

**APLIKASI PENGKAJIAN LUKA BERBASIS ANDROID
TERINTEGRASI DENGAN SISTEM INFORMASI KLINIK
KEPERAWATAN LUKA**

Skripsi

**Disusun untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer**



Intelligentia - Dignitas

Oleh:
Muhammad Ardani
1313618014

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

2025

LEMBAR PENGESAHAN

Dengan ini saya mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam,
Universitas Negeri Jakarta

Nama : MUHAMMAD ARDANI
No. Registrasi : 1313618014
Jurusan : Ilmu Komputer
Judul : APLIKASI PENGKAJIAN LUKA BERBASIS
ANDROID TERINTEGRASI DENGAN SISTEM
INFORMASI KLINIK KEPERAWATAN LUKA

Menyatakan bahwa skripsi ini telah siap diajukan.

Menyetujui,

Dosen Pembimbing I

Dosen Pembimbing II



Muhammad Eka Suryana,
M.Kom.

NIP. 19851223 201212 1 002

Drs. Mulyono, M. Kom.

NIP. 19660517 199403 1 003

Mengetahui
Koordinator Program Studi Ilmu Komputer



Dr. Ria Arafiyah, M. Si.

NIP. 19751121 200501 2 004

LEMBAR PERNYATAAN

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul “Aplikasi Pengkajian Luka Berbasis Android Terintegrasi Dengan Sistem Informasi Klinik Keperawatan Luka” yang disusun sebagai syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Ilmu Komputer Universitas Negeri Jakarta adalah karya ilmiah saya dengan arahan dari dosen pembimbing. Sumber informasi yang diperoleh dari penulis lain yang telah dipublikasikan dan disebutkan dalam teks skripsi ini, telah dicantumkan dalam Daftar Pustaka sesuai dengan norma, kaidah dan etika penulisan ilmiah. Jika dikemudian hari ditemukan sebagian besar skripsi ini bukan hasil karya saya sendiri dalam bagian bagian tertentu, saya bersedia menerima sanksi pencabutan gelar akademik yang saya sanding dan sanksi-sanksi lainnya sesuai dengan peraturan perundang-undangan yang berlaku.

Jakarta, 07 Februari 2025

Muhammad Ardani
1313618014

HALAMAN PERSEMBAHAN

Untuk Keluargaku dan Diriku Sendiri.

ABSTRAK

MUHAMMAD ARDANI. Aplikasi Pengkajian Luka Berbasis Android Terintegrasi Dengan Sistem Informasi Klinik Keperawatan Luka. Skripsi. Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta. 2025. Di bawah bimbingan Muhammad Eka Suryana, M.Kom dan Drs. Mulyono, M.Kom.

Penyebab luka kronis dan faktor yang mempengaruhi penyembuhan luka cukup kompleks dan beragam. Dalam proses diagnosis, pengobatan dan manajemen luka kronis, dokter dan perawat perlu memiliki pemahaman menyeluruh serta kelengkapan rekam medis, manajemen yang efektif dan pemantauan kondisi fisik umum pasien secara tepat waktu, laporan pemeriksaan laboratorium, penilaian dan pengobatan luka fraksional. Secara umum, proses pengkajian luka masih dilakukan secara tradisional dengan metode pencatatan atau pengarsipan berbasis kertas. Saat ini, data-data tersebut masih dicatat secara manual, yang berisiko menimbulkan kesalahan dalam perhitungan. Selain itu, penelitian sebelumnya memiliki kekurangan penilaian data pengkajian luka kronis secara akurat. Sehingga diperlukannya penelitian ini dengan tujuan untuk membuat aplikasi pengkajian luka kronis dengan modul *Bates-Jensen Assessment Wound Tools(BWAT)* berbasis *Android*. Data kajian yang digunakan dalam penelitian ini terfokus pada proses pencatatan pasien berobat penanganan luka kronis. Proses pengembangan sistem ini menggunakan metode *Scrum* dan seluruh aplikasi yang dibuat menggunakan bahasa pemrograman *Kotlin* dan diintegrasikan dengan penelitian-penelitian sebelumnya. Hasil akhir dari penelitian ini berupa aplikasi pengkajian luka yang membantu perawat dalam melakukan pencatatan proses penyembuhan luka pasien dan pasien dapat melihat progres penyembuhan dirinya sendiri.

Kata kunci: *android, aplikasi, luka kronis, BWAT, scrum, pencatatan, integrasi*

ABSTRACT

MUHAMMAD ARDANI. Android-Based Wound Assessment Application Integrated with the Wound Care Clinic Information System. Thesis. Computer Science Study Program, Faculty of Mathematics and Natural Sciences, Universitas Negeri Jakarta. 2025. Supervised by Muhammad Eka Suryana, M.Kom, and Drs. Mulyono, M.Kom.

The causes of chronic wounds and the factors influencing wound healing are quite complex and diverse. In the process of diagnosing, treating, and managing chronic wounds, doctors and nurses need to have a comprehensive understanding, complete medical records, effective management, and timely monitoring of the patient's general physical condition, laboratory examination reports, wound assessment, and fractional wound treatment. In general, wound assessment is still carried out traditionally using paper-based recording or archiving methods. Currently, this data is still recorded manually, which poses a risk of calculation errors. Moreover, previous research lacked an accurate assessment of chronic wound evaluation data. Therefore, this study aims to develop a chronic wound assessment application with the Bates-Jensen Assessment Wound Tools (BWAT) module based on Android. The assessment data used in this study focuses on recording patients undergoing chronic wound treatment. The system development process follows the Scrum methodology, and the entire application is built using the Kotlin programming language while integrating findings from previous research. The final result of this study is a wound assessment application that assists nurses in documenting the wound healing process and enables patients to track their own healing progress.

Key: android, application, chronic wounds, BWAT, Scrum, documentation, integration

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT, karena dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan skripsi yang berjudul "**Aplikasi Pengkajian Luka Berbasis Android Terintegrasi Dengan Sistem Informasi Klinik Keperawatan Luka**".

Keberhasilan dalam menyusun skripsi ini tidak lepas dari bantuan berbagai pihak yang mana dengan tulus dan ikhlas memberikan masukan guna sempurnanya skripsi ini. Oleh karena itu dalam kesempatan ini, dengan kerendahan hati penulis mengucapkan banyak terima kasih kepada:

1. Yth. Para petinggi di lingkungan FMIPA Universitas Negeri Jakarta.
2. Yth. Ibu Dr. Ria Arafiyah, M. Si selaku Koordinator Program Studi Ilmu Komputer.
3. Yth. Bapak Muhammad Eka Suryana, M.Kom selaku Dosen Pembimbing I yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap skripsi ini.
4. Yth. Bapak Drs. Mulyono, M.Kom selaku Dosen Pembimbing II yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap skripsi ini.
5. Ibu Ns. Ratna Aryani, M.Kep yang senantiasa berpartisipasi dalam pembuatan aplikasi dalam skripsi saya.
6. Orang tua penulis yang selama ini telah mendukung dan membantu menyelesaikan penggerjaan skripsi ini.
7. Teman-teman Program Studi Ilmu Komputer 2018 yang telah mendukung dan membantu penggerjaan skripsi ini.
8. Teman-teman dari Skripsi Victory yang turut mendukung penulis dalam penggerjaan skripsi ini.
9. Teman-teman dari grup mabar Valorant yang senantiasa hadir untuk mendukung penulis dalam proses penggerjaan skripsi ini.

10. Teman-teman dari grup Anti Scanning yang selalu menemani penulis dalam penggerjaan skripsi ini.

Penulis menyadari bahwa penyusunan skripsi ini masih jauh dari sempurna karena keterbatasan ilmu dan pengalaman yang dimiliki. Oleh karenanya, kritik dan saran yang bersifat membangun akan penulis terima dengan senang hati. Akhir kata, penulis berharap tugas akhir ini bermanfaat bagi semua pihak khususnya penulis sendiri. Semoga Allah SWT senantiasa membala kebaikan semua pihak yang telah membantu penulis dalam menyelesaikan skripsi ini.

Jakarta, 13 Juni 2023

Muhammad Ardani

DAFTAR ISI

LEMBAR PENGESAHAN	iii
LEMBAR PERNYATAAN	iv
HALAMAN PERSEMBAHAN	v
ABSTRAK	vi
ABSTRACT	vii
KATA PENGANTAR	ix
DAFTAR ISI	xi
DAFTAR GAMBAR	xv
DAFTAR TABEL	xvii
I PENDAHULUAN	1
A. Latar Belakang Masalah	1
B. Rumusan Masalah	4
C. Pembatasan Masalah	4
D. Tujuan Penelitian	4
E. Manfaat Penelitian	4
II KAJIAN PUSTAKA	6
A. Pengertian <i>Bates-Jensen Wound Assessment Tool</i>	6
B. Kegunaan <i>Bates-Jensen Wound Assessment Tool</i>	6
C. Penggunaan Skor BWAT untuk Mengidentifikasi Status Keparahan dan Panduan Perawatan	7
D. <i>Activity vs Fragment</i>	16
E. <i>Navigation</i> dan <i>menu</i>	21
F. <i>Layouting</i>	27
G. <i>Retrofit</i>	31
H. <i>Manajemen Keperawatan</i>	36

I. <i>Scrum</i>	39
III METODOLOGI PENELITIAN	40
A. Deskripsi Sistem	40
B. Analisa Kebutuhan	41
C. Perancangan Sistem	43
D. Pengujian	45
IV HASIL DAN PEMBAHASAN	52
A. Pembahasan	52
1. <i>Sprint 1</i>	52
2. <i>Sprint 2</i>	60
3. <i>Sprint 3</i>	65
4. <i>Sprint 4</i>	68
5. <i>Sprint 5</i>	72
6. <i>Sprint 6</i>	77
7. <i>Sprint 7</i>	79
8. <i>Sprint 8</i>	86
9. <i>Sprint 9</i>	89
B. Hasil Pengujian	92
1. Unit Testing	92
2. User Acceptance Testing	101
3. Kesimpulan Pengujian	103
V KESIMPULAN DAN SARAN	104
A. Kesimpulan	104
B. Saran	104
DAFTAR PUSTAKA	106
LAMPIRAN	106
LAMPIRAN	107
A Transkrip Percakapan	107
B Bates-Jensen Wound Assessment Tools	108

C Format Pengkajian Luka Klinik Moist Care	112
RIWAYAT HIDUP	119

DAFTAR GAMBAR

Gambar 2.1	Halaman dari panduan bergambar penilaian luka <i>Bates-Jensen</i> (Sussman dan Bates-Jensen, 2012)	7
Gambar 2.2	Status keparahan berdasarkan skor BWAT. Tujuan terapi adalah (1) untuk mengurangi tingkat keparahan luka secara keseluruhan dan, dengan demikian, skor BWAT dan (2) untuk membuat penurunan secara tepat waktu. Ada perhatian yang sama mengenai tingkat keparahan luka dan durasi waktu yang dihabiskan luka dalam kondisi parah apa pun. (Sussman dan Bates-Jensen, 2012)	8
Gambar 2.3	Algoritma perawatan skor keparahan BWAT minimal untuk luka ringan, kering, dengan ketebalan sebagian. (Diadaptasi dari ConvaTec, dengan izin.) (Sussman dan Bates-Jensen, 2012)	11
Gambar 2.4	Parsial-ketebalan luka dengan algoritma pengobatan skor keparahan BWAT ringan. (Diadaptasi dari ConvaTec, dengan izin.) (Sussman dan Bates-Jensen, 2012)	12
Gambar 2.5	Ulkus tekan stadium III atau stadium IV dengan ketebalan penuh dengan algoritme pengobatan skor keparahan BWAT sedang. (Diadaptasi dari ConvaTec, dengan izin.) (Sussman dan Bates-Jensen, 2012)	13
Gambar 2.6	Luka dengan ketebalan penuh umum dengan algoritme perawatan skor keparahan BWAT kritis. (Dicetak ulang dari ConvaTec, dengan izin.) (Sussman dan Bates-Jensen, 2012) . .	14
Gambar 2.7	Luka dengan ketebalan penuh umum dengan <i>undermining</i> atau <i>pocketing</i> dengan algoritme perawatan skor keparahan BWAT sedang. (Dicetak ulang dari ConvaTec, dengan izin.) (Sussman dan Bates-Jensen, 2012)	15
Gambar 2.8	Skor keparahan BWAT kritis dengan algoritme perawatan eschar kering. (Dicetak ulang dari ConvaTec, dengan izin.) (Sussman dan Bates-Jensen, 2012)	16
Gambar 2.9	Contoh menambahkan repositori <i>Google Maven</i> pada <i>dependency</i> (DeveloperAndroid, 2023)	17

Gambar 2.10 Contoh menambahkan <i>AndroidX Fragment</i> (DeveloperAndroid, 2023)	17
Gambar 2.11 Contoh berikan <i>resource layout fragment</i> ke konstruktor dasar (DeveloperAndroid, 2023)	18
Gambar 2.12 Contoh <i>FragmentContainerView</i> pada <i>XML</i> (DeveloperAndroid, 2023)	19
Gambar 2.13 Contoh menambahkan <i>fragment</i> secara terprogram. (DeveloperAndroid, 2023)	19
Gambar 2.14 Contoh membuat <i>FragmentTransaction</i> . (DeveloperAndroid, 2023)	20
Gambar 2.15 Contoh menambahkan beberapa data awal. (DeveloperAndroid, 2023)	20
Gambar 2.16 Contoh memanggil <i>requireArgumenet()</i> . (DeveloperAndroid, 2023)	21
Gambar 2.17 Contoh dependensi untuk <i>navigation</i> (DeveloperAndroid, 2023)	22
Gambar 2.18 Grafik navigasi yang menampilkan pratinjau dari enam tujuan berbeda yang terhubung melalui lima tindakan (DeveloperAndroid, 2023)	23
Gambar 2.19 <i>Navigation Editor</i> (DeveloperAndroid, 2023)	24
Gambar 2.20 Contoh <i>XML navigation graph</i> (DeveloperAndroid, 2023)	25
Gambar 2.21 Contoh <i>menu</i> (DeveloperAndroid, 2023)	26
Gambar 2.22 Ilustrasi hierarki tampilan, yang menentukan tata letak UI (DeveloperAndroid, 2023)	27
Gambar 2.23 Contoh <i>layout XML</i> yang menggunakan <i>LinearLayout</i> (DeveloperAndroid, 2023)	28
Gambar 2.24 Contoh menggunakan <i>setContentView()</i> (DeveloperAndroid, 2023)	29
Gambar 2.25 Contoh sintaks untuk <i>tag ID</i> (DeveloperAndroid, 2023)	29
Gambar 2.26 Contoh menggunakan <i>namespace package</i> Android (DeveloperAndroid, 2023)	30
Gambar 2.27 Menetapkan <i>ID</i> unik (DeveloperAndroid, 2023)	30
Gambar 2.28 Membuat <i>instance objek view</i> (DeveloperAndroid, 2023)	30
Gambar 2.29 Contoh <i>retrofit</i> mengubah API menjadi <i>interface</i> (Square, 2023)	31

Gambar 2.30 Kelas <i>retrofit</i> (Square, 2023)	31
Gambar 2.31 <i>Call</i> (Square, 2023)	31
Gambar 2.32 Contoh request GET (Square, 2023)	32
Gambar 2.33 Contoh request GET dengan menentukan parameter kueri di URL (Square, 2023)	32
Gambar 2.34 Contoh request URL secara dinamis (Square, 2023)	32
Gambar 2.35 Contoh parameter kueri ditambahkan (Square, 2023)	32
Gambar 2.36 Contoh menggunakan Map (Square, 2023)	32
Gambar 2.37 Contoh anotasi Body (Square, 2023)	33
Gambar 2.38 Contoh dikode (Square, 2023)	33
Gambar 2.39 Contoh multipart (Square, 2023)	33
Gambar 2.40 Contoh anotasi header (Square, 2023)	34
Gambar 2.41 Contoh anotasi header (Square, 2023)	34
Gambar 2.42 Contoh header dinamis (Square, 2023)	34
Gambar 2.43 Contoh header kompleks menggunakan Map (Square, 2023) .	34
Gambar 2.44 Contoh kelas GsonConverterFactory (DeveloperAndroid, 2023)	35
Gambar 2.45 Contoh mendownload retrofit pada Gradle (DeveloperAndroid, 2023)	36
Gambar 2.46 Flowchart manajemen keperawatan	39
Gambar 3.1 Tahapan Penelitian	41
Gambar 3.2 Use Case	42
Gambar 3.3 Tahapan Pengembangan Metode Scrum	43
Gambar 4.1 <i>Fork repo frontend dan backend</i> penelitian sebelumnya	54
Gambar 4.2 <i>Mock-up skoring</i> pengkajian luka lanjutan	54
Gambar 4.3 Class Diagram Activity	55
Gambar 4.4 Class Diagram Fragment	56
Gambar 4.5 <i>Navigation Graph</i>	57
Gambar 4.6 <i>Entity Relationship Diagram</i>	59
Gambar 4.7 <i>Dashboard User</i>	60
Gambar 4.8 <i>Mock-up Pemeriksaan Kesehatan</i>	61
Gambar 4.9 <i>Class Diagram Sprint 2</i>	62
Gambar 4.10 Fragment Pemeriksaan Kesehatan	64
Gambar 4.11 <i>ERD Sprint-3</i>	67

Gambar 4.12 <i>Mock-up</i> Rekap Kunjungan	67
Gambar 4.13 <i>ERD Sprint-4</i>	70
Gambar 4.14 <i>UI Design</i> Tujuan Perawatan	72
Gambar 4.15 Fragment Tujuan Perawatan	72
Gambar 4.16 Database <i>Sprint-5</i>	76
Gambar 4.17 <i>ERD Sprint-5</i>	77
Gambar 4.18 <i>Entity Relation Diagram</i> Sistem Keseluruhan	78
Gambar 4.19 <i>Flowchart Wound History</i>	78
Gambar 4.20 <i>Flowchart Wound History</i>	81
Gambar 4.21 Desain tampilan registrasi pasien	82
Gambar 4.22 Fragment registrasi pasien	83
Gambar 4.23 Fragment registrasi pasien	84
Gambar 4.24 Tampilan riwayat luka	85
Gambar 4.25 Tampilan inventaris	87
Gambar 4.26 <i>ERD Sprint-8</i>	87
Gambar 4.27 <i>UI Design Sprint-9</i>	90
Gambar 4.28 Fragment daftar pasien	92

DAFTAR TABEL

Tabel 1.1	Diabetes Mellitus DALYs (<i>disability-adjusted life year</i>) secara global (WorldHealthOrganization, 2022)	1
Tabel 3.1	<i>Product Backlog</i>	44
Tabel 3.2	Skenario <i>Unit Testing</i>	46
Tabel 3.3	Skenario <i>Unit Acceptance Testing</i>	49
Tabel 4.1	<i>Sprint 1</i>	53
Tabel 4.2	<i>Sprint 2</i>	60
Tabel 4.3	<i>Sprint 3</i>	66
Tabel 4.4	<i>Sprint 4</i>	69
Tabel 4.5	<i>Sprint 5</i>	73
Tabel 4.6	<i>Sprint 6</i>	77
Tabel 4.7	<i>Sprint 7</i>	80
Tabel 4.8	<i>Sprint 8</i>	86
Tabel 4.9	<i>Sprint 9</i>	90
Tabel 4.10	Unit Testing Halaman Awal.	93
Tabel 4.11	Unit Testing Halaman Buat Akun.	93
Tabel 4.12	Unit Testing Halaman Buat Akun Pasien.	94
Tabel 4.13	Unit Testing Halaman Login.	94
Tabel 4.14	Unit Testing Beranda Perawat.	94
Tabel 4.15	Unit Testing Profil Perawat.	95
Tabel 4.16	Unit Testing Halaman Daftar Pasien.	95
Tabel 4.17	Unit Testing Halaman Daftar Seluruh Pasien.	95
Tabel 4.18	Unit Testing Halaman Detail Pasien.	96
Tabel 4.19	Unit Testing Halaman Histori Kajian.	96
Tabel 4.20	Unit Testing Halaman Periksa Kesehatan.	97
Tabel 4.21	Unit Testing Halaman Tambah Kajian.	97
Tabel 4.22	Unit Testing Halaman Tujuan Perawatan.	98
Tabel 4.23	Unit Testing Halaman inventaris.	98
Tabel 4.24	Unit Testing Halaman Rekap Kunjungan.	99
Tabel 4.25	Unit Testing Beranda Pasien.	99
Tabel 4.26	Unit Testing Profil Pasien.	99
Tabel 4.27	Unit Testing Riwayat Kajian Pasien.	100

Tabel 4.28	Unit Testing Detail Riwayat Kajian Pasien.	100
Tabel 4.29	Unit Testing Rekap Kunjungan Pasien.	100
Tabel 4.30	Unit Testing Detail Rekap Kunjungan Pasien.	101
Tabel 4.31	<i>User Acceptance Test</i>	101

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Lebih dari 13 juta orang di seluruh dunia menderita luka kronis setiap tahun. Jika luka kronis tidak sembuh dalam waktu lama, maka akan memperburuk kualitas hidup pasien dan keluarganya, yang mengakibatkan penyebaran infeksi atau bahkan komplikasi yang mengancam jiwa. Diagnosis, pengobatan, dan manajemen luka kronis yang efektif sangat penting dalam praktik klinis. Namun, penyebab luka kronis dan faktor yang mempengaruhi penyembuhan luka cukup kompleks dan beragam. Dalam proses diagnosis, pengobatan dan manajemen luka kronis, dokter dan perawat perlu memiliki pemahaman menyeluruh serta kelengkapan rekam medis, manajemen yang efektif dan pemantauan kondisi fisik umum pasien secara tepat waktu, laporan pemeriksaan laboratorium, penilaian dan pengobatan luka fraksional. (Wang dkk., 2018). Berikut data luka kronis diabetes mellitus secara global berdasarkan DALYs.

Tabel 1.1: Diabetes Mellitus DALYs (*disability-adjusted life year*) secara global (WorldHealthOrganization, 2022)

Tahun	Rank	<i>Diabetes Mellitus</i>
2000	14	38,5 juta
2010	11	52,7 juta
2019	9	70,4 juta

Dapat kita lihat dari data di atas, DALYs dari diabetes mellitus meningkat lebih dari 80% antara tahun 2000 dan 2019. Hal ini menyebabkan peningkatan jumlah pasien luka kronis di seluruh dunia. Tidak diragukan lagi kehidupan orang-orang setelah terjangkit diabetes mellitus sangat terpengaruh. Banyak yang tidak dapat bekerja, menjadi bergantung pada orang lain dan tidak dapat mengejar kehidupan sosial yang aktif. Di negara berkembang, situasinya bahkan lebih buruk karena seluruh keluarga mungkin tidak memiliki penghasilan jika anggota aktif menderita luka kronis atau telah menjalani amputasi. (Setacci C, 2020).

