>	✓		
3	Ketika tidak mengisi kolom kemudian klik submit, maka akan ditampilkan pesan error	✓		
4	Ketika mengisi kolom dengan lengkap kemudian klik submit, maka akan masuk ke halaman Edit Data Pasien dengan data yang sudah diperbarui	✓		

#### I. Pengujian Tambah Hasil Kajian

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat pertama kali masuk ke halaman tambah hasil kajian baru hanya terdapat nama pasien, nomor rekam medis, jenis kelamin, usia, dan tombol kamera.	✓		
2	Ketika pengguna belum mengambil foto luka maka tidak akan muncul form kajian luka.	✓		
3	Ketika pengguna mengklik tombol kamera, maka akan masuk ke halaman pengambilan foto luka.	✓		
4	Ketika pengguna selesai mengambil foto luka, maka akan diarahkan ke halaman anotasi tepi luka.	✓		
5	Ketika pengguna selesai menganotasi tepi luka dan klik tombol "SAVE" maka akan diarahkan untuk manganotasi diameter koordinat X.	✓		
6	Ketika pengguna selesai manganotasi diameter luka koordinat X dan klik tombol "SAVE" maka diarahkan untuk manganotasi diameter koordinat Y.	✓		
7	Ketika pengguna selesai manganotasi diameter luka koordinat X dan klik tombol "SAVE" maka diarahkan untuk manganotasi diameter koordinat Y.	✓		
8	Ketika pengguna selesai manganotasi diameter luka koordinat Y dan klik tombol "SAVE" maka diarahkan untuk mengisi form kajian luka.	✓		
9	Ketika mengisi form kajian dengan lengkap kemudian klik submit, maka akan masuk ke halaman detail pasien yang baru saja diinput.	✓		

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
10	Ketika mengisi form kajian secara tidak lengkap kemudian klik submit, maka akan masuk ke halaman detail pasien yang baru saja diinput.	✓		

J. Pengujian Halaman Detail Kajian

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat masuk pertama kali ke halaman detail kajian, maka akan ditampilkan informasi tanggal kajian,nama pasien, perawat yang menangani, NIP perawat, foto luka, anotasi tepi luka, anotasi diameter luka, skoring kajian luka, dan ikon pensil untuk mengedit data	✓		
2	Saat klik ikon pensil di atas foto luka atau foto anotasi, maka akan masuk ke halaman untuk melakukan foto ulang atau anotasi ulang	✓		
3	Saat klik ikon pensil pada sisi kanan tulisan “Skoring Kajian Luka” maka akan masuk ke halaman Edit Skoring Kajian	✓		
4	Saat klik ikon panah pada kiri atas, maka akan kembali ke halaman detail pasien.	✓		

K. Pengujian Halaman Edit Skoring Kajian

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat pertama kali masuk ke halaman Edit Skoring Kajian maka akan ditampilkan kembali informasi skoring yang telah ada serta ikon pensil di sebelah kanannya	✓		
2	Saat klik ikon pensil, maka akan ditampilkan pesan perintah, kolom isian, dan tombol <i>submit</i>	✓		
3	Ketika tidak mengisi kolom kemudian klik submit, maka akan ditampilkan pesan error	✓		
4	Ketika mengisi kolom dengan lengkap kemudian klik submit, maka akan masuk ke halaman Edit Skoring Kajian dengan data yang sudah diperbarui	✓		

L. Pengujian Halaman Detail Foto

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat pertama kali masuk ke halaman detail foto, maka akan ditampilkan foto tersebut (kecuali yang berformat SVG), informasi filename, informasi nomor rekam medis pasien, informasi NIP perawat yang mengambil foto, tanggal unggah, dan kategori.	✓		

M. Pengujian Arsir Warna Luka

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat klik ikon galeri biru yang terdapat saat pertama kali membuka fitur arsir warna luka, maka pengguna diarahkan untuk mengunggah foto dari galeri.	✓		
2	Ketika pengguna klik tombol panah pada kiri atas, maka akan diarahkan kembali ke halaman sebelumnya.	✓		
3	Pada saat pengguna menggambar, warna yang pertama kali dipakai adalah warna hitam.	✓		
4	Ketika tombol "Red" diklik maka warna <i>stroke</i> akan berubah menjadi warna merah.	✓		
5	Ketika tombol "Yellow" diklik maka warna <i>stroke</i> akan berubah menjadi warna kuning.	✓		
6	Ketika tombol "Black" diklik maka warna <i>stroke</i> akan berubah menjadi warna hitam.	✓		
7	Ketika tombol "Undo" diklik maka stroke yang baru saja digambar akan hilang.	✓		
8	Ketika ikon bulat biru digeser ke kanan, maka <i>stroke</i> akan semakin lebar.	✓		
9	Ketika selesai mengarsir warna luka dan klik tombol save, maka foto akan diunggah ke server lalu kembali ke halaman beranda.	✓		