Pada penelitian yang berjudul “*A New Smart Mobile System for Chronic Wound Care Management*”, sistem ini memberikan solusi praktis untuk mengatasi tantangan utama dalam proses perawatan luka kronis, yang meliputi pengukuran luka yang tepat secara otomatis dan komposisi jaringan dasar luka, pemantauan penyembuhan luka berbentuk graph, penilaian luka standar dan komprehensif, dan manajemen dokumentasi kasus luka terintegrasi dalam konteks sistem informasi klinis rumah sakit umum yang ada. Tetapi model yang dikembangkan dalam deteksi tepi luka masih belum presisi. (Wang dkk., 2018).

Anuja Titus dan beberapa temannya berhasil mengembangkan algoritma baru berbasis mean shift berupa pewarnaan luka secara otomatis. Bagian kaki dapat diuraikan berdasarkan warna kulit, dan batas luka ditemukan menggunakan metode deteksi wilayah terhubung sederhana. Dalam batas luka, status penyembuhan selanjutnya dinilai berdasarkan model evaluasi warna merah-kuning-hitam. Selain itu, status penyembuhan dinilai secara kuantitatif, berdasarkan analisis tren catatan waktu untuk pasien tertentu. Tetapi algoritma ini memerlukan pengaturan parameter terlebih dahulu sehingga sensitifitas algoritma berubah terhadap data citra yang berbeda. Algoritma ini juga tidak bisa menyelesaikan kasus dimana lebih dari satu luka dalam satu gambar dan sensitifitas terhadap parameter berpengaruh secara signifikan terhadap hasil. (Titus dkk., 2017).

Dalam penelitian *medical imaging* sebelumnya oleh Aprilia Khairunnisa yang berjudul “*Pengaruh Penggunaan Color Model Lab Dalam Kalibrasi Warna Luka Menggunakan Metode Segmentasi K-Means dan Mean Shift*”, penentuan nilai k dan *centroid* awal pada metode *k-means* serta penentuan nilai w dan σ pada metode *mean shift* sangat berpengaruh pada hasil dari segmentasi kedua metode ini dimana nilai tersebut yang akan menentukan banyaknya jumlah *cluster* yang dihasilkan. Salah satu kekurangan dari penelitian ini, yaitu penelitian yang dilakukan masih kurang komprehensif, di mana hasil dari penelitian ini belum dapat memperlihatkan pengaruh dari penggunaan model warna LAB pada proses segmentasi. (Khairunnisa, 2021). Pada payung penelitian yang sama juga sedang dilakukan penelitian oleh Muhamad Rizki untuk mendeteksi tepi luka menggunakan metode *Active Contour* dan *Gradient vector flow*. Tujuannya untuk mendapat hasil terbaik yang dapat mendeteksi objek luka pada *medical imaging* luka kronis. (Rizki, 2022).

Pada buku yang berjudul “*Rancang Bangun Aplikasi Mobile Android Sebagai Alat Deteksi Warna Dasar Luka Dalam Membantu Proses Pengkajian Luka*

Kronis Dengan Nekrosis”, salah satu teknik pengkajian luka berdasarkan warna luka biasa dikenal dengan *The RYB (Red-Yellow-Black) wound classification system*. Pada umumnya, metode tersebut digunakan dengan mengandalkan subyektifitas dari perawat luka. Hasil penelitian dalam buku ini ialah perawat mampu mengetahui perbedaan warna luka secara otomatis yang membantu proses pengkajian luka kronis dengan nekrosis. (Ratna Aryani, 2018). Ratna Aryani bersama teman lainnya juga membuktikan bahwa debridement luka merupakan langkah penting dalam perawatan luka yang dilakukan oleh perawat, terutama untuk menghilangkan jaringan nekrotik dan merangsang munculnya jaringan granulasi. Luka tidak akan bisa sembuh selama luka hitam (jaringan nekrotik) belum diangkat. (Ratna Aryani, 2017). Ia juga melakukan penelitian yang menghasilkan perban basah dapat mempercepat proses penyembuhan luka. Perawat harus mempertimbangkan untuk menggunakan balutan lembab daripada perawatan standar untuk meningkatkan proses penyembuhan. Namun, perawat harus menjaga luka dari kelembaban yang berlebihan karena akan menyebabkan kerusakan pada kulit di sekitar luka atau di dalam luka itu sendiri. (Aryani, 2016).

Banyak perawat kurang pengetahuan tentang manajemen luka dan penilaian luka, dan telah disarankan bahwa WAT (*Wound Assessment Tool*) dapat memberikan dukungan bagi perawat di bidang ini. (Setacci C, 2015). Salah satu alat yang membantu perawat dalam proses mengkaji luka penyembuhan pasien adalah BWAT (*Bates-Jensen Wound Assessment Tool*). Pada buku yang berjudul “*Wound Care A Collaborative Practice Manual for Health Professionals*”, *Bates-Jensen Wound Assessment Tool* (BWAT) dievaluasi sebagai bagian dari penilaian standar dan program pengobatan dalam studi multicenter prospektif hasil penyembuhan luka. Studi ini memberikan data tentang penggunaan skor BWAT untuk mengidentifikasi perawatan dan mengukur penyembuhan. Selain digunakan untuk mengidentifikasi perawatan luka tertentu, BWAT telah digunakan untuk menggambarkan karakteristik tekanan ulkus berulang pada orang dengan cedera tulang belakang karena ulkus ini belum dijelaskan dengan baik. Karena BWAT mengevaluasi beberapa karakteristik luka, ini sangat cocok untuk menggambarkan karakteristik luka yang spesifik pada populasi atau luka khusus. BWAT dimasukkan ke dalam beberapa *electronic medical records*(EMR) organisasi perawatan kesehatan dan EMR cocok dalam hal entri data dan akses data untuk laporan. (Sussman dan Bates-Jensen, 2012). Dalam penelitian sebelumnya yang sedang berjalan oleh Salsa Rahmadati juga merancang aplikasi pengkajian luka kronis berbasis *Android* yang terfokus pada modul pengolahan citra.

Penelitian tersebut dilakukan untuk mengumpulkan data *ground truth* sehingga meningkatkan ketepatan akurasi pada penelitian selanjutnya. (Rahmadati, 2023).

Berdasarkan tulisan di atas, penulis tertarik untuk melanjutkan penelitian (Rahmadati, 2023) dalam pembuatan aplikasi skoring luka, dimana pada penelitian sebelumnya hanya terbatas pada skoring pengolahan data gambar. Penelitian ini diharapkan dapat membantu perawat dalam melakukan scoring luka kronis untuk mengidentifikasi tingkat penyembuhan luka pasien melalui bantuan aplikasi keperawatan luka (scoring luka). Penelitian ini sekaligus akan melanjutkan rangkaian penelitian sebelumnya terkait *medical imaging* dan menjadi dasar untuk penelitian selanjutnya.

B. Rumusan Masalah

Rumusan masalah pada penelitian ini adalah “Bagaimana perancangan aplikasi pengkajian luka berbasis android dengan menggunakan *Bates-Jensen Assessment Tool*? ”

C. Pembatasan Masalah

Pembatasan masalah pada penelitian ini adalah pengkajian luka kronis menggunakan *Bates-Jensen Wound Assessment Tool* (BWAT).

D. Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah membuat aplikasi pengkajian luka berbasis android dengan menggunakan *Bates-Jensen Assessment Tool*.

E. Manfaat Penelitian

Penelitian ini diharapkan dapat mengoptimisasi pengkajian luka pada *medical imaging* dari segi waktu dan ruang serta efisiensi. Penelitian ini juga diharapkan memberikan manfaat lainnya, antara lain:

1. Bagi penulis

Memperluas pengetahuan tentang *medical imaging*, menambah pengalaman dalam *programming*, memperoleh gelar sarjana di bidang Ilmu Komputer,

serta menjadi media untuk penulis dalam mengaplikasikan ilmu yang didapatkan dari kampus.

2. Bagi Program Studi Ilmu Komputer

Penelitian ini dapat menjadi pembuka untuk penelitian di masa depan, dan dapat memberikan panduan bagi mahasiswa program studi Ilmu Komputer tentang perancangan aplikasi pengkajian luka berbasis *Bates-Jensen*.

3. Bagi Universitas Negeri Jakarta

Menjadi evaluasi akademik program studi Ilmu Komputer dalam penyusunan skripsi sehingga dapat meningkatkan kualitas pendidikan dan lulusan program studi Ilmu Komputer di Universitas Negeri Jakarta.

BAB II

KAJIAN PUSTAKA

A. Pengertian *Bates-Jensen Wound Assessment Tool*

Bates-Jensen Assessment Tool atau BWAT adalah alat ukur yang digunakan untuk menilai dan memantau penyembuhan luka tekan dan luka kronis lainnya. Pada tahun 1990, *Bates-Jensen* mengembangkan PSST, yang direvisi pada tahun 2001 sebagai BWAT. PSST asli dikembangkan sebagai alat penilaian luka untuk penggunaan klinis dan penelitian. Selama bertahun-tahun, penggunaan BWAT telah berkembang untuk mengukur dan memprediksi penyembuhan luka dan digunakan dalam berbagai macam luka di luar ulkus dekubitus. BWAT telah memberikan dasar untuk banyak alat penilaian luka lainnya dan merupakan instrumen yang paling banyak digunakan. Hanya perubahan kecil yang dilakukan pada PSST untuk membuat alat generasi kedua, BWAT.

B. Kegunaan *Bates-Jensen Wound Assessment Tool*

BWAT direkomendasikan untuk digunakan dalam menilai dan memantau penyembuhan luka tekan dan luka kronis lainnya. Ini menggunakan skala numerik untuk menilai karakteristik luka dari yang terbaik hingga yang terburuk (lihat Lampiran 5D). Dua item tidak dinilai: lokasi dan bentuk. 13 sisanya adalah item yang diberi skor. Ini adalah: ukuran, kedalaman, tepi, *undermining* atau kantong, jenis jaringan nekrotik, jumlah jaringan nekrotik, jenis eksudat, jumlah eksudat, warna kulit di sekitarnya, edema jaringan perifer, indurasi jaringan perifer, jaringan granulasi, dan epitelisasi. Setiap item yang diberi skor muncul dengan deskriptor karakteristik yang dinilai pada skala (1 menunjukkan yang terbaik untuk karakteristik itu dan 5 menunjukkan yang terburuk). Setelah selesai dinilai untuk setiap item pada BWAT, 13 skor item dapat dijumlahkan untuk mendapatkan skor total untuk luka. Skor total kemudian dapat diplot pada rangkaian luka di bagian bawah alat untuk “melihat sekilas” penyembuhan atau degenerasi luka. Skor total berkisar dari 9 (penutupan luka) hingga 65 (degenerasi jaringan yang dalam). Alat ini memiliki lembar petunjuk penggunaan satu halaman, selain deskripsi item (Lampiran B). Ada juga panduan bergambar untuk melatih profesional kesehatan dalam menggunakan BWAT. Panduan Bergambar BWAT mencakup 102 foto dari

berbagai jenis luka, bukan hanya luka tekan, yang menggambarkan setiap deskriptor untuk setiap item BWAT. Validasi konten fotografi dicapai dalam proses konsensus tiga tahap yang bekerja dengan perawat yang mengkhususkan diri dalam perawatan luka. Gambar 2.1 memberikan contoh halaman dari panduan bergambar.



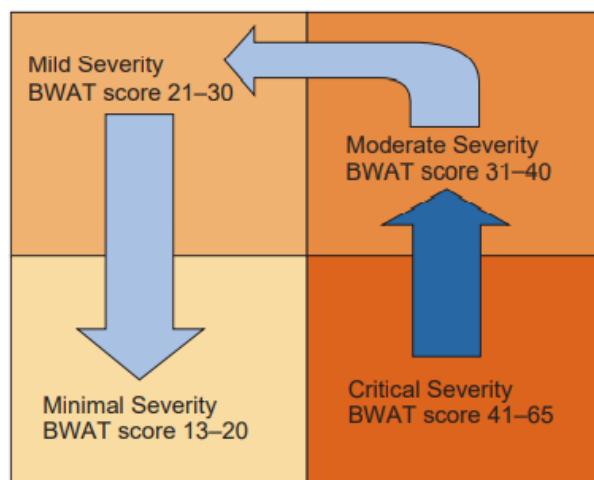
Gambar 2.1: Halaman dari panduan bergambar penilaian luka *Bates-Jensen* (Sussman dan Bates-Jensen, 2012)

Luka harus dinilai dengan BWAT pada awalnya untuk penilaian dasar dan secara berkala (yaitu, setidaknya setiap minggu) untuk mengevaluasi efektivitas intervensi.

C. Penggunaan Skor BWAT untuk Mengidentifikasi Status Keparahan dan Panduan Perawatan

Tingkat keparahan luka, serta status kesehatan pasien secara keseluruhan, dapat menentukan pendekatan manajemen yang tepat untuk penyembuhan. Status keparahan adalah ukuran derajat kerusakan jaringan atau beban luka pada pasien. Tujuan perawatan luka adalah untuk mengurangi status keparahan secara keseluruhan dan membuat penurunan ini tepat waktu. Perawat dapat menggunakan

BWAT untuk membantu perawat mengidentifikasi tingkat keparahan luka, dan dengan demikian memandu perencanaan perawatan pasien. Seperti yang ditunjukkan pada Gambar 2.2, skor BWAT dapat dibagi menjadi empat status keparahan yang disarankan: skor total 13-20 menunjukkan keparahan minimal; 21–30 menunjukkan keparahan ringan; 31-40 adalah tingkat keparahan sedang; dan 41-65 adalah tingkat keparahan yang ekstrim. Contoh algoritme perawatan untuk satu luka di masing-masing kondisi keparahan ini disajikan di bawah ini. Algoritma pengobatan ini berasal dari pedoman praktik klinis.



Gambar 2.2: Status keparahan berdasarkan skor BWAT. Tujuan terapi adalah (1) untuk mengurangi tingkat keparahan luka secara keseluruhan dan, dengan demikian, skor BWAT dan (2) untuk membuat penurunan secara tepat waktu. Ada perhatian yang sama mengenai tingkat keparahan luka dan durasi waktu yang dihabiskan luka dalam kondisi parah apa pun. (Sussman dan Bates-Jensen, 2012)

1. Skor Tingkat Keparahan Minimal BWAT 13–20

Luka dengan skor total BWAT 13-20 umumnya merupakan luka dengan ketebalan parsial yang dangkal. Gambar 5.23 menyajikan algoritme generik untuk perawatan luka dalam keadaan parah ini. Tujuan utama luka dalam keadaan parah ini adalah untuk mencegah kerusakan lebih lanjut dan menyediakan lingkungan luka yang lembab untuk penyembuhan.

2. Skor Tingkat Keparahan Ringan BWAT 21–30

Luka dengan tingkat keparahan ringan meliputi luka sebagian dan seluruh ketebalan. Gambar 5.24 menyajikan algoritme perawatan umum untuk luka dengan ketebalan parsial dengan skor keparahan ringan. Tujuan perawatan

luka parsial dengan tingkat keparahan ringan adalah untuk menyerap eksudat luka yang berlebihan, mempertahankan dasar luka yang bersih, dan menjaga lingkungan yang lembab. Luka full-thickness dengan skor keparahan ringan menawarkan lebih banyak pilihan untuk pengobatan karena luka dapat muncul sebagai luka bersih, full-thickness atau sebagai luka yang diisi dengan puing-puing nekrotik.

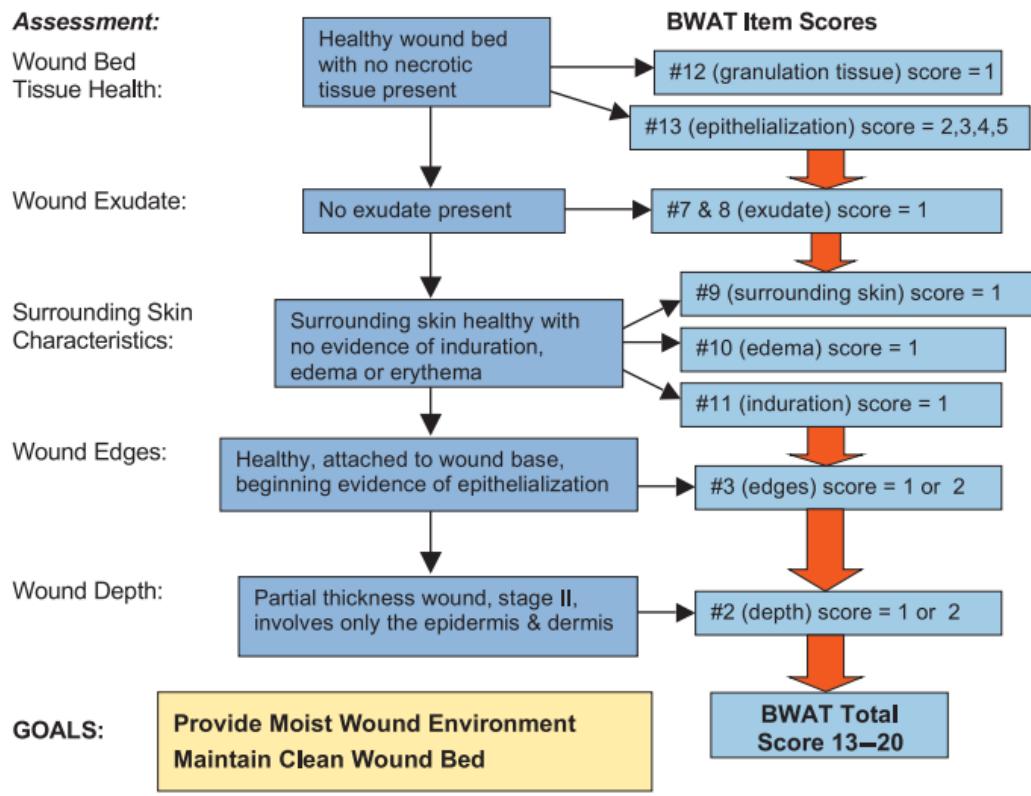
3. Skor Tingkat Keparahan Sedang BWAT 31–40

Gambar 5.25 adalah rencana perawatan untuk luka full-thickness dengan jaringan nekrotik. Tujuan perawatan luka full-thickness dengan tingkat keparahan sedang adalah untuk mendapatkan/mempertahankan dasar luka yang bersih, menyediakan lingkungan yang lembab, menyerap eksudat berlebih, mencegah penutupan dini, dan mengurangi ruang mati luka. Luka dengan skor keparahan sedang (dan ringan) memiliki presentasi klinis yang paling beragam, sehingga pilihan mengenai pengobatan sangat banyak. Gambar 5.26 sampai 5.28 menunjukkan algoritma pengobatan umum. Algoritma ini dapat digunakan untuk menentukan perawatan yang tepat untuk berbagai luka kronis dengan skor keparahan BWAT sedang hingga ekstrim. Luka dengan tingkat keparahan sedang sebagian besar merupakan luka full-thickness seperti ulkus dekubitus stadium III atau IV. Gambar 5.26 menyajikan kasus luka full-thickness dengan puing-puing nekrotik dan eksudat dalam jumlah besar. Perawatan difokuskan pada debridement dan penyerapan eksudat. Gambar 5.27 menyajikan kasus luka bersih full-thickness dengan undermining atau dead space, dan fokus pengobatan adalah menghilangkan dead space dan pencegahan penutupan luka prematur. Tujuan perawatan luka pada keadaan keparahan ini adalah untuk mendapatkan/mempertahankan dasar luka yang bersih, menyerap eksudat berlebih, menghilangkan ruang mati untuk mencegah penutupan luka prematur, dan menyediakan lingkungan luka yang lembab.

4. Skor Tingkat Kritis BWAT 41–65

Luka dengan skor total BWAT antara 41 dan 65 umumnya merupakan luka full-thickness yang dalam dengan manifestasi klinis yang lebih kritis, termasuk undermining dan nekrosis. Gambar 5.28 menyajikan algoritma untuk perawatan luka dengan eskar nekrotik. Tujuan perawatan luka pada

keadaan keparahan ini adalah untuk mengidentifikasi dan mengobati infeksi, mendapatkan dasar luka yang bersih, menyerap eksudat berlebih, menghilangkan ruang mati untuk mencegah penutupan luka prematur, dan menyediakan lingkungan luka yang lembab. Pertimbangan dalam Menggunakan Skor BWAT untuk Memandu Perawatan Penggunaan skor BWAT untuk menentukan keadaan keparahan dan membimbing pengobatan menawarkan satu pendekatan untuk mengelola luka. Pendekatan ini mungkin berguna dalam merancang pedoman pengobatan generik yang luas; namun, pasien tetap harus menggunakan penilaian klinis perawat untuk mengindividualisasikan rencana perawatan. Selain itu, rencana perawatan yang disajikan berdasarkan skor keparahan BWAT hanya berfokus pada perawatan luka topikal. Perhatian pada nutrisi, penggunaan permukaan penopang yang memadai, penentuan kebutuhan perawatan luka yang lebih lanjut, dan pertimbangan status seluruh pasien juga merupakan bagian dari rencana perawatan.



Treatment Plan:

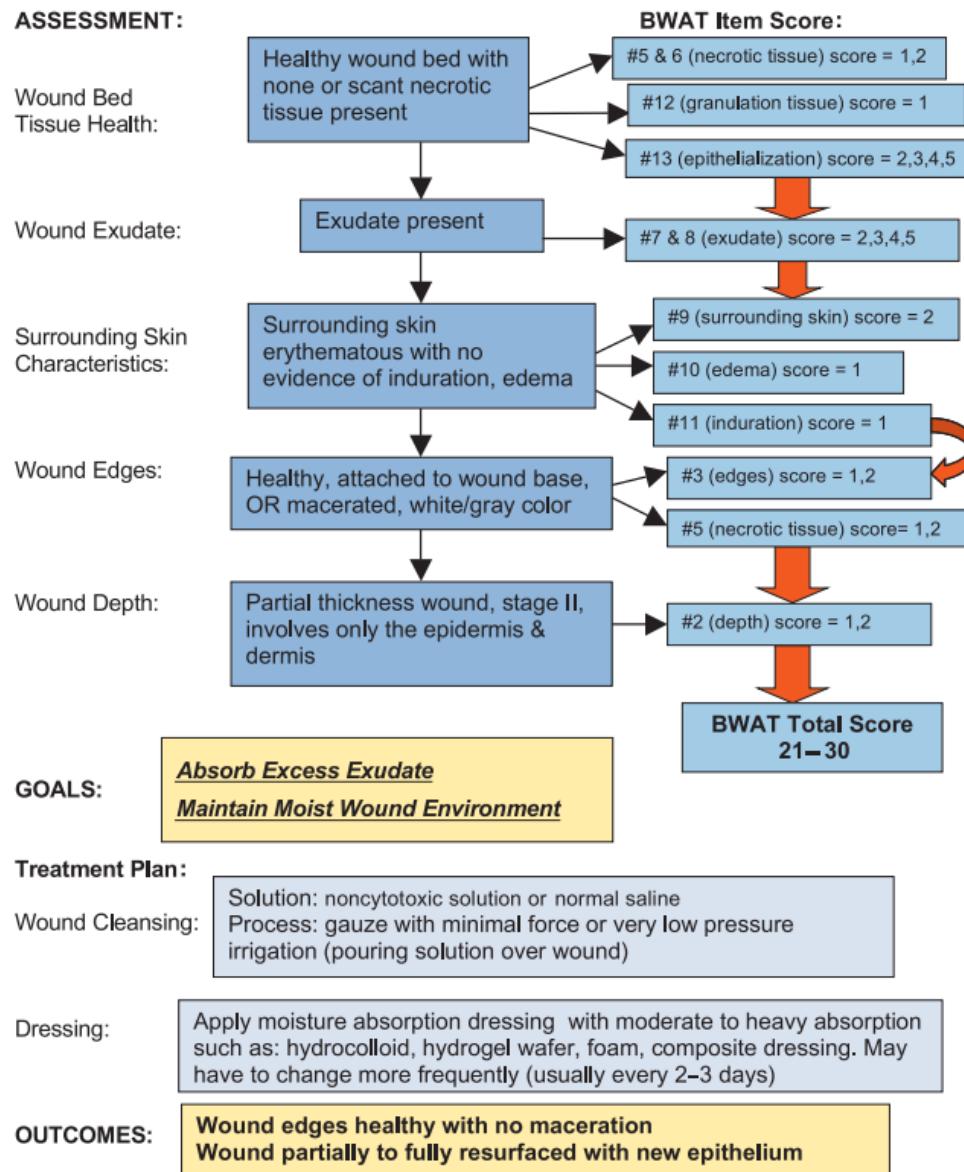
Wound Cleansing: Solution: noncytotoxic solution or normal saline
Process: gauze with minimal force or very low pressure irrigation (pouring solution over wound)

Primary Dressing: Apply moisture retentive dressing such as: hydrocolloid, hydrogel wafer, foam, composite dressing, thin film dressing

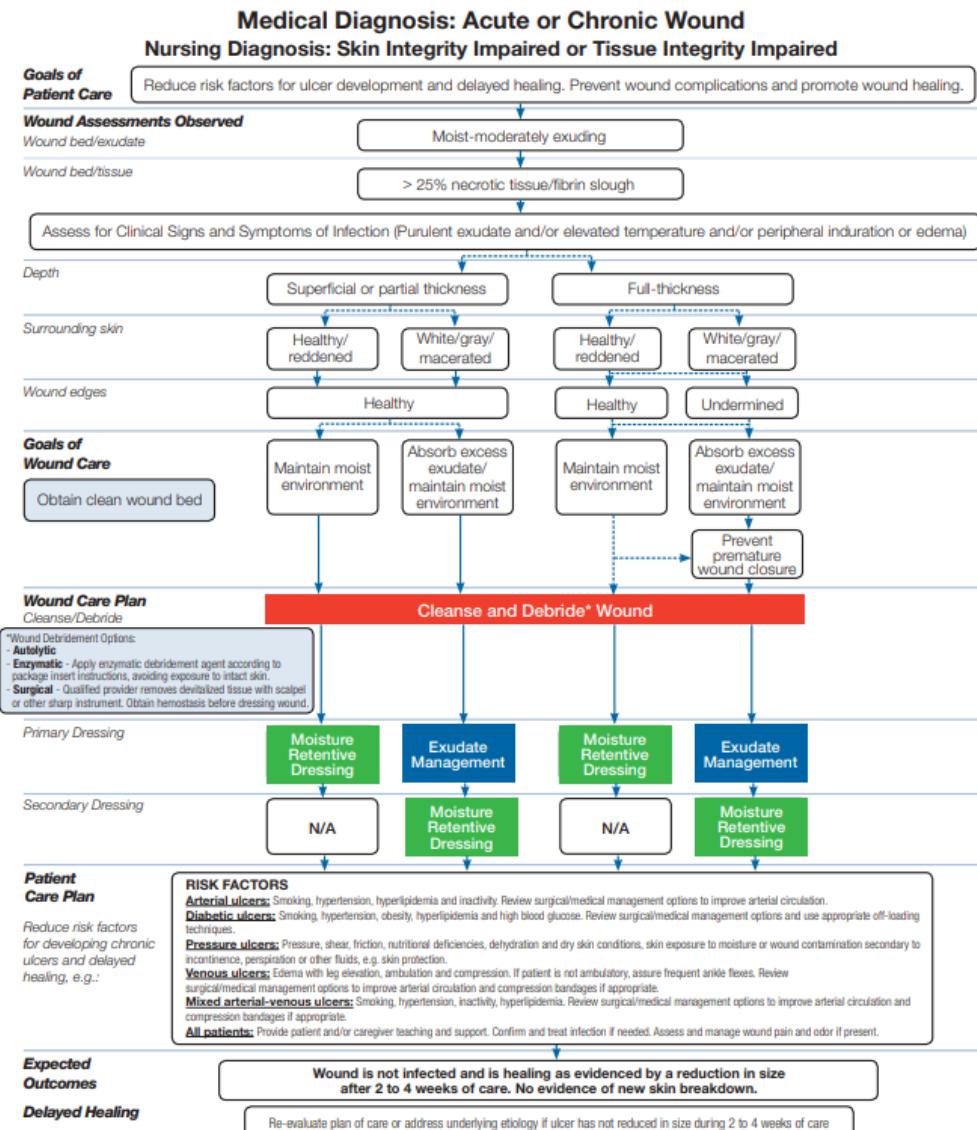
Secondary Dressing: Usually not required. If using amorphous hydrogel dressing or non-adherent foam dressing may need dressing or tape to keep gel or foam in place.

OUTCOME: **Wound 100% resurfaced with new epithelium**

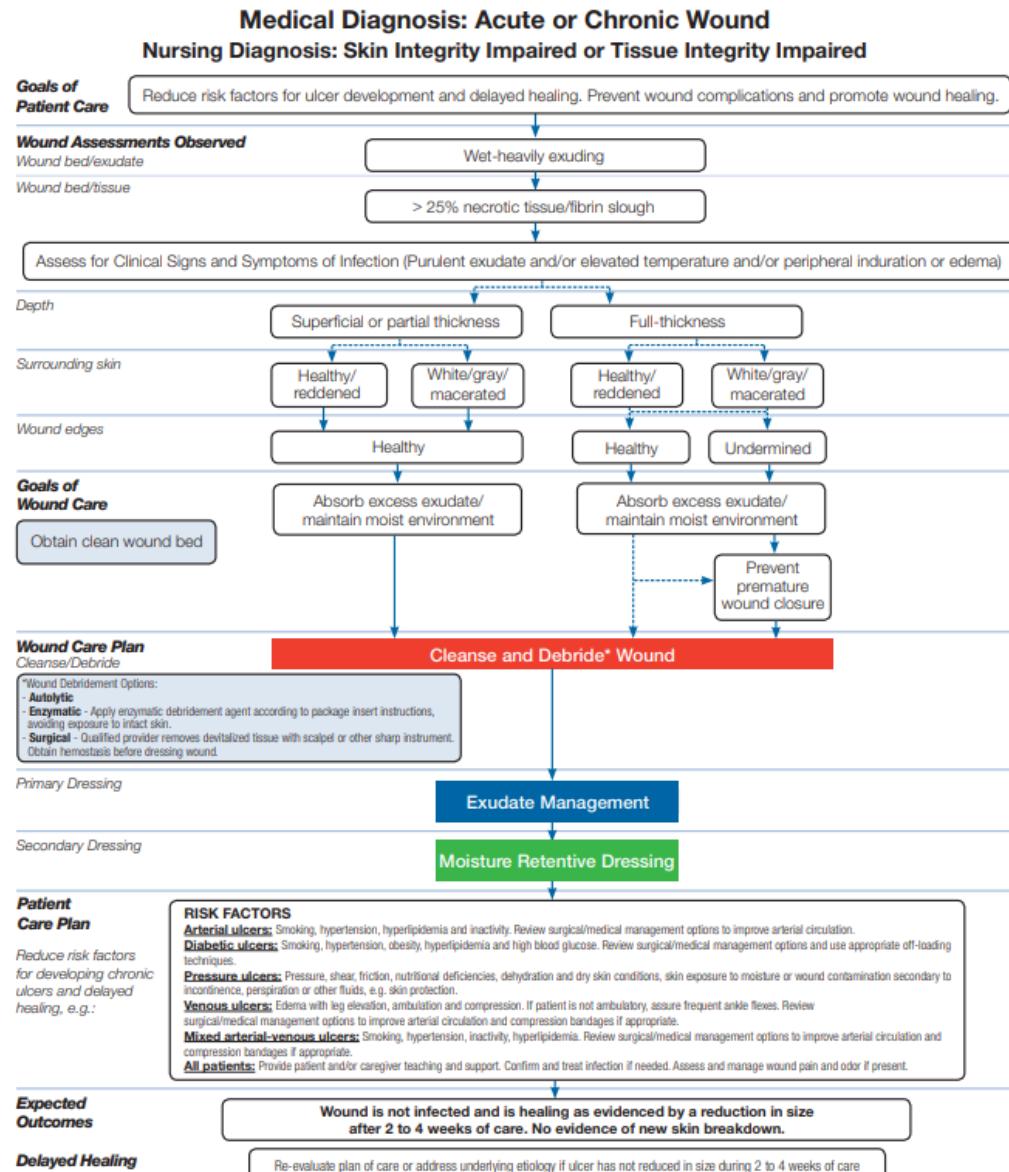
Gambar 2.3: Algoritma perawatan skor keparahan BWAT minimal untuk luka ringan, kering, dengan ketebalan sebagian. (Diadaptasi dari Convatec, dengan izin.) (Sussman dan Bates-Jensen, 2012)



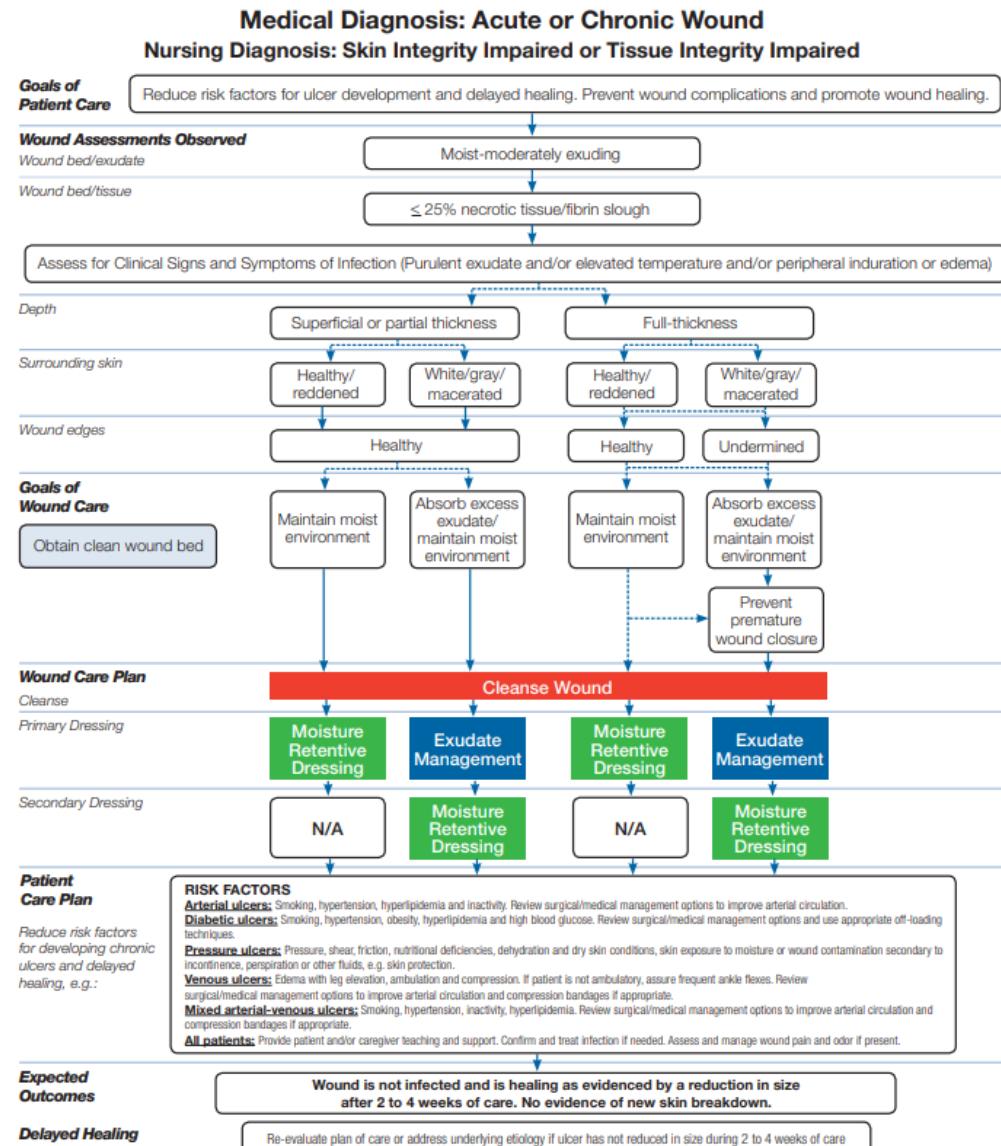
Gambar 2.4: Parsial-ketebalan luka dengan algoritma pengobatan skor keparahan BWAT ringan. (Diadaptasi dari Convatec, dengan izin.) (Sussman dan Bates-Jensen, 2012)



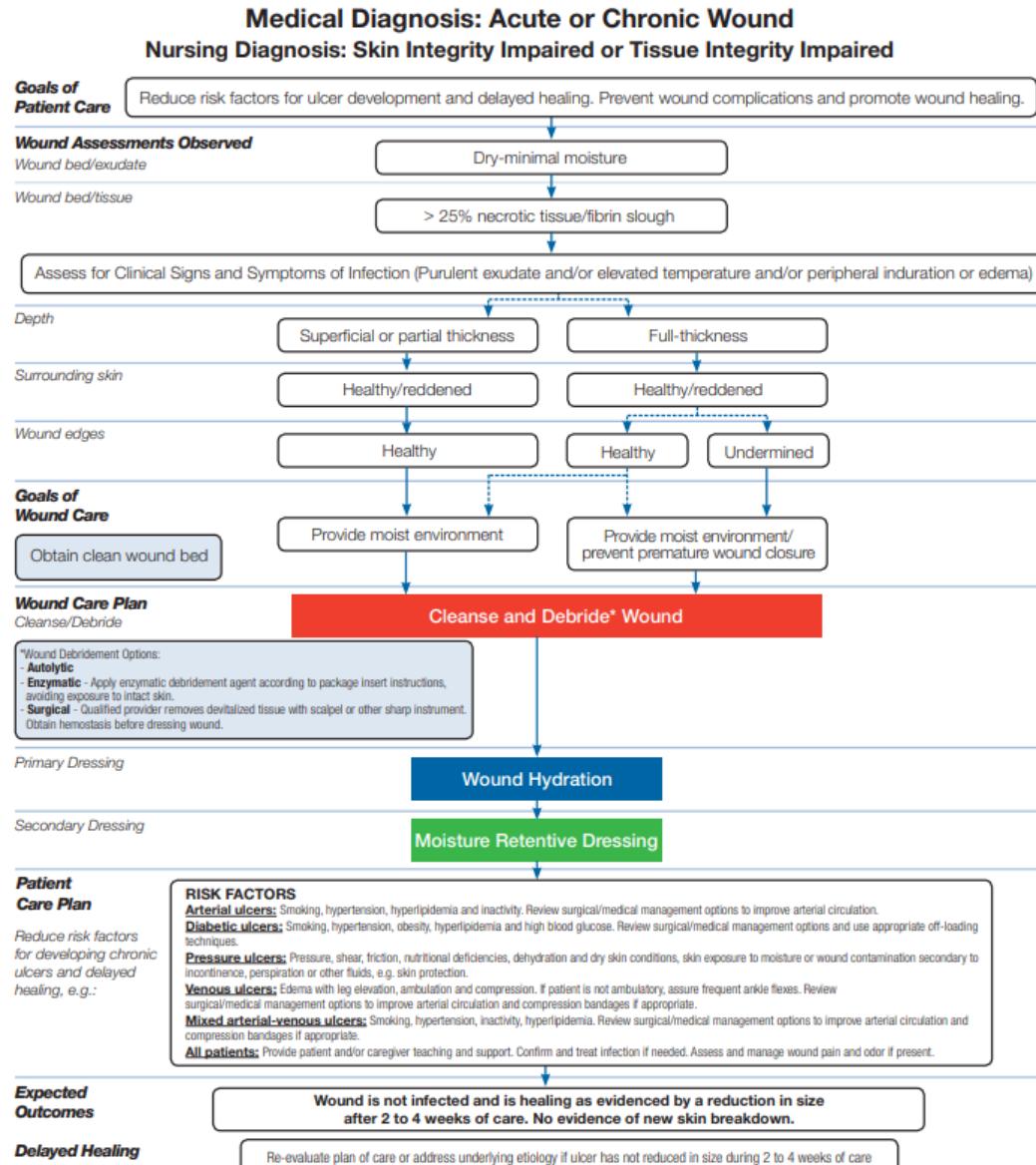
Gambar 2.5: Ulkus tekan stadium III atau stadium IV dengan ketebalan penuh dengan algoritme pengobatan skor keparahan BWAT sedang. (Diadaptasi dari ConvaTec, dengan izin.) (Sussman dan Bates-Jensen, 2012)



Gambar 2.6: Luka dengan ketebalan penuh umum dengan algoritme perawatan skor keparahan BWAT kritis. (Dicetak ulang dari Convatec, dengan izin.) (Sussman dan Bates-Jensen, 2012)



Gambar 2.7: Luka dengan ketebalan penuh umum dengan *undermining* atau *pocketing* dengan algoritme perawatan skor keparahan BWAT sedang. (Dicetak ulang dari ConvaTec, dengan izin.) (Sussman dan Bates-Jensen, 2012)



Gambar 2.8: Skor keparahan BWAT kritis dengan algoritme perawatan eschar kering. (Dicetak ulang dari Convatec, dengan izin.) (Sussman dan Bates-Jensen, 2012)

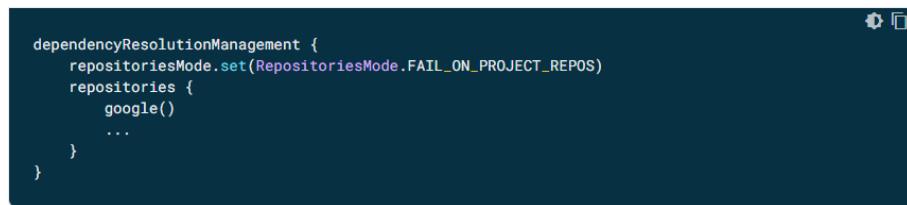
D. Activity vs Fragment

Activity adalah satu hal yang terfokus dapat dilakukan pengguna. Hampir semua *activity* berinteraksi dengan pengguna, jadi *class Activity* menangani pembuatan desain untuk menempatkan *User Interface*(UI) dengan *setContentView(View)*. *Fragment* mewakili bagian UI aplikasi yang dapat digunakan

kembali. *Fragment* mendefinisikan dan mengelola tata letaknya sendiri, memiliki siklus hidupnya sendiri, dan dapat menangani kejadian *inputnya* sendiri. *Fragment* tidak dapat hidup sendiri, harus dihosting oleh *activity* atau *fragment* lain. Perbedaan utama yang muncul dari definisi di atas adalah bahwa *fragment* bergantung pada *activity* yang ada sehingga hanya mewakili sebagian UI. Sebaliknya, *activity* dapat dianggap seperti wadah dimana semua komponen UI lainnya (termasuk *fragment*) akan ditempatkan. Tanpa *activity*, tidak akan ada *interface* pengguna. Cara untuk membuat *fragment* ialah:

1. *Setup environment*

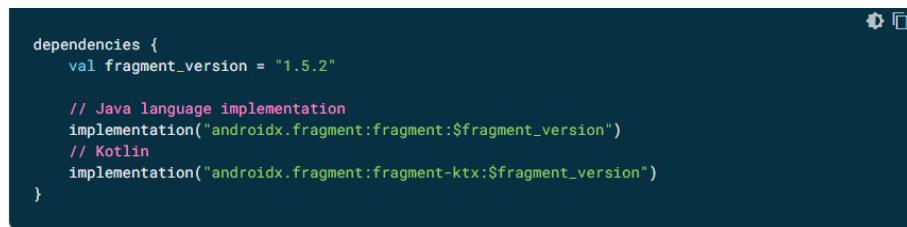
Fragment memerlukan dependency pada library *AndroidX Fragment*. Diperlukan menambahkan repositori Google Maven ke file *settings.gradle* proyek untuk menyertakan dependency



```
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        ...
    }
}
```

Gambar 2.9: Contoh menambahkan repositori *Google Maven* pada *dependency* (DeveloperAndroid, 2023)

Untuk menyertakan *AndroidX Fragment* ke proyek, tambahkan *dependency* berikut dalam file *build.gradle*:



```
dependencies {
    val fragment_version = "1.5.2"

    // Java language implementation
    implementation("androidx.fragment:fragment:$fragment_version")
    // Kotlin
    implementation("androidx.fragment:fragment-ktx:$fragment_version")
}
```

Gambar 2.10: Contoh menambahkan *AndroidX Fragment* (DeveloperAndroid, 2023)

2. Buat kelas *fragment*

Untuk membuat *fragment*, perluas kelas *AndroidX Fragment*, dan ganti metodenya untuk menyisipkan logika aplikasi, mirip dengan cara membuat

kelas *Activity*. Untuk membuat *fragment* minimal yang menentukan *layout*nya sendiri, berikan *resource layout fragment* ke konstruktor dasar, contohnya sebagai berikut:



```
class ExampleFragment : Fragment(R.layout.example_fragment)
```

Gambar 2.11: Contoh berikan *resource layout fragment* ke konstruktor dasar (DeveloperAndroid, 2023)

Library fragment juga menyediakan kelas dasar *fragment* yang lebih khusus:

(a) *DialogFragment*

Menampilkan dialog mengambang. Menggunakan kelas ini untuk membuat dialog adalah alternatif yang baik untuk menggunakan metode pembantu dialog di kelas *Activity*, karena *fragment* secara otomatis menangani pembuatan dan pembersihan dialog.