N. Pengujian Halaman Profil Perawat

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat pertama kali masuk ke halaman Profil Perawat maka akan ditampilkan nama perawat, NIP perawat, email perawat, tombol <i>log out</i> , dan ikon pensil	✓		
2	Ketika tombol "LOG OUT" diklik, maka pengguna akan kembali ke halaman awal dan tidak bisa mengakses fitur pada aplikasi.	✓		
3	Ketika ikon pensil diklik maka akan diarahkan ke halaman edit data perawat.	✓		

O. Pengujian Fitur Edit Perawat

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat pertama kali masuk ke halaman Edit Profil Perawat maka akan ditampilkan kalimat perintah, kolom teks untuk mengedit, dan tombol <i>submit</i> .	✓		
2	Ketika mengisi kolom dengan lengkap kemudian klik submit, maka akan masuk ke halaman Profil Pasien	✓		
3	Ketika tidak mengisi kolom kemudian klik submit, maka akan ditampilkan pesan error	✓		

P. Pengujian Fitur Log User

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat mengunjungi link <a href="http://jft.web.id/woundapi/get_logs">http://jft.web.id/woundapi/get_logs</a> maka akan ditampilkan histori logging seluruh pengguna.	✓		
2	Ketika tombol "LOG OUT" diklik, maka pengguna akan kembali ke halaman awal dan tidak bisa mengakses fitur pada aplikasi.	✓		

Q. Pengujian Fitur Unduh Dataset Luka

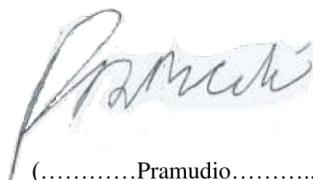
No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Saat mengunjungi link <a href="http://jft.web.id/woundapi/">http://jft.web.id/woundapi/</a> maka akan ditampilkan halaman untuk mengunduh dataset foto luka kronis.	✓		
2	Saat tombol "Unduh Semua Foto" diklik, maka akan mengunduh semua foto luka kronis tanpa terkecuali.	✓		
3	Saat tombol "Unduh" pada card Raw Image diklik, maka akan mengunduh semua foto luka kronis tanpa anotasi.	✓		
4	Saat tombol "Unduh" pada card Anotasi Tepi diklik, maka akan mengunduh semua foto anotasi tepi luka kronis.	✓		
5	Saat tombol "Unduh" pada card Anotasi Diameter diklik, maka akan mengunduh semua foto anotasi diameter luka kronis.	✓		
6	Setelah mengisikan nomor rekam medis pada form yang tersedia lalu menekan tombol "Unduh", maka akan mengunduh semua foto pasien berdasarkan nomor rekam medis yang telah di-input.	✓		

Saya yang bertanda tangan di bawah ini, menyatakan bahwa pada hari Kamis tanggal 26 Januari 2023 telah dilakukan pengujian User Acceptance Test dengan *requirement* pengujian sebagai berikut:

1. Fitur aplikasi secara keseluruhan.
2. Fitur untuk mengecek *log user*.
3. Fitur untuk mengunduh seluruh dataset foto luka kronis.

Dengan hasil (~~tidak lulus uji~~ / lulus uji)\*

Jakarta, 26 Januari 2023



\*Coret yang tidak perlu.

(.....Pramudio.....)