(b) *PreferenceFragmentCompat*

Menampilkan hierarki objek *Preference* sebagai daftar. Kita dapat menggunakan *PreferenceFragmentCompat* untuk membuat layar pengaturan untuk aplikasi.

3. Tambahkan *fragment* ke *activity*

Umumnya, *fragment* harus disematkan dalam *AndroidX FragmentActivity* untuk menyumbangkan UI ke *layout activity* tersebut. *FragmentActivity* adalah kelas dasar untuk *AppCompatActivity*, jadi jika sudah membuat subkelas *AppCompatActivity* maka tidak perlu mengubah kelas dasar *activity*. Kita dapat menambahkan *fragment* ke hierarki *view activity* baik dengan mendefinisikan *fragment* di file *layout activity* atau dengan mendefinisikan wadah *fragment* di file *layout activity*, lalu menambahkan *fragment* secara terprogram dari dalam *activity*. Dalam kedua kasus tersebut, kita perlu menambahkan *FragmentContainerView* yang menentukan lokasi tempat *fragment* harus ditempatkan dalam hierarki *view activity*. Sangat disarankan untuk selalu menggunakan *FragmentContainerView* sebagai wadah untuk *fragment*, karena *FragmentContainerView* menyediakan perbaikan khusus untuk *fragment* yang tidak disediakan oleh grup *view* lain seperti *FrameLayout*.

(a) Menambahkan *fragment* melalui *XML*

Untuk menambahkan *fragment* secara deklaratif ke *XML layout activity*, gunakan elemen *FragmentContainerView* seperti berikut:



```
<!-- res/layout/example_activity.xml -->
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name="com.example.ExampleFragment" />
```

Gambar 2.12: Contoh *FragmentContainerView* pada *XML* (DeveloperAndroid, 2023)

Atribut *android:name* menentukan nama kelas *Fragment* yang akan dibuat *instance*-nya. Saat *layout activity* di-*inflate*, *fragment* yang ditentukan akan dibuat *instance*-nya, *onInflate()* dipanggil pada *fragment* yang baru dibuat *instance*-nya, dan *FragmentTransaction* dibuat untuk menambahkan *fragment* ke *FragmentManager*.

(b) Menambahkan *fragment* secara terprogram

Untuk menambahkan *fragment* secara terprogram ke *layout activity*, *layout* harus menyertakan *FragmentContainerView* yang berfungsi sebagai wadah *fragment*, seperti yang ditunjukkan dalam contoh berikut:



```
<!-- res/layout/example_activity.xml -->
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Gambar 2.13: Contoh menambahkan *fragment* secara terprogram. (DeveloperAndroid, 2023)

Tidak seperti pendekatan *XML*, atribut *android:name* tidak digunakan pada *FragmentContainerView* di sini, jadi tidak ada *fragment* tertentu yang dibuat secara otomatis. Sebagai gantinya, *FragmentTransaction* digunakan untuk membuat *instance fragment* dan menambahkannya ke *layout activity*

Saat *activity* berjalan, kita dapat melakukan transaksi *fragment* seperti menambahkan, menghapus, atau mengganti *fragment*. Di *FragmentActivity*,

kita bisa mendapatkan *instance FragmentManager*, yang dapat digunakan untuk membuat *FragmentTransaction*. Kemudian, kita bisa membuat *instance fragment* dalam metode *onCreate()* *activity* menggunakan *FragmentTransaction.add()*, meneruskan *ID ViewGroup* dari *container* di *layout* dan kelas *fragment* yang ingin ditambahkan, lalu kemudian melakukan transaksi, seperti yang ditunjukkan dalam contoh berikut:

```

class ExampleActivity : AppCompatActivity(R.layout.example_activity) {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        if (savedInstanceState == null) {
            supportFragmentManager.commit {
                setReorderingAllowed(true)
                add<ExampleFragment>(R.id.fragment_container_view)
            }
        }
    }
}

```

Gambar 2.14: Contoh membuat *FragmentTransaction*. (DeveloperAndroid, 2023)

Pada contoh sebelumnya, perhatikan bahwa transaksi *fragment* hanya dibuat ketika *storedInstanceState* adalah *null*. Ini untuk memastikan bahwa *fragment* hanya ditambahkan sekali, saat *activity* pertama kali dibuat. Saat terjadi perubahan konfigurasi dan *activity* dibuat ulang, *saveInstanceState* tidak lagi *null*, dan *fragment* tidak perlu ditambahkan untuk kedua kalinya, karena *fragment* secara otomatis dipulihkan dari *saveInstanceState*. Jika *fragment* memerlukan beberapa data awal, argumen dapat diteruskan ke *fragment* dengan menyediakan *bundle* dalam panggilan ke *FragmentTransaction.add()*, seperti yang ditunjukkan di bawah ini:

```

class ExampleActivity : AppCompatActivity(R.layout.example_activity) {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        if (savedInstanceState == null) {
            val bundle = bundleOf("some_int" to 0)
            supportFragmentManager.commit {
                setReorderingAllowed(true)
                add<ExampleFragment>(R.id.fragment_container_view, args = bundle)
            }
        }
    }
}

```

Gambar 2.15: Contoh menambahkan beberapa data awal. (DeveloperAndroid, 2023)

Bundle argumen kemudian dapat diambil dari dalam *fragment* dengan

memanggil *requireArgument()*, dan metode pengambil *bundle* yang sesuai dapat digunakan untuk mengambil setiap argumen.



Gambar 2.16: Contoh memanggil *requireArgmunet()*. (DeveloperAndroid, 2023)

E. *Navigation dan menu*

Navigation mengacu pada interaksi yang memungkinkan pengguna bernaligasi melintasi, masuk, dan mundur dari berbagai bagian konten dalam aplikasi. Komponen navigation dalam Android membantu menerapkan navigasi mulai dari klik tombol sederhana hingga pola yang lebih kompleks, seperti bilah aplikasi dan panel samping navigasi. Komponen navigation juga memastikan pengalaman pengguna yang konsisten dan dapat diprediksi dengan mengikuti serangkaian prinsip yang telah ditetapkan. Komponen navigation terdiri dari tiga bagian penting:

1. *Navigation graph*: sumber daya XML yang berisi semua informasi terkait navigasi di satu lokasi terpusat. Ini mencakup semua area konten individual dalam aplikasi, yang disebut *destinations*, serta kemungkinan jalur yang dapat diambil *user* melalui aplikasi.
2. *NavHost*: wadah kosong yang menampilkan tujuan dari *navigation graph*. Komponen *navigation* berisi implementasi *default NavHost*, *NavHostFragment*, yang menampilkan tujuan *fragment*.
3. *NavController*: objek yang mengelola navigasi aplikasi dalam *NavHost*. *NavController* mengatur pertukaran konten tujuan di *NavHost* saat *user* bergerak di seluruh aplikasi.

Saat menavigasi melalui aplikasi, kita memberi tahu *NavController* bahwa kita ingin menavigasi di sepanjang jalur tertentu dalam *Navigation graph* atau langsung ke tujuan tertentu. *NavController* kemudian menunjukkan tujuan yang sesuai di *NavHost*. Komponen *navigation* memberikan sejumlah manfaat lain sebagai berikut:

1. Menangani transaksi fragment.
2. Menangani action up dan action back dengan benar secara default.
3. Menyediakan sumber daya standar untuk animasi dan transisi.
4. Menerapkan dan menangani deep linking.
5. Termasuk pola navigasi UI, seperti panel samping navigasi dan navigasi bawah, dengan sedikit pekerjaan tambahan.
6. Safe Args – plugin Gradle yang menyediakan keamanan jenis saat menavigasi dan meneruskan data antar tujuan.
7. Dukungan ViewModel – kita dapat memasukkan ViewModel ke navigation graph untuk berbagi data terkait UI di antara destination graph.
8. Selain itu juga dapat menggunakan editor navigasi android studio untuk melihat dan mengedit navigation graph.

Cara untuk menggunakan *navigation* adalah sebagai berikut:

1. *Setup environment*

Untuk menyertakan dukungan *Navigation* dalam proyek, tambahkan dependensi berikut ke file *build.gradle* aplikasi:



```

dependencies {
    val nav_version = "2.5.2"

    // Java language implementation
    implementation("androidx.navigation:navigation-fragment:$nav_version")
    implementation("androidx.navigation:navigation-ui:$nav_version")

    // Kotlin
    implementation("androidx.navigation:navigation-fragment-ktx:$nav_version")
    implementation("androidx.navigation:navigation-ui-ktx:$nav_version")

    // Feature module Support
    implementation("androidx.navigation:navigation-dynamic-features-fragment:$nav_version")

    // Testing Navigation
    androidTestImplementation("androidx.navigation:navigation-testing:$nav_version")

    // Jetpack Compose Integration
    implementation("androidx.navigation:navigation-compose:$nav_version")
}

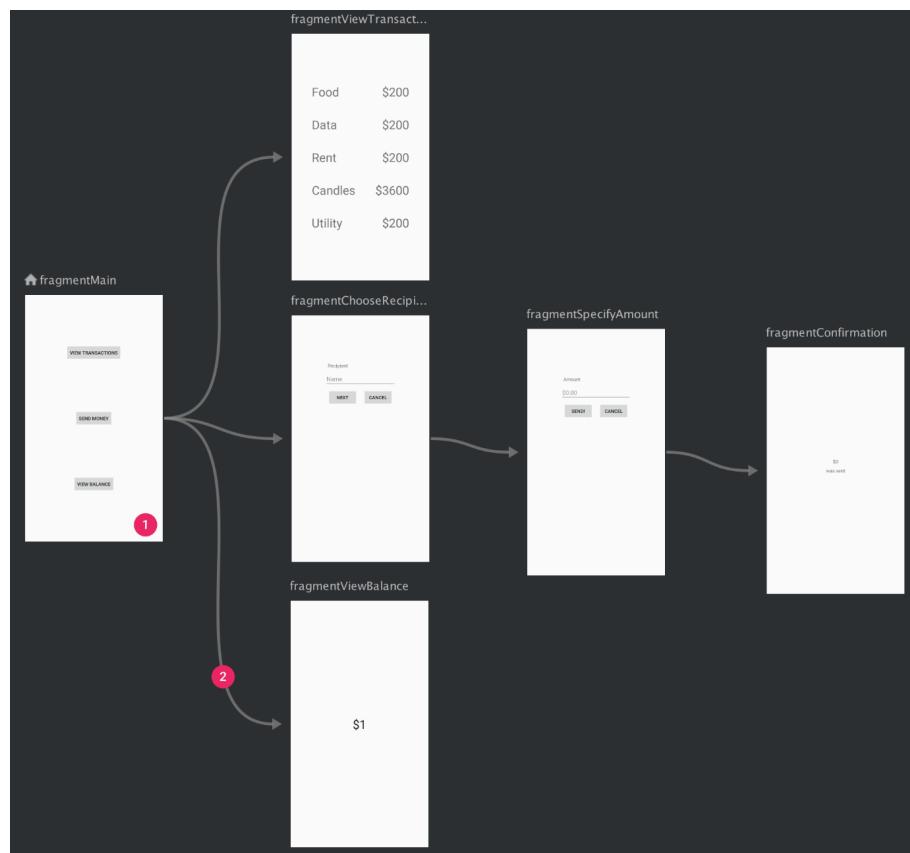
```

Gambar 2.17: Contoh dependensi untuk *navigation* (DeveloperAndroid, 2023)

2. Buat *navigation graph*

Navigasi terjadi di antara tujuan aplikasi, dimana saja di aplikasi yang dapat dinavigasi oleh pengguna. Tujuan ini terhubung melalui tindakan. *Navigation graph* adalah *file resource* yang berisi semua tujuan dan tindakan. *Graph* mewakili semua jalur *navigation* aplikasi.

Gambar 1 menunjukkan representasi visual *graph navigation* untuk aplikasi sampel yang berisi enam tujuan yang dihubungkan oleh lima tindakan. Setiap tujuan diwakili oleh gambar mini pratinjau, dan tindakan menghubungkan diwakili oleh panah yang menunjukkan bagaimana pengguna dapat menavigasi dari satu tujuan ke tujuan lainnya.



Gambar 2.18: Grafik navigasi yang menampilkan pratinjau dari enam tujuan berbeda yang terhubung melalui lima tindakan (DeveloperAndroid, 2023)

- (a) *Destinations* adalah area konten yang berbeda di aplikasi.
- (b) *Actions* adalah koneksi logis antara tujuan yang mewakili jalur yang dapat diambil pengguna.

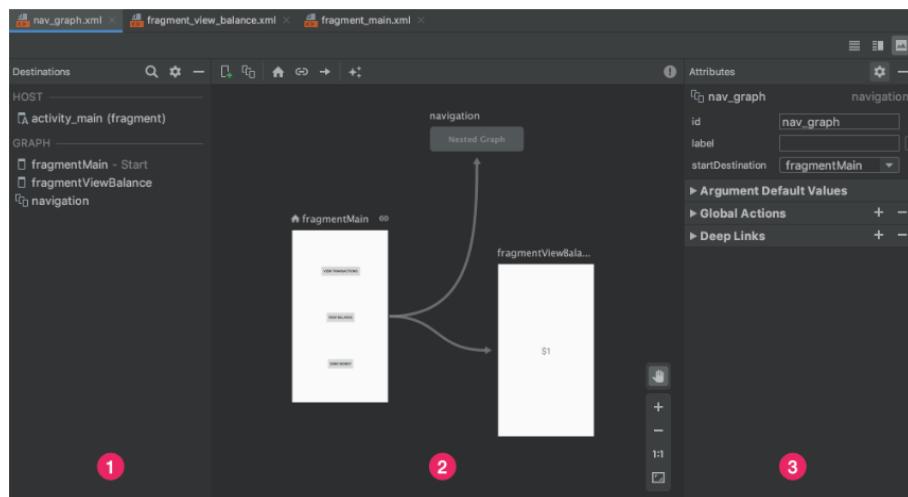
Untuk menambahkan *graph navigation* ke proyek adalah sebagai berikut:

- Di jendela *project*, klik kanan pada direktori *res* dan pilih *New > Android Resource File*. Dialog *file resource* baru muncul.
- Ketik nama di *field* nama *file*, seperti "*navgraph*".
- Pilih navigasi dari daftar *drop-down* jenis *resource*, lalu klik *Ok*.

Ketika menambahkan *graph navigation* pertama, *Android Studio* membuat direktori *resource navigation* di dalam direktori *res*. Direktori ini berisi *file resource graph navigation* (*navgraph.xml*, misalnya).

3. *Navigation editor*

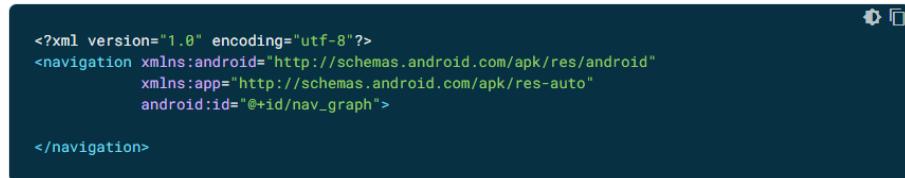
Setelah menambahkan *graph*, *Android Studio* membuka *frap* di *Navigation Editor*. Di sini dapat mengedit *navigation graph* secara visual atau langsung mengedit *XML* yang mendasarinya.



Gambar 2.19: *Navigation Editor* (DeveloperAndroid, 2023)

- Destinations panel*: mencantumkan *host* navigasi dan semua tujuan yang saat ini ada di *Graph Editor*.
- Graph Editor*: berisi representasi visual dari *navigation graph*. Dapat beralih antara tampilan desain dan representasi *XML* yang mendasarinya dalam tampilan teks.
- Attributes*: menampilkan atribut untuk item yang saat ini dipilih dalam *navigation graph*.

Klik tab *Text* untuk melihat *XML* yang sesuai, yang akan terlihat mirip dengan gambar berikut:



```
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/nav_graph">

</navigation>
```

Gambar 2.20: Contoh *XML navigation graph* (DeveloperAndroid, 2023)

Elemen *<navigation>* adalah elemen *root* dari *navigation graph*. Saat menambahkan tujuan dan menghubungkan tindakan ke *graph*, dapat melihat elemen *<destination>* dan *<action>* yang sesuai di sini sebagai elemen turunan. Jika memiliki *nest graph*, *graph* tersebut muncul sebagai elemen anak *<navigation>*.

4. *Menu*

Menu adalah komponen *user interface* yang umum di banyak jenis aplikasi. Untuk memberikan pengalaman pengguna yang familiar dan konsisten harus menggunakan *menu API* untuk menyajikan *user action* dan opsi lain dalam *activity*.

Mulai *android 3.0*, perangkat yang diberdayakan android tidak lagi diharuskan menyediakan tombol *menu* khusus. Dengan perubahan ini, aplikasi android harus bermigrasi dari ketergantungan pada panel menu 6 item tradisional dan sebagai gantinya menyediakan bilah aplikasi untuk menyajikan tindakan pengguna yang umum.

Meskipun desain dan pengalaman pengguna untuk beberapa *item menu* telah berubah, semantik untuk menentukan serangkaian tindakan dan opsi masih didasarkan pada *menu API*. Cara membuat tiga tipe dasar *menu*: *Options menu and app bar*, *Context menu and contextual action mode*, *Popup menu*.

Untuk semua jenis *menu*, Android menyediakan format *XML* standar untuk mendefinisikan *item menu*. Alih-alih membuat *menu* dalam kode *aktivitas*, kita harus mendefinisikan *menu* dan semua itemnya dalam sumber data *menu XML*. Anda kemudian dapat mengembangkan sumber daya menu (memuatnya sebagai objek *Menu*) di *activity* atau *fragment*.

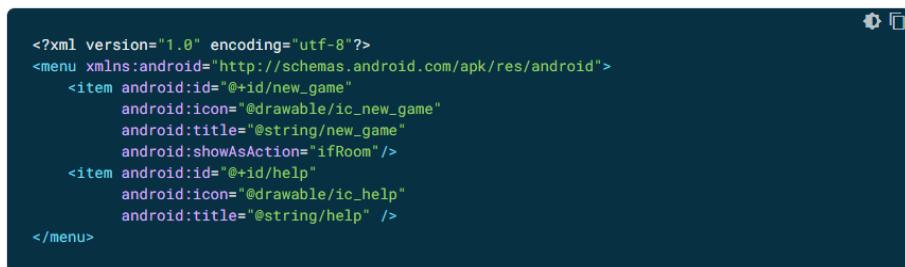
Menggunakan menu resource adalah praktik yang baik karena beberapa alasan:

- (a) Lebih mudah untuk memvisualisasikan struktur *menu* dalam *XML*.
- (b) Ini memisahkan konten untuk *menu* dari *applications behavioral code*.
- (c) Ini memungkinkan membuat konfigurasi *menu* alternatif untuk berbagai versi platform, ukuran layar, dan konfigurasi lainnya dengan memanfaatkan *framework* aplikasi.

Untuk menentukan menu, buat file XML di dalam direktori res/menu/ dan buat menu dengan elemen berikut:

- (a) 1. *<menu>* = Mendefinisikan menu, yang merupakan wadah untuk *item menu*. Elemen *<menu>* harus menjadi simpul akar untuk *file* dan dapat menampung satu atau lebih elemen *<item>* dan *<group>*.
- (b) *<item>* = Membuat *menu item*, yang mewakili satu *item* dalam *menu*. Elemen ini mungkin berisi elemen sarang *<menu>* yang membuat *submenu*.
- (c) *<group>* = Wadah opsional yang tidak terlihat untuk elemen *<item>*. Ini memungkinkan untuk mengkategorikan *item menu* sehingga mereka berbagi properti seperti status aktif dan visibilitas.

Berikut contoh menu bernama *game_menu.xml*:



```

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/new_game"
          android:icon="@drawable/ic_new_game"
          android:title="@string/new_game"
          android:showAsAction="ifRoom"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>

```

Gambar 2.21: Contoh *menu* (DeveloperAndroid, 2023)

Elemen *<item>* mendukung beberapa atribut yang dapat digunakan untuk menentukan tampilan dan perilaku item. *Item* dalam *menu* di atas mencakup atribut berikut:

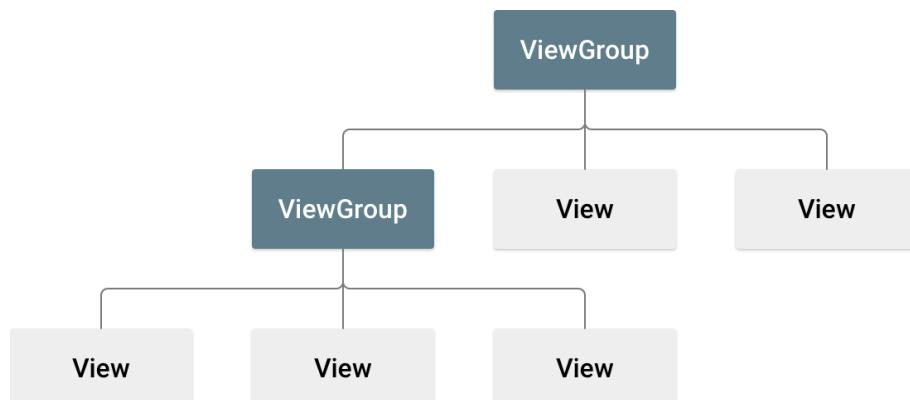
- (a) *android:id* = *ID resource* yang unik untuk *item*, yang memungkinkan aplikasi mengenali *item* saat pengguna memilihnya.

- (b) *android:icon* = Referensi ke *resource* gambar untuk digunakan sebagai ikon *item*.
- (c) *android:title* = referensi ke *string* untuk digunakan sebagai judul *item*.
- (d) *android:showAsAction* = Menentukan kapan dan bagaimana *item* ini akan muncul sebagai *item* tindakan di bilah aplikasi.

Ini adalah atribut terpenting yang harus digunakan, tetapi masih banyak lagi yang tersedia.

F. Layouting

Layout mendefinisikan struktur untuk user interface di aplikasi, seperti dalam suatu aktivitas. Semua elemen dalam layout dibangun menggunakan hierarki objek View dan ViewGroup. View biasanya menggambarkan sesuatu yang dapat dilihat dan berinteraksi dengan pengguna. Sedangkan ViewGroup adalah wadah tak terlihat yang mendefinisikan struktur layout untuk View dan objek ViewGroup lainnya.



Gambar 2.22: Ilustrasi hierarki tampilan, yang menentukan tata letak UI (DeveloperAndroid, 2023)

Objek *View* biasanya disebut "widget" dan bisa menjadi salah satu dari banyak *subclass*, seperti *Button* atau *TextView*. Objek *ViewGroup* biasanya disebut "layout" dapat menjadi salah satu dari banyak jenis yang menyediakan struktur *layout* yang berbeda, seperti *LinearLayout* atau *ConstraintLayout*. Deklarasi *layout* dapat dengan dua cara:

1. Deklarasikan elemen UI dalam *XML*.

Android menyediakan kosakata *XML* langsung yang sesuai dengan kelas dan subkelas *View*, seperti untuk *widget* dan *layout*. Kita juga dapat menggunakan *layout editor android studio* untuk membangun *layout XML* menggunakan *interface “drag-and-drop”*.

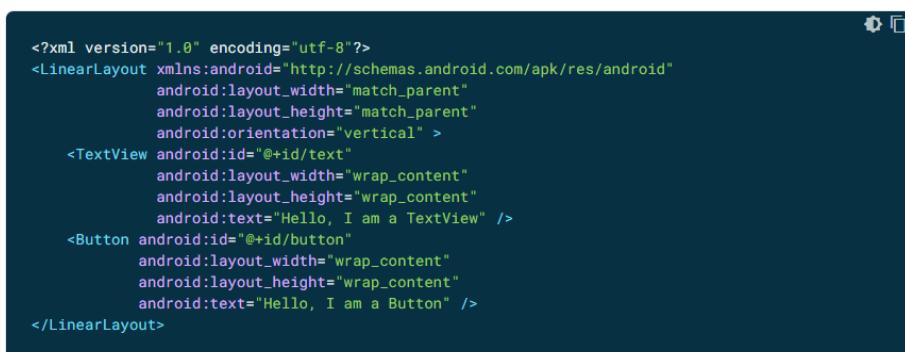
2. Buat *instance* elemen *layout* saat *runtime*.

Aplikasi kita dapat membuat objek *View* dan *ViewGroup* (dan memanipulasi propertinya) secara terprogram.

Mendeklarasikan UI dalam *XML* memungkinkan kita untuk memisahkan presentasi aplikasi dari kode yang mengontrol perilakunya. Menggunakan file *XML* juga memudahkan untuk menyediakan *layout* yang berbeda untuk ukuran dan orientasi layar yang berbeda. Kerangka kerja android memberi kita fleksibilitas untuk menggunakan salah satu atau kedua metode ini untuk membangun UI aplikasi. Misalnya bisa mendeklarasikan *layout default* aplikasi dalam *XML*, lalu memodifikasi *layout* saat waktu proses.

Dengan menggunakan kosakata *XML* Android, kita dapat dengan cepat mendesain tata letak UI dan elemen layar yang dikandungnya dengan cara yang sama seperti membuat halaman web dalam *HTML* dengan serangkaian elemen bersarang.

Setiap *file layout* harus berisi tepat satu elemen *root* yang harus berupa objek *View* atau *ViewGroup*. Setelah menentukan elemen *root*, dapat menambahkan objek atau *widget* *layout* tambahan sebagai elemen turunan untuk membangun hierarki *View* yang mendefinisikan *layout* secara bertahap. Contohnya ini *layout XML* yang menggunakan *LinearLayout* vertikal untuk menampung *TextView* dan *Button*:



```

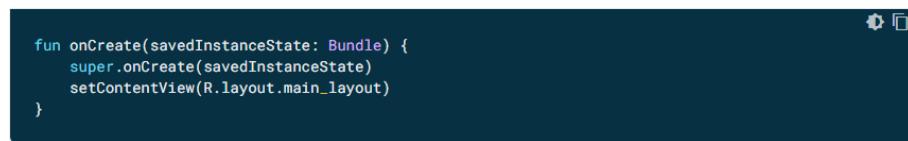
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>

```

Gambar 2.23: Contoh *layout XML* yang menggunakan *LinearLayout* (DeveloperAndroid, 2023)

Setelah dideklarasikan *layout* dalam *XML*, simpan file dengan ekstensi *.xml* di direktori *res/layout/* proyek Android kita sehingga akan dikompilasi dengan benar.

Saat mengompilasi aplikasi, setiap *file layout XML* dikompilasi ke dalam *resource View*. Diharuskan memuat *resource layout* dari kode aplikasi, dalam implementasi *callback Activity.onCreate()*. Lakukan dengan memanggil *setContentView()*, meneruskannya referensi ke *resource layout* dalam bentuk: *R.layout.layout_file_name*. Misalnya, jika *layout XML* disimpan sebagai *main_layout.xml*, maka akan memuatnya untuk *activity* seperti berikut:



Gambar 2.24: Contoh menggunakan *setContentView()* (DeveloperAndroid, 2023)

Metode *callback onCreate()* dalam *activity* dipanggil oleh *framework* Android saat *activity* diluncurkan. Setiap objek *View* dan *ViewGroup* mendukung berbagai atribut *XML* mereka sendiri. Beberapa atribut khusus untuk objek *View* (misalnya, *TextView* mendukung atribut *textSize*), tetapi atribut ini juga diwarisi oleh objek *View* apa pun yang dapat memperluas kelas ini. Beberapa umum untuk semua objek *View*, karena mereka diwarisi dari kelas *View root* (seperti atribut *id*). Dan, atribut lainnya dianggap sebagai "*parameter layout*", yang merupakan atribut yang menjelaskan orientasi *layout* tertentu dari objek *View*, seperti yang didefinisikan oleh objek induk *ViewGroup* objek tersebut. Objek *View* apa pun mungkin memiliki *ID integer* yang terkait dengannya, untuk mengidentifikasi *View* secara unik di dalam hierarki. Saat aplikasi dikompilasi, *ID* ini direferensikan sebagai bilangan bulat, tetapi *ID* biasanya ditetapkan dalam *file XML layout* sebagai *string*, dalam atribut *id*. Ini adalah atribut *XML* yang umum untuk semua objek *View* (didefinisikan oleh kelas *View*) dan akan sering digunakan. Sintaks untuk *ID* di dalam *tag XML* adalah:



Gambar 2.25: Contoh sintaks untuk *tag ID* (DeveloperAndroid, 2023)

Simbol (@) di awal *string* menunjukkan bahwa parser *XML* harus mengurai dan memperluas sisa *string ID* dan mengidentifikasinya sebagai *resource ID*. Tanda

tambah (+) berarti bahwa ini adalah nama *resource* baru yang harus dibuat dan ditambahkan ke *resource* kita (dalam file *R.java*). Ada sejumlah *resource ID* lain yang ditawarkan oleh *framework* Android. Saat mereferensikan *ID resource* Android, kita tidak memerlukan simbol tambah, tetapi harus menambahkan *namespace package* Android, seperti:



Gambar 2.26: Contoh menggunakan *namespace package* Android (DeveloperAndroid, 2023)

Dengan *namespace package* Android di tempat, kita sekarang mereferensikan *ID* dari kelas *resource android.R*, bukan dari kelas *resource* lokal. Untuk membuat View dan mereferensikannya dari aplikasi, pola umumnya adalah:

1. Tentukan *View/widget* dalam *file layout* dan tetapkan *ID* unik



Gambar 2.27: Menetapkan *ID* unik (DeveloperAndroid, 2023)

2. Kemudian buat *instance objek view* dan ambil dari *layout* (biasanya dalam metode *onCreate()*)



Gambar 2.28: Membuat *instance objek view* (DeveloperAndroid, 2023)

Mendefinisikan *ID* untuk objek *view* penting saat membuat *RelativeLayout*. Dalam *RelativeLayout*, *view* dapat menentukan *layoutnya* relatif terhadap tampilan *view* lainnya, yang direferensikan oleh *ID* unik. *ID* tidak harus unik diseluruh *tree*, tetapi harus unik di dalam bagian pohon yang dicari (yang mungkin sering berupa seluruh *tree*, jadi sebaiknya benar-benar unik jika memungkinkan).

G. Retrofit

1. Pengenalan

Cara menggunakan *retrofit* dalam Android dengan contoh API *GithubService*:

- (a) *Retrofit* mengubah API HTTP menjadi *interface* Java.

```
public interface GitHubService {
    @GET("users/{user}/repos")
    Call<List<Repo>> listRepos(@Path("user") String user);
}
```

Gambar 2.29: Contoh *retrofit* mengubah API menjadi *interface* (Square, 2023)

- (b) Kelas *Retrofit* menghasilkan implementasi *interface* *GitHubService*.

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.github.com/")
    .build();

GitHubService service = retrofit.create(GitHubService.class);
```

Gambar 2.30: Kelas *retrofit* (Square, 2023)

- (c) Setiap *Call* dari *GitHubService* yang dibuat dapat membuat *request* HTTP sinkron atau asinkron ke server web jarak jauh.

```
Call<List<Repo>> repos = service.listRepos("octocat");
```

Gambar 2.31: *Call* (Square, 2023)

Menggunakan anotasi untuk menjelaskan request HTTP:

- (a) Penggantian parameter URL dan dukungan parameter kueri
- (b) Konversi objek ke request body (misalnya JSON, protocol buffers)
- (c) Multi-bagian request body dan unggahan file

2. Deklarasi API

Anotasi pada metode interface dan parameternya menunjukkan bagaimana request akan ditangani.

(a) *Request method*

Setiap metode harus memiliki anotasi HTTP yang menyediakan metode request dan URL relatif. Ada delapan anotasi bawaan: HTTP, GET, POST, PUT, PATCH, DELETE, OPTIONS, dan HEAD. URL relatif resource ditentukan dalam anotasi.

```
@GET("users/list")
```

Gambar 2.32: Contoh request GET (Square, 2023)

Dapat juga menentukan parameter kueri di URL

```
@GET("users/list?sort=desc")
```

Gambar 2.33: Contoh request GET dengan menentukan parameter kueri di URL (Square, 2023)

(b) *Manipulasi URL*

Request URL dapat diperbarui secara dinamis menggunakan blok dan parameter pengganti pada metode. Blok pengganti adalah string alfanumerik yang dikelilingi oleh `{}` dan `.`. Parameter yang sesuai harus dianotasi dengan `@Path` menggunakan string yang sama.

```
@GET("group/{id}/users")
Call<List<User>> groupList(@Path("id") int groupId);
```

Gambar 2.34: Contoh request URL secara dinamis (Square, 2023)

Parameter kueri juga dapat ditambahkan.

```
@GET("group/{id}/users")
Call<List<User>> groupList(@Path("id") int groupId, @Query("sort") String sort);
```

Gambar 2.35: Contoh parameter kueri ditambahkan (Square, 2023)

Untuk kombinasi parameter kueri yang kompleks, Map dapat digunakan.

```
@GET("group/{id}/users")
Call<List<User>> groupList(@Path("id") int groupId, @QueryMap Map<String, String> options);
```

Gambar 2.36: Contoh menggunakan Map (Square, 2023)

(c) *Request Body*

Sebuah objek dapat ditentukan untuk digunakan sebagai body request HTTP dengan anotasi @Body.

```
@POST("users/new")
Call<User> createUser(@Body User user);
```

Gambar 2.37: Contoh anotasi Body (Square, 2023)

Objek juga akan dikonversi menggunakan konverter yang ditentukan pada instance Retrofit. Jika tidak ada konverter yang ditambahkan, hanya RequestBody yang dapat digunakan.

(d) Formulir dikode dan multipart

Metode juga dapat dideklarasikan untuk mengirim data yang disandikan bentuk dan multibagian. Data yang disandikan formulir dikirim ketika @FormUrlEncoded ada di metode. Setiap pasangan nilai kunci dianotasi dengan @Field yang berisi nama dan objek yang memberikan nilai.

```
@FormUrlEncoded
@POST("user/edit")
Call<User> updateUser(@Field("first_name") String first, @Field("last_name") String last);
```

Gambar 2.38: Contoh dikode (Square, 2023)

Request multipart digunakan ketika @Multipart hadir pada metode. Bagian dideklarasikan menggunakan anotasi @Part.

```
@Multipart
@PUT("user/photo")
Call<User> updateUser(@Part("photo") RequestBody photo, @Part("description") RequestBody description);
```

Gambar 2.39: Contoh multipart (Square, 2023)

Bagian multipart menggunakan salah satu konverter Retrofit atau mereka dapat mengimplementasikan RequestBody untuk menangani serialisasi mereka sendiri.

(e) Manipulasi *header*

Kita dapat mengatur tajuk statis untuk suatu metode menggunakan anotasi @Headers.

```
@Headers("Cache-Control: max-age=640000")
@GET("widget/list")
Call<List<Widget>> widgetList();
```

Gambar 2.40: Contoh anotasi header (Square, 2023)

```
@Headers({
    "Accept: application/vnd.github.v3.full+json",
    "User-Agent: Retrofit-Sample-App"
})
@GET("users/{username}")
Call<User> getUser(@Path("username") String username);
```

Gambar 2.41: Contoh anotasi header (Square, 2023)

Perhatikan bahwa header tidak saling menimpa. Semua header dengan nama yang sama akan disertakan dalam request. Request header dapat diperbarui secara dinamis menggunakan anotasi @Header. Parameter yang sesuai harus diberikan ke @Header. Jika nilainya null, header akan dihilangkan. Jika tidak, toString akan dipanggil pada nilai, dan hasilnya digunakan.

```
@GET("user")
Call<User> getUser(@Header("Authorization") String authorization)
```

Gambar 2.42: Contoh header dinamis (Square, 2023)

Mirip dengan parameter kueri, untuk kombinasi tajuk yang kompleks, Map dapat digunakan.

```
@GET("user")
Call<User> getUser(@HeaderMap Map<String, String> headers)
```

Gambar 2.43: Contoh header kompleks menggunakan Map (Square, 2023)

Header yang perlu ditambahkan ke setiap request dapat ditentukan menggunakan interseptor OkHttp.

3. Konfigurasi retrofit

Retrofit adalah kelas yang digunakan untuk mengubah interface API menjadi objek yang dapat dipanggil. Secara default, Retrofit akan memberi standar yang normal untuk platform tetapi memungkinkan untuk penyesuaian.

(a) Converters

Secara default, Retrofit hanya dapat melakukan deserialize body HTTP ke dalam tipe ResponseBody OkHttp dan hanya dapat menerima tipe RequestBody untuk @Body. Konverter dapat ditambahkan untuk mendukung jenis lain. Enam modul bersaudara mengadaptasi pustaka serialisasi populer untuk kenyamanan pengguna.

- i. Gson: com.squareup.retrofit2:converter-gson
- ii. Jackson: com.squareup.retrofit2:converter-jackson
- iii. Moshi: com.squareup.retrofit2:converter-moshi
- iv. Protobuf: com.squareup.retrofit2:converter-protobuf
- v. Wire: com.squareup.retrofit2:converter-wire
- vi. Simple XML: com.squareup.retrofit2:converter-simplexml
- vii. JAXB: com.squareup.retrofit2:converter-jaxb
- viii. Scalars (primitives, boxed, dan String): com.squareup.retrofit2:converter-scalars

Berikut adalah contoh penggunaan kelas GsonConverterFactory untuk menghasilkan implementasi interface GitHubService yang menggunakan Gson untuk deserialisasinya.

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.github.com/")
    .addConverterFactory(GsonConverterFactory.create())
    .build();

GitHubService service = retrofit.create(GitHubService.class);
```

Gambar 2.44: Contoh kelas GsonConverterFactory (DeveloperAndroid, 2023)

(b) Custom Converters

Jika perlu berkomunikasi dengan API yang menggunakan format konten yang tidak didukung Retrofit secara langsung (misalnya YAML, txt, format kustom) atau ingin menggunakan pustakan lain untuk mengimplementasikan format yang sudah ada, pengguna dapat dengan mudah membuat konverter pengguna sendiri. Buat kelas yang memperluas kelas Converter.Factory dan berikan instance saat membuat adaptor pengguna.

4. Download retrofit

Kode sumber Retrofit, sampelnya, dan situs web resmi tersedia di GitHub. Lalu pada Gradle tambahkan:

```
implementation 'com.squareup.retrofit2:retrofit:(insert latest version)'
```

Gambar 2.45: Contoh mendownload retrofit pada Gradle (DeveloperAndroid, 2023)

Retrofit membutuhkan minimal Java 8+ atau Android API 21+.

H. Manajemen Keperawatan

1. *Medical Record*

Sebelum dilakukannya proses penyembuhan oleh perawat, pasien diharuskan untuk mengisi data-data terkait diri mereka. Data-data tersebut diperlukan dalam meningkatkan keberhasilan penyembuhan sang pasien. Salah satu data yang diperlukan dalam proses penyembuhan yaitu rekap kunjungan pasien.

2. Rekap kunjungan pasien

Hal ini untuk mencatat tanggal kunjungan, jumlah kunjungan, perawat yang mengobati, dan hasil dari pemeriksaan serta keterangan yang diperlukan untuk membantu proses penyembuhan di hari kunjungan berikutnya.

3. Surat pernyataan persetujuan tindakan

Selanjutnya yaitu surat pernyataan persetujuan dari pasien. Pihak rumah sakit atau klinik meminta agar pasien menandatangani surat pernyataan persetujuan tindakan ini untuk dapat leluasa dalam melakukan perawatan. Pasien diberikan informasi mengenai kondisi pasien tersebut dan diharap mengerti serta memahami konsekuensi dari tindakan yang akan dilakukan sesuai dengan penjelasan yang telah diberikan oleh tenaga kesehatan yang melakukan perawatan.

4. Status pasien

Status pasien ini berisi data-data pribadi pasien tersebut. Status pasien terdiri dari nomor registrasi, nama lengkap pasien, agama/keyakinan pasien, tempat, tanggal lahir dan usia pasien, jenis kelamin pasien, alamat dan nomor telepon rumah pasien, nomor *handphone* dan alamat email pasien, diagnosa

keperawatan saat ini, *therapist* utama, tim *therapist*, dan alergi yang dialami oleh pasien.

5. Pengkajian umum luka

Pada pengkajian umum luka ini meliputi riwayat kejadian luka, riwayat perawatan sebelumnya dan faktor-faktor penyulit proses penyembuhan luka pada pasien.

6. Pengkajian lokal luka

Perawat melakukan pemeriksaan pada pasien yang nantinya akan menjadi penunjang keberhasilan penyembuhan luka pasien. Pemeriksaan yang dilakukan oleh perawat ialah tipe luka, tipe penyembuhan, gambar luka, pemeriksaan tanda-tanda vital dan penunjang, serta pemeriksaan fisik dan penunjang lainnya. Dilanjutkan dengan skoring luka yang dilakukan oleh perawat menggunakan assessment tools yang sudah disiapkan. Dalam gambar ini saya mengambil contoh skoring luka berdasarkan klinik moist care pusat penyembuhan luka.

7. Catatan perkembangan

Diakhir pemeriksaan pengkajian luka, perawat atau therapist memberi kesimpulan tujuan keperawatan yang nantinya dijadikan evaluasi dan rencana tindakan atau intervensi selanjutnya pada pasien.

8. Standar kompetensi perawat dalam perawatan luka

Enterostomal Therapy Nurse atau ETN merupakan spesialisasi keperawatan dalam bidang *enterostomal therapy*, “expert” dalam melakukan asuhan keperawatan *stoma*, perawatan luka dan *inkontinensia*. Atau yang dikenal dengan nama lainnya adalah *Wound Stoma and Continence Nurse Specialist* atau WOCNS. Pertama kalinya “*Enterostomal Therapist*” dikenalkan oleh Dr. Rupert Turnbull, dari *Cleveland clinic*, USA pada tahun 1958. Beliau bersama dengan salah satu pasien ileostominya, Mrs. Norma Gill, mengadakan program pelatihan untuk para profesional pertama kali tentang bagaimana membantu pasien stoma dalam menjalani kehidupannya yang baru.

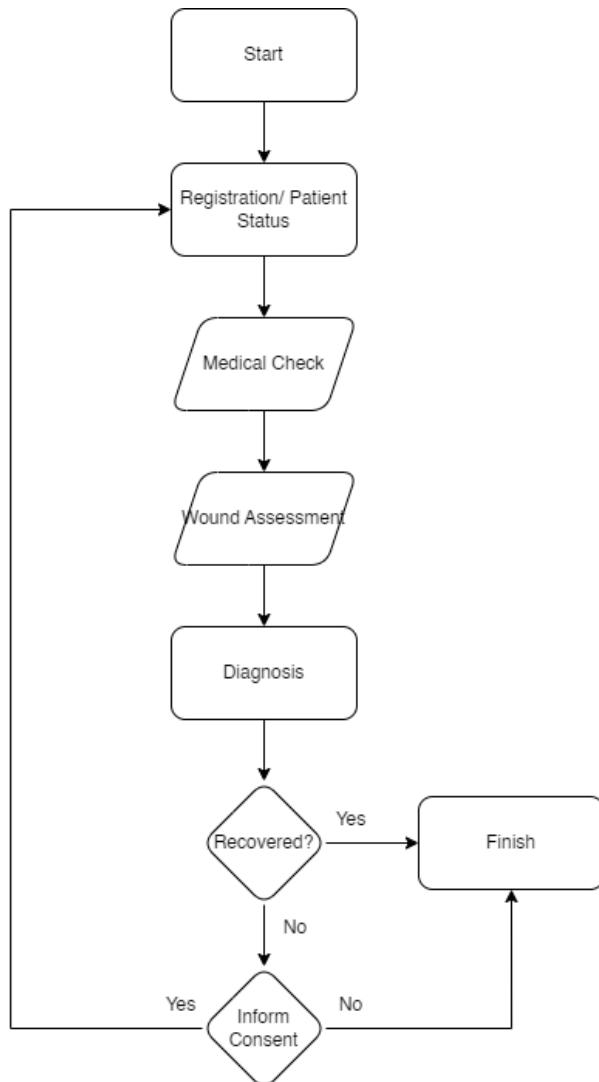
Sejak pelatihan pertama di *Cleveland clinic*, USA, tanggung jawab keilmuan *Enterostomal Therapist* semakin berkembang, kemudian berdirilah sekolah perawatan *Enterostomal Therapist* dibawah *World Council of Enterostomal*

Therapist(WCET) atau yang dikenal dengan ETNEP(*Enterostomal Therapy Nurse Education Programme*). Di akhir tahun 1970, *Enterostomal Therapist* mengembangkan diri pada bidang spesialisasi managemen perawatan luka kronik. Awalnya adalah karena keberhasilan dalam merawat komplikasi kulit disekitar stoma. Saat ini, hampir sebagian team *Enterostomal Therapist* melakukan perawatan luka kronik yang meliputi *pressure ulcer*, *dehisensi* luka operasi, *fistula*, *malignant wound*, *vascular ulcer* dan luka diabetikum.

ETN dalam wadah WCET (*World Council of Enterostomal Therapists*) sebagai lembaga non profit internasional telah mengembangkan sayapnya diseluruh penjuru eropa, america dan asia pasific. Keberadaan organisasi ini telah diakui secara international sebagai profesi keperawatan yang profesional. Pengembangan profesi dilakukan dengan mengadakan kongres 2 tahunan dan ETNEP. Di lima tahun terakhir ini, WCET telah membuka diri dengan konsep *twinning* program edukasi untuk lebih memperluas bidang keilmuan *enterostomal therapy nurse* terutama di asia, untuk mempermudah adanya kesulitan dalam bahasa.

Peran ETN adalah untuk memberikan perawatan secara langsung dan konseling pada ostomate selama berada di RS dan di rumah. Memfasilitasi rehabilitasi bagi ostomate untuk berhubungan dengan dokter, ahli gizi atau *sosial worker*. Merancang program edukasi bagi ostomate dan keluarganya dalam bentuk grup/ kelompok. Berpartisipasi dalam pendidikan keperawatan dalam hal pembelajaran tentang perawatan ostomi, *wound* dan *inkontinensia*. Melakukan uji coba produk dan melakukan penelitian yang berhubungan dengan pemakaianya untuk meningkatkan kualitas pelayanan. Melakukan pencatatan data, statistik dan yang berhubungan dengan pelayanan *Enterostomal Therapy*.

Di Indonesia, organisasi yang resmi menyelenggarakan pendidikan ETNEP dari WCET adalah InWOCNA (*Indonesia Wound Ostomy Continence Nurse Association*). Dengan mengikuti ETNEP, perawat dapat sertifikat keahlian resmi dari InWOCNA dan sudah diakui oleh PPNI (Persatuan Perawat Nasional Indonesia) bahwa perawat tersebut telah mengikuti pelatihan sesuai dengan kurikulum InWOCNA.
(IndonesianWoundOstomyContinenceNursesAssociation, 2023)



Gambar 2.46: Flowchart manajemen keperawatan

I. *Scrum*

Dalam mengerjakan penelitian ini saya menggunakan scrum untuk dijadikan kerangka kerja agar dapat menghasilkan solusi yang adaptif untuk masalah yang kompleks. Mengenai metode scrum ini sudah dijelaskan dengan sangat lengkap di penelitian sebelumnya oleh Salsa Rahmadati yang berjudul “Perancangan Aplikasi Pengkajian Luka Kronis Berbasis Android Modul Pengolahan Citra”.

BAB III

METODOLOGI PENELITIAN

A. Deskripsi Sistem

Penelitian yang akan dibuat oleh penulis adalah memperbarui dan melanjutkan penelitian sistem aplikasi yang telah dibuat sebelumnya oleh (Rahmadati, 2023) yang berjudul “PERANCANGAN APLIKASI PENGKAJIAN LUKA KRONIS BERBASIS ANDROID MODUL PENGOLAHAN CITRA” dalam pembuatan aplikasi skoring luka. Aplikasi ini berfungsi sebagai alat untuk membantu perawat dalam mengidentifikasi tingkat penyembuhan luka pasien. Selain hal tersebut, aplikasi ini juga akan diintegrasikan dengan sistem pewarnaan luka otomatis (Aprillia, 2021) dan sistem anotasi luka otomatis (Rizki, 2022) serta menambahkan fitur manajemen keperawatan. Penelitian ini juga akan terhubung dengan (Insan, 2023) pada database yang sedang dikembangkan.

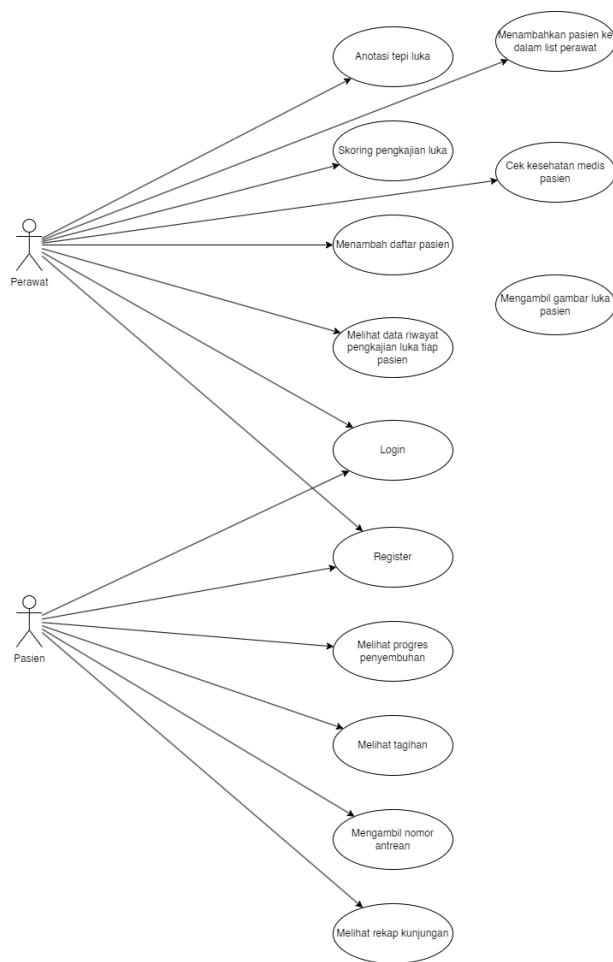
Penelitian ini dimulai dengan menganalisis kebutuhan, kemudian lanjut ke proses pengembangan, dan diakhiri dengan pengujian serta evaluasi. Tahapan penelitian yang penulis lakukan dalam penelitian aplikasi ini dapat dilihat pada gambar 3.1.



Gambar 3.1: Tahapan Penelitian

B. Analisa Kebutuhan

Berdasarkan wawancara dengan bu Ratna pada lampiran A, prioritas fitur pada aplikasi ini terfokus pada kajian luka yang belum ada pada aplikasi sebelumnya (Rahmadati, 2022) yang terdapat pada *Bates-Jensen Wound Assessment Tools* dan fitur manajemen keperawatan serta penambahan user pada pasien. Berikut adalah *usecase* yang telah didefinisikan berdasarkan wawancara.



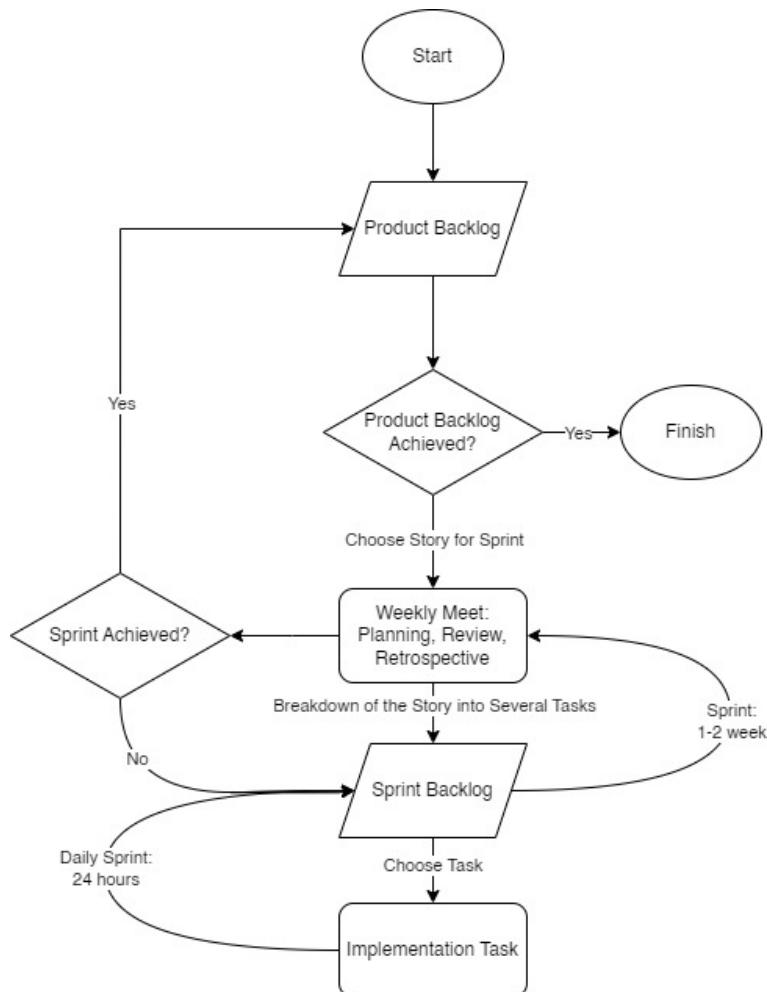
Gambar 3.2: Use Case

Perangkat yang dibutuhkan oleh penulis adalah sebagai berikut:

1. Laptop dengan spesifikasi minimum dapat menjalankan program aplikasi android studio.
2. Smartphone android untuk uji coba aplikasi ketika selesai.
3. Android Studio sebagai IDE untuk pembuatan aplikasi berbasis Android.
4. Kotlin sebagai bahasa pemrograman penyusun aplikasi.
5. Flask sebagai web framework.
6. MongoDB sebagai basis data.
7. Figma untuk desain tampilan aplikasi.

C. Perancangan Sistem

Metode perancangan sistem pada penelitian ini sesuai dengan komponen yang ada di dalam metode *Scrum*. Komponen-komponen tersebut terdiri dari *product backlog*, *sprint backlog*, *sprint*, *daily scrum*, dan pengujian sistem. Tahapan ini dilakukan setelah dilakukannya tahapan analisis kebutuhan.



Gambar 3.3: Tahapan Pengembangan Metode Scrum

1. *Product Backlog*

Product Backlog berisi fitur-fitur yang dibuat berdasarkan kebutuhan pengguna dan sudah didiskusikan dengan Scrum Master. Tabel 3.1 merupakan rincian *product backlog* dari aplikasi yang akan dibuat.

Tabel 3.1: Product Backlog

No	User Story	Role	Priority	Sprint
1	Skoring Bates-Jensen	Perawat	High	1
2	Konversi seluruh activity menjadi fragment		High	
3	Login user	Perawat, Pasien	High	
4	Evaluasi kondisi kesehatan pasien sebelum perawatan	Perawat	High	
5	Catatan perkembangan/ diagnosis untuk langkah penyembuhan selanjutnya	Perawat	High	3
6	Rekap kunjungan pasien	Perawat	High	4
7	Implementasi web service program warna luka	Perawat	Low	5
8	Implementasi web service anotasi luka	Perawat	Low	
9	Mengembangkan progres penyembuhan luka pasien penelitian sebelumnya	Perawat	Low	
10	<i>Wound History</i>	Perawat	High	6
11	Registrasi akun pasien	Pasien	Low	7
12	Mengambil nomor antrean	Pasien	Low	
13	Melihat rekap kunjungan	Pasien	Low	
14	Melihat histori kajian	Pasien	High	
15	Inventaris	Perawat	Low	8
16	Menambah pasien ke dalam akun perawat	Perawat	Low	9

Berdasarkan tabel di atas, Product Backlog pada penelitian ini terdiri dari 5 komponen yaitu: User Story, Role, Sprint, Status. User Story berisi fitur-fitur besar yang akan dibuat pada aplikasi ini. Komponen role pada tabel ini menandakan siapa yang menggunakan fitur tersebut. Priority menentukan tingkat prioritas dari sebuah User Story. Sprint menandakan informasi urutan fitur yang akan dibuat. Dan status menjelaskan apakah fitur tersebut sudah selesai atau belum.

2. Sprint Backlog

Sprint Backlog adalah daftar task yang perlu dikerjakan ataupun yang sudah dikerjakan pada sprint. Di dalam sprint backlog, berbagai task kecil dibuat. Dengan sprint backlog, seluruh anggota tim dapat melihat perkembangan dari setiap pekerjaan.

3. Sprint

Setelah dilakukan perencanaan pada Sprint Backlog, maka penggeraan sprint sudah dapat dimulai dan harus mengikuti jadwal penggeraan yang telah disepakati bersama tim. Dalam penelitian ini, interval sprint yang digunakan adalah satu sampai dua minggu. Berikut merupakan laporan hasil Sprint-1.

4. Sprint Review dan Sprint Retrospective

Sprint review dan sprint retrospective dilakukan pada setiap wal pekan yaitu di hari Selasa melalui daring menggunakan platform discord dan tatap muka secara langsung di UNJ. Pada awal pekan ini akan dilakukan evaluasi mengenai perkembangan proses pembuatan aplikasi maupun hambayan yang terjadi selama penggeraan di setiap sprint.

5. Deploy

Pada tahap ini peneliti akan melakukan uji aplikasi pengkajuan luka kronis 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 yang mengetahui apakah aplikasi sudah sesuai kebutuhan dan layak untuk diuji secara masif.

D. Pengujian

Pada tahap ini peneliti akan melakukan uji aplikasi pengkajuan luka kronis 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 yang 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 3.3.

Tabel 3.2: Skenario *Unit Testing*

Skenario Unit Testing	
Uji Fitur	Skenario Pengujian
Halaman Awal	Saat aplikasi dibuka akan muncul tampilan awal dengan pilihan "Login" dan "Register"
	Jika tombol "Login" ditekan, guest akan masuk ke halaman Login
	Jika tombol "Register" ditekan, guest akan masuk ke halaman Register
Register	Ketika mengisi form registrasi dengan lengkap kemudian submit, maka akan kembali ke halaman Login
	Ketika mengisi form registrasi tidak lengkap kemudian submit, maka akan menampilkan pesan kesalahan untuk melengkapi data yang kosong
Login	Ketika mengisi form login dengan data yang tidak sesuai kemudian klik submit, maka akan menampilkan pesan kesalahan username atau password
	Ketika mengisi form login dengan data yang sesuai dengan role perawat, maka akan masuk ke halaman beranda perawat
	Ketika mengisi form login dengan data yang sesuai dengan role pasien, maka akan masuk ke halaman beranda pasien
Beranda Perawat	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 pewarnaan luka ditekan, maka akan muncul halaman pewarnaan luka otomatis

Skenario Unit Testing	
Uji Fitur	Skenario Pengujian
	Saat tombol medical record ditekan, maka akan muncul halaman medical record pasien
Tambah hasil kajian	Saat pertama kali klik tombol tambah kajian baru muncul halaman pemeriksaan kesehatan pasien sebelum perawatan
	Ketika mengisi form pemeriksaan kesehatan dengan tidak lengkap maka akan menampilkan pesan kesalahan bahwa data yang diisi kurang sesuai atau belum lengkap
	Ketika mengisi form pemeriksaan kesehatan dengan lengkap maka akan masuk ke halaman ambil foto luka
	Ketika pengguna menekan tombol kamera, maka akan masuk ke halaman pengambilan foto luka
	Ketika pengguna selesai mengambil foto luka, maka akan diarahkan ke halaman anotasi tepi luka otomatis.
	Ketika anotasi tepi luka otomatis selesai bekerja, maka pengguna dapat menekan tombol "SAVE" untuk lanjut ke halaman kajian luka atau tekan tombol "EDIT" untuk menganotasi luka secara manual.
	Ketika mengisi form kajian luka secara tidak lengkap kemudian klik "NEXT", maka akan muncul pesan kesalahan bahwa data yang diisi belum lengkap
	Ketika mengisi form kajian luka secara lengkap kemudian klik "NEXT", maka akan diarahkan ke halaman tujuan perawatan.
	Ketika mengisi form tujuan perawatan secara tidak lengkap kemudian klik submit, maka akan muncul pesan kesalahan bahwa data yang diisi belum lengkap
	Ketika mengisi form tujuan perawatan secara lengkap kemudian klik submit, maka akan diarahkan ke halaman detail pasien yang baru saja diinput

Skenario Unit Testing	
Uji Fitur	Skenario Pengujian
Detail pasien	Saat masuk pertama kali ke halaman detail pasien, maka akan muncul informasi umum pasien dan progres tingkat kesembuhan luka pasien berdasarkan skoring kajian luka sebelumnya
	Saat tab Histori Kajian diklik maka akan muncul histori kajian pasien dan progres tingkat kesembuhan luka pasien berdasarkan skoring kajian luka sebelumnya
Beranda Pasien	Saat ikon profil pada kanan atas di-klik maka akan muncul halaman profil pasien
	Saat tombol riwayat kajian luka ditekan, maka akan muncul halaman riwayat kajian luka pasien
	Saat tombol rekap kunjungan ditekan, maka akan muncul halaman rekap kunjungan pasien
	Saat tombol medical record ditekan, maka akan muncul halaman medical record pasien
Profil Pasien	Saat ikon profil pada kanan atas di-klik maka akan muncul halaman profil pasien
	Saat tombol ambil antrean ditekan, maka akan diarahkan ke halaman antrean.
	Saat tombol riwayat kajian luka ditekan, maka akan muncul halaman riwayat kajian luka pasien.
	Saat tombol rekap kunjungan ditekan, maka akan muncul halaman rekap kunjungan pasien.
	Saat tombol medical record ditekan, maka akan muncul halaman medical record pasien.
Antrean	Saat masuk pertama kali ke halaman antrean, maka akan muncul nomor antrean saat ini, nama, nomor registrasi dan tombol ambil antrean.
	Saat tombol antrean ditekan, maka akan muncul nomor antrean yang diambil lalu tombol ambil antrean berubah menjadi selesai berobat.

Skenario Unit Testing	
Uji Fitur	Skenario Pengujian
	Saat tombol selesai berobat ditekan maka tombol tersebut akan menghilang lalu diarahkan kembali ke halaman beranda.
Riwayat kajian luka	Saat ikon profil pada kanan atas di-klik maka akan muncul halaman profil pasien
	Saat tombol riwayat kajian luka ditekan, maka akan muncul halaman riwayat kajian luka pasien
	Saat tombol rekap kunjungan ditekan, maka akan muncul halaman rekap kunjungan pasien
	Saat tombol medical record ditekan, maka akan muncul halaman medical record pasien
Rekap kunjungan	Saat ikon profil pada kanan atas di-klik maka akan muncul halaman profil pasien
	Saat tombol riwayat kajian luka ditekan, maka akan muncul halaman riwayat kajian luka pasien
	Saat tombol rekap kunjungan ditekan, maka akan muncul halaman rekap kunjungan pasien
	Saat tombol medical record ditekan, maka akan muncul halaman medical record pasien

2. User Acceptance Testing

Skenario dalam Uji Penerimaan Pengguna disusun berdasarkan fitur-fitur yang dapat diakses oleh pengguna dalam backlog produk. Detail skenario UAT yang akan dilaksanakan tercantum dalam tabel 3.4.

Tabel 3.3: Skenario *Unit Acceptance Testing*

Uji Fitur	Skenario Pengujian	Jenis Pengujian
Registrasi	Menekan tombol buat akun lalu mengisi form registrasi kemudian submit	UAT

Uji Fitur	Skenario Pengujian	Jenis Pengujian
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
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
Melihat histori kajian luka	Buka halaman detail pasien lalu klik tab histori kajian luka	UAT

Uji Fitur	Skenario Pengujian	Jenis Pengujian
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

Perancangan aplikasi pengkajian luka kronis ini menggunakan metode Scrum. Dalam metode ini, pengembangan sistem dilakukan secara bertahap melalui Sprint. Penelitian ini mencakup beberapa Sprint, di mana setiap Sprint berlangsung selama dua minggu. Pada awal setiap pekan, dilakukan perencanaan Sprint Backlog berdasarkan Product Backlog yang telah disepakati. Selain itu, setiap Sprint dalam proses pengembangan sistem didokumentasikan dalam laporan sebagai bagian dari evaluasi dan pemantauan perkembangan sebagai berikut:

1. *Sprint 1*

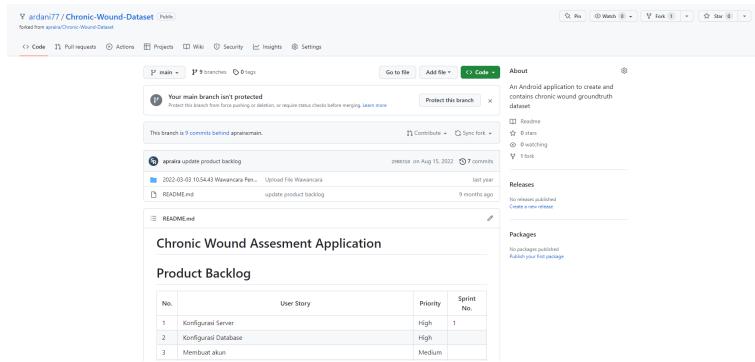
Pada *Sprint-1* ini terdapat 3 *Story* dan dipecah menjadi beberapa *task* sebagai berikut.

Tabel 4.1: Sprint 1

No.	User Story	Task	Status
1.	Skoring Bates-Jensen	fork repo frontend dan backend dari salsa	selesai
		Mockup untuk skoring	selesai
2.	Konversi seluruh activity menjadi fragment	Listing seluruh activity yang ada	selesai
		Buat fragment yang relevan dari activity yang ingin dikonversi	selesai
		Membuat navigation graph dari fragment yang ada	selesai
		Implementasi fragment logic dari fragment login yang sudah terintegrasi dengan web service	selesai
		Mengubah retrofit pemanggilan menjadi courutine	selesai
		Perubahan database	selesai
3.	Login user	Dashboard user setelah login	selesai

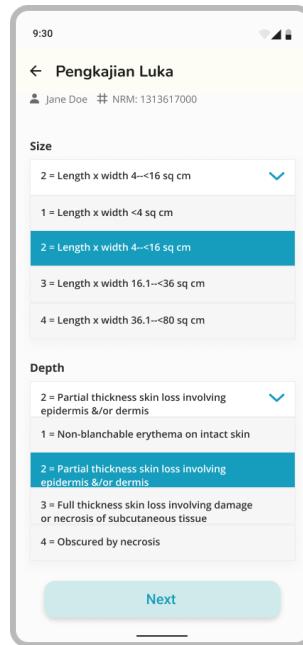
1. Skoring Bates-Jensen

Hal yang pertama kali dilakukan ialah melihat sejauh mana penelitian sebelumnya berjalan. Dengan cara fork repo frontend dan backend dari penelitian sebelumnya penulis dapat mengetahui skoring Bates-Jensen mana saja yang masih belum dilaksanakan dan melengkapinya.



Gambar 4.1: Fork repo frontend dan backend penelitian sebelumnya

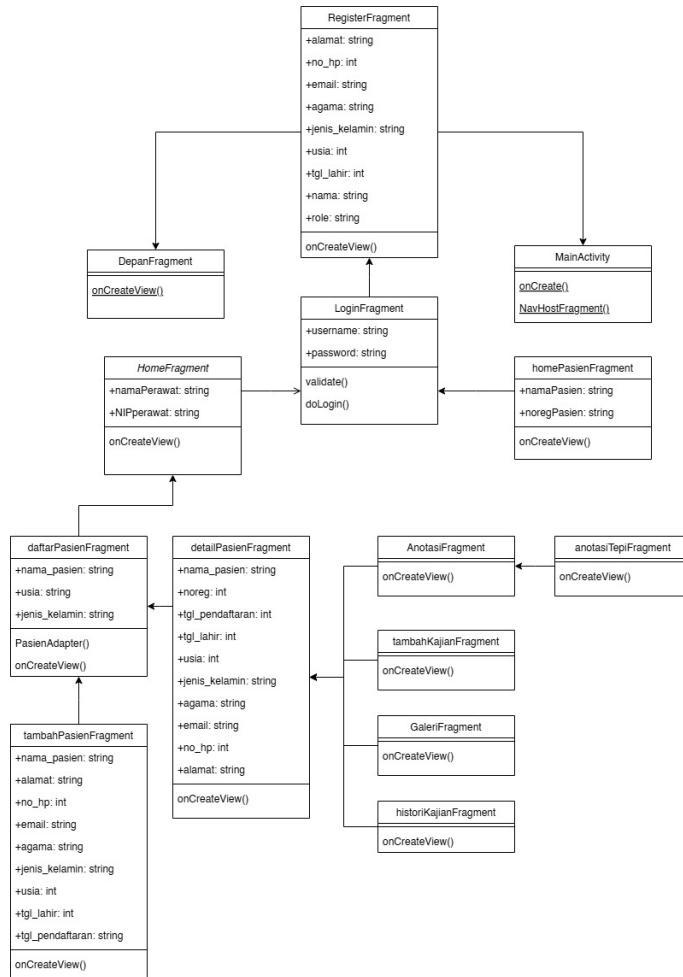
Setelah itu dibuatlah Mock-up UI untuk skoring lanjutan yang mana Design Styleguide mengikuti penelitian sebelumnya.



Gambar 4.2: Mock-up skoring pengkajian luka lanjutan

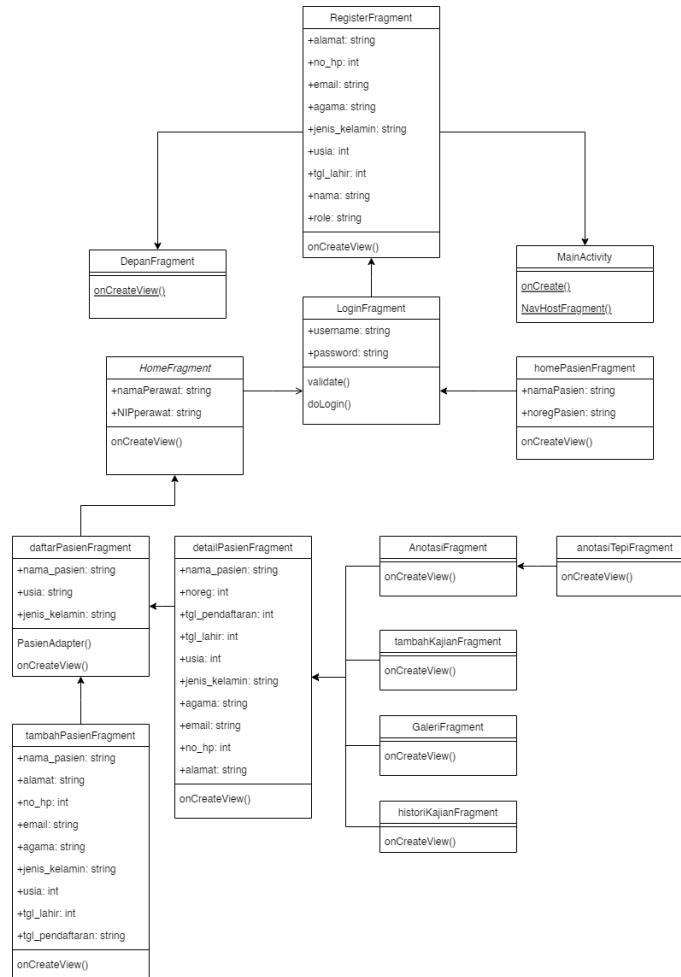
2. Konversi seluruh *activity* menjadi *fragment*

Hal selanjutnya yang dilakukan adalah mengkonversi seluruh activity yang terdapat pada penelitian sebelumnya lalu mengubahnya menjadi fragment. Diawali dengan mencatat seluruh activity yang ada pada penelitian sebelumnya.



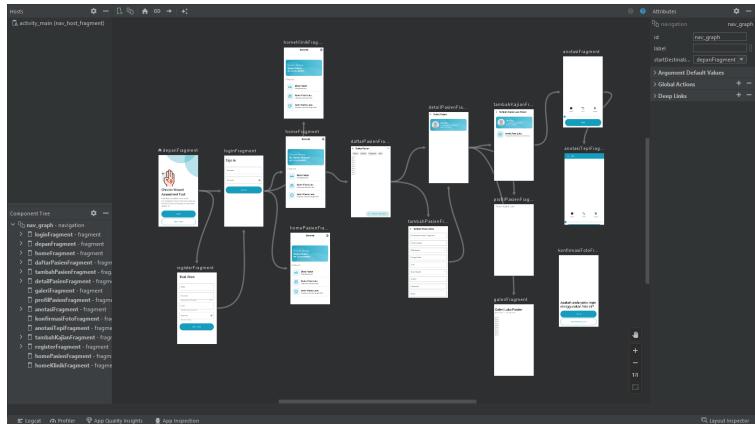
Gambar 4.3: Class Diagram Activity

Setelah itu buat fragment berdasarkan catatan activity yang sudah dilakukan.



Gambar 4.4: Class Diagram Fragment

Lalu untuk menghubungkan antara satu fragment dengan fragment lainnya dibuatlah navGraph agar tersambung secara otomatis.



Gambar 4.5: Navigation Graph

Selanjutnya apabila fragment telah terhubung satu sama lain dilanjutkan dengan implementasi fragment logic dari fragment login yang sudah terintegrasi dengan web service.

Code fragment logic pada fragment login