## Lampiran 4 User Acceptance Test Perawat

Formulir User Acceptance Test (UAT)  
Untuk Perawat

No.	Skenario Pengujian	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Tampilan aplikasi nyaman dan sudah sesuai dengan kebutuhan perawat	✓		
2	Fitur login sudah sesuai dengan kebutuhan perawat.	✓		
3	Fitur registrasi / buat akun sudah sesuai dengan kebutuhan perawat.	✓		
4	Fitur lihat daftar pasien sudah sesuai dengan kebutuhan perawat.	✓		
5	Fitur lihat detail pasien sudah sesuai dengan kebutuhan perawat.	✓		
6	Fitur pencarian pasien sudah sesuai dengan kebutuhan perawat.	✓		
7	Fitur penambahan pasien baru sudah sesuai dengan kebutuhan perawat.	✓		
8	Fitur lihat histori kajian pasien sudah sesuai dengan kebutuhan perawat.	✓		
9	Fitur lihat detail kajian pasien sudah sesuai dengan kebutuhan perawat.	✗	✓ bpjs	
10	Fitur tambah hasil kajian baru sudah sesuai dengan kebutuhan perawat	✓		
11	Fitur galeri luka pasien sudah sesuai dengan kebutuhan perawat.		✓ fasilitas delete	
✗	Fitur galeri foto luka sudah sesuai dengan kebutuhan perawat.		✓	
13	Fitur arsir warna luka sudah sesuai dengan kebutuhan perawat.			belum bisa diindeks
14	Fitur profil perawat sudah sesuai dengan kebutuhan perawat.	✓		
15	Fitur log out sudah sesuai dengan kebutuhan perawat.	✓		

Berdasarkan data uji coba di atas, apakah aplikasi sudah layak untuk diujicobakan ke perawat secara masif? (✓/ Butuh Revisi) \*

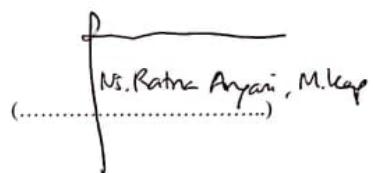
Revisi:  
1) jenis pembayaran (detail pasien) , bahasa scoring kurang familiar,  
foto tanpa anotasi , ditambahkan pertanda tangan sebelumnya , delete  
foto anotasi ,

\*Coret yang tidak perlu

Saya yang bertanda tangan di bawah ini, menyatakan bahwa pada hari Jumat tanggal  
27/01/2023 telah dilakukan pengujian User Acceptance Test fitur aplikasi secara keseluruhan,  
dengan hasil (tidak lulus uji / lulus uji)\*

Jakarta, 27 Januari 2023

\*Coret yang tidak perlu.

  
Ms. Ratna Anyani, M.Kap  
(.....)

## DAFTAR RIWAYAT HIDUP



**SALSA RAHMADATI**, Lahir di Surakarta, 29 Desember 1999. Anak kedua dari pasangan Bapak Alm. Mujono dan Ibu Maryani. Saat ini penulis tinggal di Perumahan Poncol Jaya Blok C3 Nomor 13 RT 003/019 Kelurahan Jaka Sampurna, Keamatan Bekasi Barat, Kota Bekasi, Provinsi Jawa Barat.

**Riwayat Hidup:** Penulis mengawali pendidikan di TK Islam Al-Hikmah pada tahun 2005-2006. Kemudian melanjutkan pendidikan di SDN Malaka Jaya 07 Pagi pada tahun 2006-2011. Selanjutnya penulis melanjutkan ke SMPN 252 Jakarta pada tahun 2011-2014. Kemudian melanjutkan pendidikan di SMAN 44 Jakarta pada tahun 2014-2017. Pada tahun 2017 penulis melanjutkan ke Universitas Negeri Jakarta (UNJ) di program studi Ilmu Komputer melalui jalur SNMPTN.

**Riwayat Organisaasi:** Selama di bangku perkuliahan, penulis merupakan anggota AIESEC, BEM Prodi Ilmu Komputer, dan DEFAULT UNJ. Penulis juga berpartisipasi dalam kegiatan COMPARE (Computer Academic Care) yaitu kegiatan workshop dan seminar yang diadakan oleh BEM (Badan Eksekutif Mahasiswa), penulis tergabung sebagai ketua pelaksana acara dan kegiatan BINER yaitu kegiatan seminar teknologi yang diadakan oleh DEFAULT UNJ, dimana penulis tergabung sebagai seksi desain dan dokumentasi.

**Riwayat Prestasi:** Pada saat duduk di bangku SMA, penulis sering mengikuti kegiatan perlombaan paduan suara diikuti dengan perolehan juara II lomba paduan suara tingkat provinsi DKI Jakarta. Selain itu, penulis juga pernah meraih piala perunggu dalam kejuaraan taekwondo yang diselenggarakan oleh Valentino Team pada tahun 2013.