```

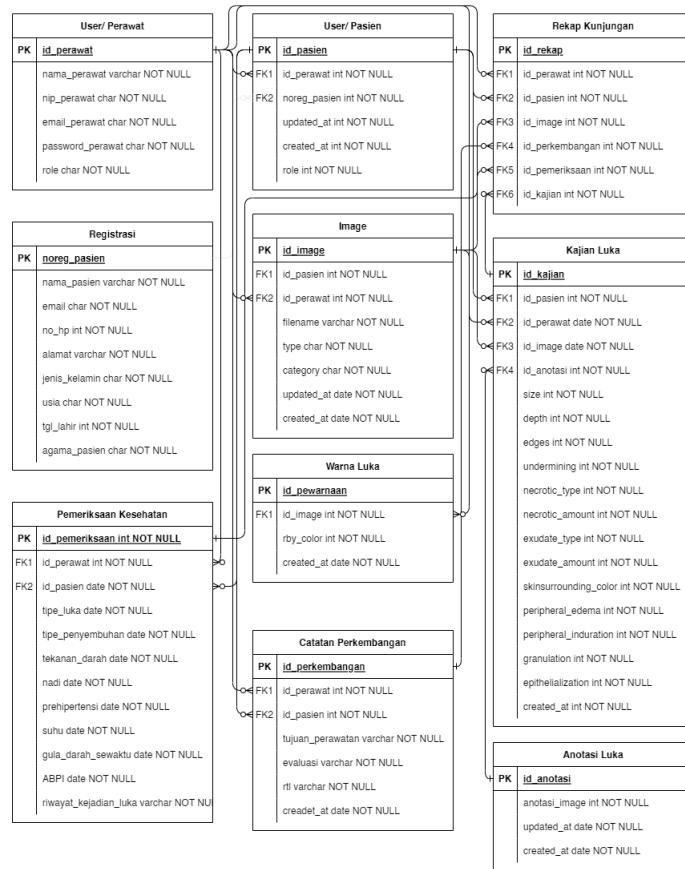
1 override fun onCreateView(
2     inflater: LayoutInflater, container: ViewGroup?,
3     savedInstanceState: Bundle?,
4     ): View? {
5     val root = inflater.inflate(R.layout.fragment_login, container, false)
6     val formLogin = root.findViewById(R.id.formLogin) as FrameLayout
7     val login = root.findViewById(R.id.buttonLogin) as Button
8     val editTextUsername = root.findViewById(R.id.editTextUserName) as EditText
9     val editTextPassword = root.findViewById(R.id.editTextPassword) as EditText
10    val textInputLayoutUsername = root.findViewById(R.id.textInputLayoutUserName)
11        as TextInputLayout
12    val textInputLayoutPassword = root.findViewById(R.id.textInputLayoutPassword)
13        as TextInputLayout
14    formLogin.setVisibility(View.VISIBLE)
15
16    fun validate(username: String?, password: String?): Boolean {
17        if (username == null || username.trim { it <= ' ' }.length == 0) {
18            textInputLayoutUsername.setError("Username tidak boleh kosong")
19            return false
20        }
21        if (password == null || password.trim { it <= ' ' }.length == 0) {
22            textInputLayoutPassword.setError("Password tidak boleh kosong")
23            return false
24        }
25        return true
26    }
27
28    fun doLogin(username: String, password: String) {
29        val retro = ApiService().getInstance().create(WoundApi::class.java)
30
31    }
32
33}

```

```

28     val coroutineExceptionHandler = CoroutineExceptionHandler{_ , throwable ->
29         throwable.printStackTrace()
30     }
31     GlobalScope.launch(Dispatchers.IO + coroutineExceptionHandler) {
32         val result = retro.getLogin(username, password)
33         val testRetro = result.body()
34         if (result.isSuccessful)
35             Log.d("data: ", result.toString())
36             Log.d("body: ", result.body()!!.name)
37             Log.d("test: ", testRetro.toString())
38             SaveSharedPreference.setLoggedIn(requireActivity() .
39                 applicationContext, true)
40             val preferences: SharedPreferences = requireActivity() .
41                 getSharedPreferences("preferences", Context.MODE_PRIVATE)
42             val editor = preferences.edit()
43             editor.putString("name", username)
44             editor.commit()
45             if (result.body()!!.role == "perawat") {
46                 findNavController().navigate(R.id.homeFragment)
47             }
48             else if (result.body()!!.role == "pasien") {
49                 findNavController().navigate(R.id.homePasienFragment)
50             }
51             else if (result.body()!!.role == "klinik") {
52                 findNavController().navigate(R.id.homeKlinikFragment)
53             }
54         }
55         login.setOnClickListener{
56             val username: String = editTextUsername.text.toString()
57             val password: String = editTextPassword.text.toString()
58             //Check user input is correct or not
59             if (validate(username, password)) {
60                 doLogin(username, password)
61             }
62         }
63
64         return root
65     }
66 }
```

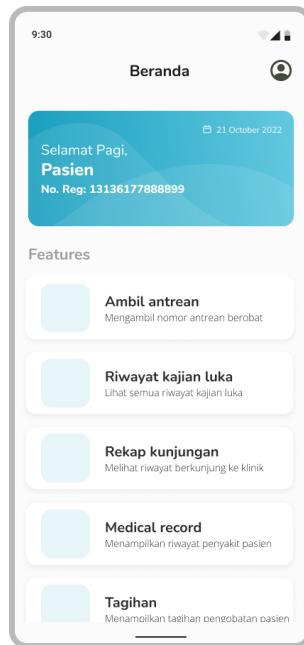
Pada code diatas retrofit pemanggilan juga diubah menjadi courutine. Yang diubah juga selain itu adalah database. Karena sekarang akan menerapkan sistem multiuser, dapat dipastikan bahwa database yang digunakan pun ada perubahan dibagian user. Ditambahkan kolom role pada database agar dapat memudahkan identifikasi user mana perawat dan mana yang pasien.



Gambar 4.6: Entity Relationship Diagram

3. Login User

Setelah mengubah database dengan menambahkan role, maka dilanjutkan dengan membuat dashboard user setelah login. Karena pada penelitian sebelumnya hanya terdapat dashboard perawat setelah login.



Gambar 4.7: Dashboard User

2. Sprint 2

Sprint-2 pada *product backlog* berisi 1 *story* yaitu membuat fitur untuk evaluasi kondisi kesehatan pasien sebelum perawatan. *Story* ini dipecah menjadi beberapa bagian sebagai berikut.

Tabel 4.2: Sprint 2

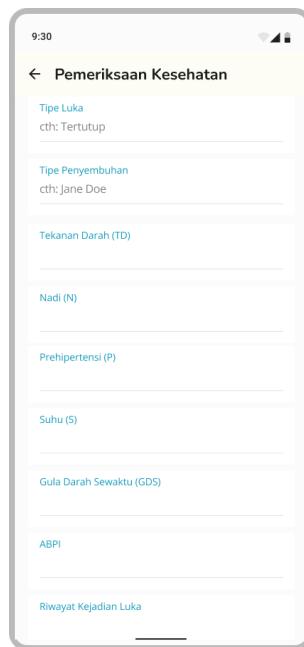
No.	User Story	Task	Status
1.	Evaluasi kondisi kesehatan pasien sebelum perawatan	Buat mock-up pemeriksaan kesehatan sebelum perawatan	selesai
		Pengembangan backend bagian pengecekan kesehatan	selesai
		Pengembangan android bagian yang mengkoneksikan dengan pengecekan kondisi pasien	selesai
		Pembuatan XML Android yang mendefinisikan	selesai

Tujuan dari *sprint-2* ini adalah untuk membuat desain halaman evaluasi

kesehatan, pengembangan *backend* bagian pengecekan kesehatan, lalu pembuatan dan pengembangan *XML Android* terkait bagian ini. Kendala yang dialami oleh penulis ialah mengintegrasikan *code* baru ini kedalam yang sudah ada.

1. *Mock-up* pemeriksaan kesehatan

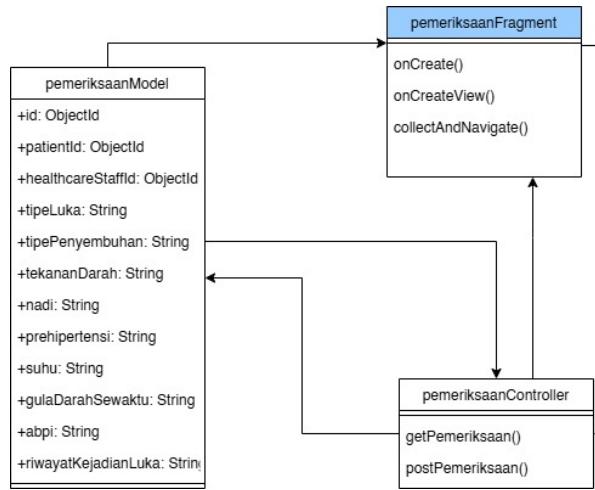
Hal yang pertama dilakukan ketika menjalankan *sprint* ini ialah membuat desain *mock-up* untuk aplikasi yang akan digunakan. *Field-field* yang terdapat pada *mock-up* didapatkan dari form pengkajian luka klinik moist care pada lampiran C.



Gambar 4.8: *Mock-up* Pemeriksaan Kesehatan

2. *Class Diagram*

Pada *class diagram* ini menggambarkan kelas-kelas apa saja yang akan digunakan dalam sistem. Biasanya terdapat 3 kelas yaitu model, view, dan controller. Pada sprint-2 ini penulis membuat 3 kelas yaitu model yang berwarna hijau, view berwarna biru, dan controller berwarna kuning.



Gambar 4.9: Class Diagram Sprint 2

3. Backend bagian pengecekan kesehatan

Sebelum *mock-up UI* sebelumnya diterapkan, pengembangan terhadap backend terkait bagian ini dijalankan. Berikut adalah *code* model dari periksaKesehatan.

```

1 def create_medical_checkup(request):
2     check = get_from_collection("patient_info", {"user_id": ObjectId(request.
3         form["patient_id"])})
4     check = json.loads(bson.json_util.dumps(list(check)))
5     if len(check) == 0:
6         raise Exception("Pasien tidak ditemukan")
7     # data = {
8     #     "patient_id" : ObjectId(request.form["patient_id"])
9     # }
10    check2 = get_from_collection("healthcare_staff_info", {"user_id": ObjectId(
11        request.form["healthcare_staff_id"])})
12    check2 = json.loads(bson.json_util.dumps(list(check2)))
13    if len(check2) == 0:
14        raise Exception("Perawat tidak ditemukan")
15    data = {
16        "patient_id" : ObjectId(request.form["patient_id"]),
17        "healthcare_staff_id" : ObjectId(request.form["healthcare_staff_id"]),
18        "created_at" : time.strftime("%d/%m/%Y %H:%M:%S"),
19        "updated_at" : time.strftime("%d/%m/%Y %H:%M:%S")
20    }
21    nullable = {"tipe_luka": "string",
22                "tipe_penyembuhan": "string",
23                "tekanan_darah": "string",
24                "nadi": "string",
25                "prehipertensi": "string",
26                "suhu": "string",
                "gula_darah_sewaktu": "string",
                "abpi": "string",
  
```

```

27         "riwayat_kajian_luka": "string"}
28     for param in nullable.keys():
29         if request.form.get(param) != '' and request.form.get(param) != "" and
request.form.get(param) != "/" and request.form.get(param) != None:
30             if nullable[param] == "string":
31                 data[param] = request.form[param]
32             elif nullable[param] == "int":
33                 data[param] = int(request.form[param])
34             elif nullable[param] == "date":
35                 data[param] = request.form[param]
36             elif nullable[param] == "float":
37                 data[param] = float(request.form[param])
38             else:
39                 data[param] = None
40     data = insert_medical_checkup(data)
41     return data.inserted_id
42

```

Dalam *code* tersebut dapat diketahui bahwa pertama yang dilakukannya ialah mengecek apakah pasien dan perawat sudah terdaftar dalam server. Lalu dilanjutkan dengan mengisi field yang disediakan oleh *mock-up* sebelumnya dan dimasukkan ke server. Data-data dari field yang disediakan bersifat *nullable* dikarenakan pengecekan ini tidak wajib dilakukan kepada pasien yang sudah pernah berobat.

4. Konfigurasi *controller*

Setelah dibuatnya model periksaKesehatan, dilanjutkan dengan pengembangan API Service untuk dikoneksikan dengan android. *Controller periksaKesehatan*

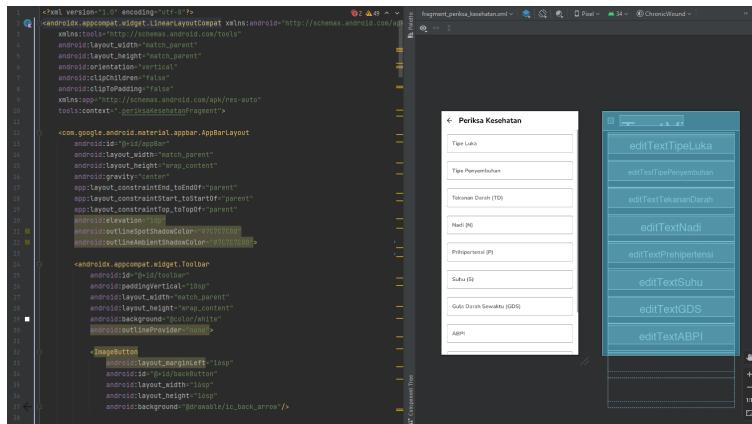
```

1 @bp.route("v1/medical_checkup/", methods=["POST"])
2 def create_medical_checkup():
3     try:
4         return json.dumps({"medical_checkup_id": str(db_pemeriksaan.
create_medical_checkup(request)) })
5     except Exception as ex:
6         print(ex)
7         return Response(response = json.dumps({"message": f"{ex}"}), mimetype
="application/json", status=500)
8

```

5. Pengembangan XML android dan konfigurasi

Bagian terakhir dari *sprint* ini ialah implementasi tampilan *mock-up* yang telah dibuat ke dalam aplikasi dan diintegrasikan dengan backend yang juga telah dibuat sebelumnya.



Gambar 4.10: Fragment Pemeriksaan Kesehatan

Code Konfigurasi

```

1 override fun onCreateView(
2     inflater: LayoutInflater, container: ViewGroup?,
3     savedInstanceState: Bundle?,
4 ): View? {
5     // Inflate the layout for this fragment
6     val root = inflater.inflate(R.layout.fragment_periksa_kesehatan,
7     container, false)
8     val nextButton = root.findViewById(R.id.buttonSubmit) as Button
9     val tipeLuka = root.findViewById(R.id.editTextTipeLuka) as EditText
10    val tipePenyembuhan = root.findViewById(R.id.editTextTipePenyembuhan)
11    as EditText
12    val tekananDarah = root.findViewById(R.id.editTextTekananDarah) as
13    EditText
14    val nadi = root.findViewById(R.id.editTextNadi) as EditText
15    val prehipertensi = root.findViewById(R.id.editTextPrehipertensi) as
16    EditText
17    val suhu = root.findViewById(R.id.editTextSuhu) as EditText
18    val gulaDarahSewaktu = root.findViewById(R.id.editTextGDS) as EditText
19    val abpi = root.findViewById(R.id.editTextABPI) as EditText
20    val riwayatKejadianLuka = root.findViewById(R.id.
21    editTextRiwayatKejadian) as EditText
22    val backButton = root.findViewById(R.id.backButton) as ImageButton
23    val patient_id = arguments?.getString("patient_id")
24    val healthcare_staff_id = arguments?.getString("healthcare_staff_id")
25    val namaPasien = arguments?.getString("namaPasien")
26    val nomorRekamMedis = arguments?.getString("nomorRekamMedis")
27    val usiaPasien = arguments?.getString("usiaPasien")
28    val jenisKelamin = arguments?.getString("jenisKelamin")
29    tipeLuka.setText(arguments?.getString("tipe_luka") ?: "")
30    tipePenyembuhan.setText(arguments?.getString("tipe_penyembuhan") ?: "")
31    tekananDarah.setText(arguments?.getString("tekanan_darah") ?: "")
32    nadi.setText(arguments?.getString("tekanan_nadi") ?: "")
33 }

```

```

28     prehipertensi.setText(arguments?.getString("tekanan_prehipertensi") ?:
29         "")
30     suhu.setText(arguments?.getString("suhu_tubuh") ?: "")
31     gulaDarahSewaktu.setText(arguments?.getString("gula_darah_sewaktu") ?: "")
32     abpi.setText(arguments?.getString("abp_i") ?: "")
33     riwayatKejadianLuka.setText(arguments?.getString(
34         "riwayat_kejadian_luka") ?: "")
35     nextButton.setOnClickListener{
36         val tipe_luka: String = tipeLuka.text.toString()
37         val tipe_penyembuhan: String = tipePenyembuhan.text.toString()
38         val tekanan_darah: String = tekananDarah.text.toString()
39         val tekanan_nadi: String = nadi.text.toString()
40         val tekanan_prehipertensi: String = prehipertensi.text.toString()
41         val suhu_tubuh: String = suhu.text.toString()
42         val gula_darah_sewaktu: String = gulaDarahSewaktu.text.toString()
43         val abp_i: String = abpi.text.toString()
44         val riwayat_kejadian_luka: String = riwayatKejadianLuka.text.
45             toString()
46         println("tipe_luka:$tipe_luka")
47         //Check user input is correct or not
48         if (validate(tipe_luka, tipe_penyembuhan, tekanan_darah,
49             tekanan_nadi, tekanan_prehipertensi, suhu_tubuh, gula_darah_sewaktu, abp_i
50             , riwayat_kejadian_luka)) {
51             //do periksa kesehatan
52             val action = periksaKesehatanFragmentDirections.
53             actionPeriksaKesehatanFragmentToTambahKajianFragment()
54             findNavController().navigate(action)
55         }
56     }
57     return root
58 }
59 }
```

3. Sprint 3

Sprint-3 kali ini sama seperti *sprint* sebelumnya yaitu berisi 1 *story* terkait fitur rekap kunjungan pasien. *Story* ini memiliki beberapa *task* sebagai berikut.

Tabel 4.3: Sprint 3

No.	User Story	Task	Status
1.	Rekap kunjungan pasien	Identifikasi kebutuhan pengguna terkait fitur rekap kunjungan pasien	selesai
		Mendesain struktur database untuk menyimpan data kunjungan pasien	selesai
		Mendesain tampilan rekap kunjungan pasien	selesai
		Mengembangkan tampilan daftar kunjungan pasien di aplikasi	selesai
		Menghubungkan tampilan dengan API untuk mengambil data kunjungan	selesai

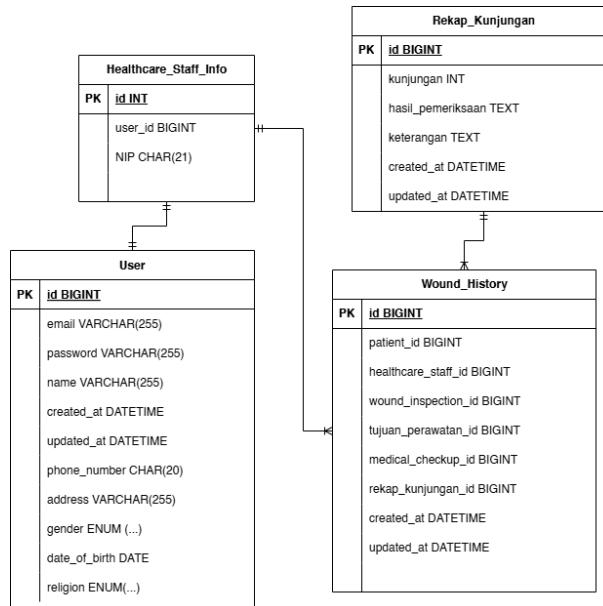
Pada *sprint* ini *story* yang dipilih untuk diuraikan adalah membuat fitur rekap kunjungan pasien. Tujuan dari *sprint* ini ialah untuk membuat fitur rekap kunjungan pasien, pengembangan struktur database, desain tampilan dan implementasi kedalam aplikasi. Kendala yang dialami oleh penulis pada *sprint* ini adalah mengembangkan struktur database yang ada saat ini.

1. Analisis data yang diperlukan untuk menampilkan rekap kunjungan

Identifikasi kebutuhan pengguna terkait fitur rekap kunjungan pasien. Diskusi dengan *scrum master* mengenai format dan informasi yang ditampilkan dalam rekap kunjungan. Serta analisis data yang diperlukan berdasarkan pada lampiran C.

2. Struktur database untuk menyimpan data kunjungan pasien

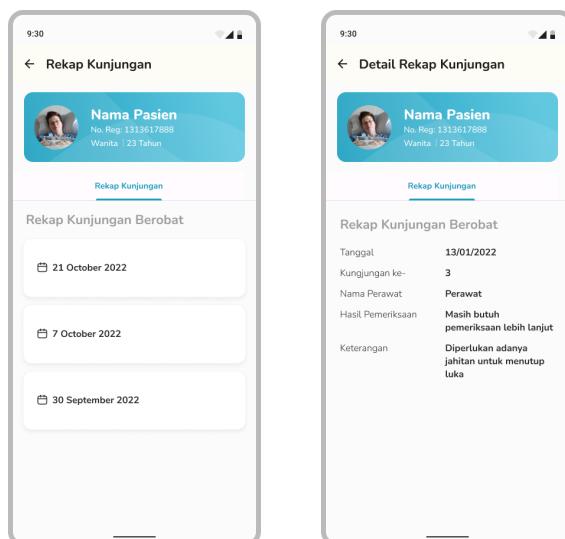
Selanjutnya yaitu mengembangkan database pada fitur rekap kunjungan pasien. Masukkan data rekap kunjungan pasien pada erd yang sudah dirancang.



Gambar 4.11: *ERD Sprint-3*

3. Desain tampilan rekap kunjungan pasien

Setelah database dirancang dilanjutkan dengan membuat tampilan daftar kunjungan pasien di aplikasi. Isi dari konten berdasarkan diskusi dengan *scrum master* dan data pada lampiran C.



Gambar 4.12: *Mock-up Rekap Kunjungan*

4. Pengembangan tampilan daftar kunjungan pasien pada aplikasi dan konfigurasi

Sprint ini diakhiri dengan mengembangkan tampilan daftar kunjungan pasien pada aplikasi dan konfigurasinya. Tampilan rekap kunjungan pasien berdasarkan *mock-up* sebelumnya dan konfigurasi berdasarkan rancangan database yang telah dibuat. *Code konfigurasi rekap kunjungan*

```

1 class pasienRekapKunjunganFragment : Fragment() {
2     private var _binding: FragmentPasienRekapKunjunganBinding? = null
3     private val binding get() = _binding!!
4     private lateinit var patient_id: String
5     private lateinit var recyclerView: RecyclerView
6     private lateinit var adapter: pasienRekapKunjunganAdapter
7     private var kunjunganArrayList: ArrayList<PasienRekapKunjunganItem> =
8         ArrayList()
9
10    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
11        super.onViewCreated(view, savedInstanceState)
12
13        patient_id = arguments?.getString("patient_id")!!
14        val namaPasien = arguments?.getString("nama")
15        val NRMpasiens = arguments?.getString("nrm")
16        binding.backButton.setOnClickListener {
17            val action = pasienRekapKunjunganFragmentDirections.
18                actionPasienRekapKunjunganFragmentToHomePasienFragment()
19            action.arguments.putString("patient_id", patient_id)
20            action.arguments.putString("nama", namaPasien)
21            action.arguments.putString("nrm", NRMpasiens)
22            findNavController().navigate(action)
23        }
24        cariPasien(patient_id)
25        getPasienRekapKunjungan(patient_id)
26        recyclerView = binding.recyclerView
27        recyclerView.layoutManager = LinearLayoutManager(context)
28        adapter = pasienRekapKunjunganAdapter(kunjunganArrayList, patient_id,
29        namaPasien ?: "", NRMpasiens ?: "")
```

4. *Sprint 4*

Pada *sprint-4* ini terdapat 1 *story* yaitu untuk mencatat perkembangan pasien dan diagnosis agar dapat menentukan langkah penyembuhan selanjutnya secara lebih efektif. *Sprint* ini diuraikan menjadi beberapa bagian sebagai berikut.

Tabel 4.4: Sprint 4

No.	User Story	Task	Status
1.	Catatan perkembangan/ diagnosis untuk langkah penyembuhan selanjutnya	<i>Requirement Gathering dan Analysis</i>	selesai
		<i>Database dan API Development</i>	selesai
		<i>UI Design</i>	selesai
		<i>Frontend Development</i>	selesai

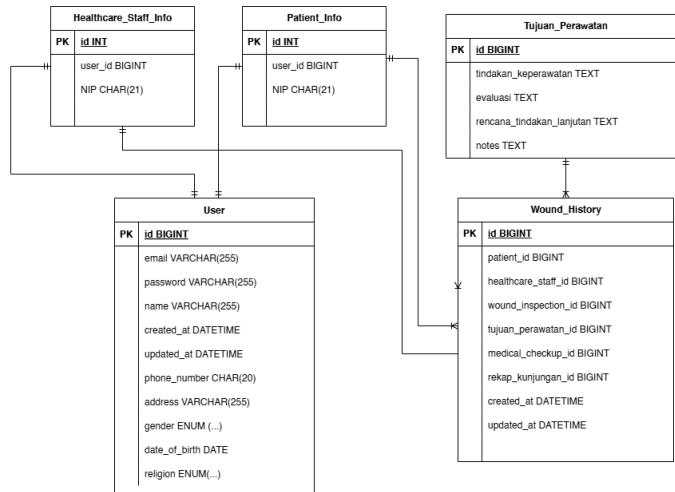
Tujuan *Sprint-4* ini adalah membuat fitur catatan perkembangan dan diagnosis untuk langkah penhemahan selanjutnya. Pada *sprint* ini identifikasi kebutuhan, mendesain struktur database dan implementasi, mendesain tampilan input catatan perkembangan dan diagnosis pasien, serta menghubungkan tampilan dengan API untuk menyimpan dan menampilkan data catatan perkembangan.

1. *Requirement Gathering dan Analysis*

Identifikasi kebutuhan pengguna terkait fitur "Tujuan Perawatan". Diskusi dengan *scrum master* mengenai informasi yang perlu dicatat (diagnosis, perkembangan, tindakan lanjutan). Analisis data yang diperlukan untuk mendukung fitur ini berdasar dari formulir pada lampiran C menjadi pijakan untuk langkah selanjutnya.

2. *Database dan API Development*

Pada *task* ini penulis mendesain struktur database untuk menyimpan catatan perkembangan dan diagnosis. Mengembangkan endpoint API untuk mencatat dan mengambil data tujuan perawatan. dan diakhiri implementasi validasi input untuk memastikan data yang dimasukkan akurat.

**Gambar 4.13:** ERD Sprint-4*Endpoint API*

```

1 @bp.route("v1/tujuan_perawatan/", methods=["POST"])
2 def create_tujuan_perawatan():
3     try:
4         return json.dumps({"tujuan_perawatan_id" : str(db_tujuan_perawatan.
5             create_tujuan_perawatan(request)) })
6     except Exception as ex:
7         print(ex)
8         return Response(response = json.dumps({"message" : f"{ex}"}), mimetype
9             ="application/json", status=500)
10
11 @bp.route("v1/tujuan_perawatan/<id_tujuan_perawatan>", methods=["GET"])
12 def get_tujuan_perawatan_by_id(id_tujuan_perawatan):
13     try:
14         return db_tujuan_perawatan.get_tujuan_perawatan_by_id(
15             id_tujuan_perawatan)
16     except Exception as ex:
17         print(ex)
18         return Response(response = json.dumps({"message" : f"{ex}"}), mimetype
19             ="application/json", status=500)
  
```

Implementasi validasi input

```

1 def create_tujuan_perawatan(request):
2     check  = get_from_collection("patient_info", {"user_id":ObjectId(request.
3         form["patient_id"])})
4     check = json.loads(bson.json_util.dumps(list(check)))
5     if len(check) == 0:
6         raise Exception("Pasien tidak ditemukan")
7     check2  = get_from_collection("healthcare_staff_info", {"user_id":ObjectId(
8         request.form["healthcare_staff_id"])})
9     check2 = json.loads(bson.json_util.dumps(list(check2)))
  
```

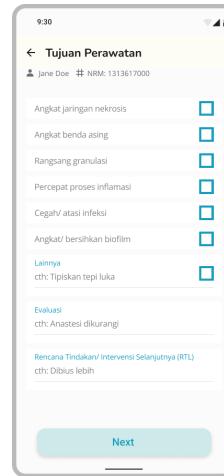
```

8     if len(check2) == 0:
9         raise Exception("Perawat tidak ditemukan")
10    data = {
11        "patient_id" : ObjectId(request.form["patient_id"]),
12        "healthcare_staff_id" : ObjectId(request.form["healthcare_staff_id"]),
13        "created_at" : time.strftime("%d/%m/%Y %H:%M:%S"),
14        "updated_at" : time.strftime("%d/%m/%Y %H:%M:%S")
15    }
16    nullable = {"tindakan_keperawatan": "list",
17                "evaluasi": "string",
18                "rencana_tindakan_lanjutan": "string",
19                "notes": "string"}
20    for param in nullable.keys():
21        if request.form.get(param) != '' and request.form.get(param) != "" and
request.form.get(param) != None and request.form.get(param) != None:
22            if nullable[param] == "string":
23                data[param] = request.form[param]
24            elif nullable[param] == "int":
25                data[param] = int(request.form[param])
26            elif nullable[param] == "float":
27                data[param] = request.form[param]
28            elif nullable[param] == "list":
29                try:
30                    # Parsing JSON ke list
31                    data[param] = json.loads(request.form[param]) # Parsing
JSON ke list
32                    # Verifikasi bahwa data adalah list dan semua elemen dalam
list adalah string
33                    if not isinstance(data[param], list):
34                        raise Exception(f"Format {param} harus berupa array (
list) JSON")
35                    for item in data[param]:
36                        if not isinstance(item, str):
37                            raise Exception(f"Semua elemen dalam {param} harus
berupa string")
38                except json.JSONDecodeError:
39                    raise Exception(f"Format {param} harus berupa array JSON
yang valid")
40            else:
41                data[param] = None
42    data = insert_tujuan_perawatan(data)
43    return data.inserted_id

```

3. UI Design

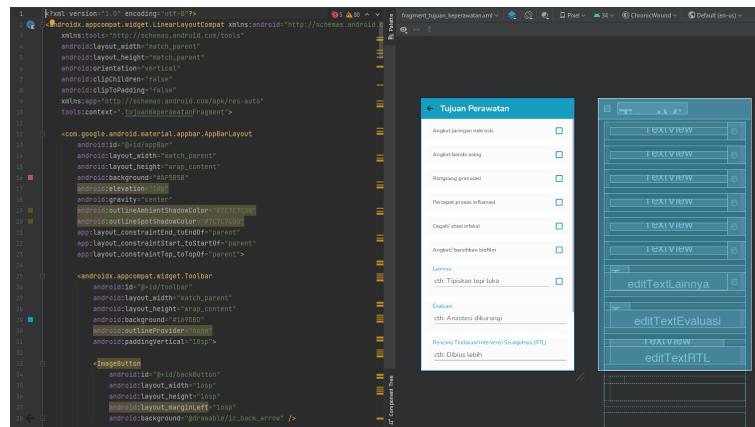
Mendesain tampilan input catatan perkembangan dan diagnosis pasien. Menentukan elemen UI seperti formulir input, daftar catatan sebelumnya, dan tombol navigasi. Serta review desain dengan *scrum master*.



Gambar 4.14: *UI Design Tujuan Perawatan*

4. Frontend Development

Mengembangkan tampilan formulir untuk memasukkan catatan perkembangan. Menampilkan daftar catatan sebelumnya agar tenaga medis dapat melihat riwayat perawatan pasien. Menghubungkan tampilan dengan API untuk menyimpan dan menampilkan data catatan perkembangan.



Gambar 4.15: Fragment Tujuan Perawatan

5. Sprint 5

Pada *Sprint* kali ini terdapat 3 *story* yang berisi implementasi web service program warna luka, implementasi web service anotasi luka, dan mengembangkan progres penyembuhan luka pasien penelitian sebelumnya.

Tabel 4.5: Sprint 5

No.	User Story	Task	Status
1.	Implementasi web service program warna luka	Meninjau kembali hasil penelitian sebelumnya mengenai web service warna luka	tidak selesai
		Menganalisis kebutuhan data warna luka yang akan digunakan dalam sistem	tidak selesai
2.	Implementasi web service program warna luka	Kajian Sistem dan Identifikasi Kebutuhan	selesai
		Menghubungkan API anotasi luka dengan sistem utama	selesai
		Menyesuaikan struktur database untuk menampung informasi anotasi yang lebih kompleks	selesai
		Konfigurasi tampilan anotasi penelitian sebelumnya	selesai
		Menampilkan riwayat anotasi berupa tampilan aplikasi	selesai
3.	Mengembangkan progres penyembuhan luka pasien penelitian sebelumnya	Menentukan parameter utama yang mencerminkan perkembangan penyembuhan luka	selesai
		Merancang skema database yang dapat menyimpan riwayat progres luka pasien	selesai
		Mengembangkan API untuk input dan pengambilan data progres penyembuhan luka	selesai

Tujuan dari *sprint* kali ini ialah agar dapat mengidentifikasi kondisi luka dengan lebih akurat, dapat menandai area spesifik pada luka untuk dokumentasi dan

analisis lebih lanjut, dan melihat perkembangan penyembuhan luka pasien secara berkala agar perawat dapat menilai efektivitas perawatan yang diberikan. Kendala yang dialami penulis pada *sprint* ini ialah belum dapat menampilkan riwayat anotasi pada aplikasi dalam bentuk gambar.

1. Meninjau kembali hasil penelitian sebelumnya mengenai web service warna luka

Setelah ditinjau kembali hasil penelitian sebelumnya, masih terdapat banyak kekurangan yang membuat tidak memungkinkan mengintegrasikan penelitian tersebut kedalam aplikasi. Oleh karena itu *story* mengenai implementasi web service program warna luka ditunda hingga menemukan penelitian yang memungkinkan.

2. Kajian Sistem dan Identifikasi Kebutuhan

Meneliti mekanisme anotasi luka dalam penelitian sebelumnya. Kebutuhan sistem untuk menyimpan dan memproses anotasi luka ialah untuk. Memahami format data anotasi yang digunakan dalam penelitian sebelumnya yaitu berupa koordinat dari gambar. Menyesuaikan format penyimpanan anotasi agar kompatibel dengan sistem yang dikembangkan.

3. Menghubungkan API anotasi luka dengan sistem utama

Memastikan bahwa data anotasi dapat disimpan dan diambil kembali tanpa kendala. Menyesuaikan struktur database untuk menampung informasi anotasi yang lebih kompleks. *Implementasi*

```

1 def create_annotation(path,request):
2     data = {}
3     nullable = {"manual_annotation": "vector_list", "minor_axis": "vector_list"
4                 ", "major_axis": "vector_list","circle_center":"vector","radius": "float"}
5     for param in nullable.keys():
6         if request.form.get(param) != '' and request.form.get(param) != "" and
7             request.form.get(param) !=''' and request.form.get(param) !=None:
8             print(param,request.form.get(param))
9             if nullable[param]== "float":
10                 data[param] = float(request.form[param])
11             elif nullable[param] == "vector_list" or nullable[param] == "
12                 vector":
13                 data[param] = ast.literal_eval(request.form[param])
14             check = ["circle_center","radius"]
15             gate=True
16             for param in check:
17                 if request.form.get(param) != '' and request.form.get(param) != "" and
18                     request.form.get(param) !=''' and request.form.get(param) !=None:
19

```

```

15         continue
16         gate=False
17     if gate:
18         data["automatic_annotation"] = utils.automatic_annotation(path,data["circle_center"][0],data["circle_center"][1],data["radius"])
19     return insert_to_collection("wound_annotation",data)
20
21 def automatic_annotation(path, cc, cr, rad):
22     my_dpi = 96
23     img = imread(path)
24     im_gray = rgb2gray(img)
25     im_gt = imread(path)
26     sigma=3.5
27     sample=400
28     alpha = 0.015
29     beta = 10
30     gamma = 0.001 # time step
31     max_num_iter=500
32     theta = np.linspace(0, 2*np.pi, sample)
33     r = cr + rad*np.sin(theta)
34     c = cc + rad*np.cos(theta)
35     snake_init = np.array([r, c]).T
36     snake_xy = snake_init[:, ::-1]
37     x = snake_xy[:, 0].astype(float)
38     y = snake_xy[:, 1].astype(float)
39     n = len(x)
40     ext = gaussian(im_gray, sigma)
41     ext = img_as_float(ext)
42     ext = ext.astype(float, copy=False)
43     ext = sobel(ext)
44     a = beta
45     b = -(4*beta + alpha)
46     c = 6*beta + 2*alpha
47     eye_n = np.eye(n, dtype=float)
48     c_axis = c * eye_n
49     b_axis = b * ( np.roll(eye_n, -1, axis=0) + np.roll(eye_n, -1, axis=1) )
50     a_axis = a * ( np.roll(eye_n, -2, axis=0) + np.roll(eye_n, -2, axis=1) )
51     A = c_axis + b_axis + a_axis
52     inv = np.linalg.inv(A + gamma * eye_n)
53     inv = inv.astype(float, copy=False)
54     gy, gx = np.gradient(ext)
55     intpl = RectBivariateSpline(np.arange(gx.shape[1]),
56                                 np.arange(gx.shape[0]),
57                                 gx.T, kx=2, ky=2, s=0)
58     intp2 = RectBivariateSpline(np.arange(gy.shape[1]),
59                                 np.arange(gy.shape[0]),
60                                 gy.T, kx=2, ky=2, s=0)
61     max_px_move=1.0
62     xt = np.copy(x)
63     yt = np.copy(y)
64     for i in range(max_num_iter):
65         fx = intpl(xt, yt, dx=0, grid=False).astype(float, copy=False)
66         fy = intp2(xt, yt, dy=0, grid=False).astype(float, copy=False)

```

```

67         xn = np.dot(inv, gamma * xt + fx)
68         yn = np.dot(inv, gamma * yt + fy)
69         dx = max_px_move * np.tanh(xn - xt)
70         dy = max_px_move * np.tanh(yn - yt)
71         xt += dx
72         yt += dy
73     snake_final = np.array([xt, yt]).T
74     return snake_final.tolist()

```

4. Konfigurasi tampilan anotasi penelitian sebelumnya

Pada penelitian Salsa sebelumnya terdapat program anotasi manual yang hasilnya menyimpan gambar anotasi manual tersebut pada aplikasi. Lalu penulis mengkonfigurasi program tersebut menjadi menyimpan titik koordinatnya dan integrasi dengan webservice anotasi luka yang sedang dibuat.

Wound_Annotation	
PK	id INT
	manual_annotation FLOAT
	minor_axis FLOAT
	major_axis FLOAT
	circle_center FLOAT
	radius FLOAT
	automatic_annotation FLOAT

Gambar 4.16: Database *Sprint-5*

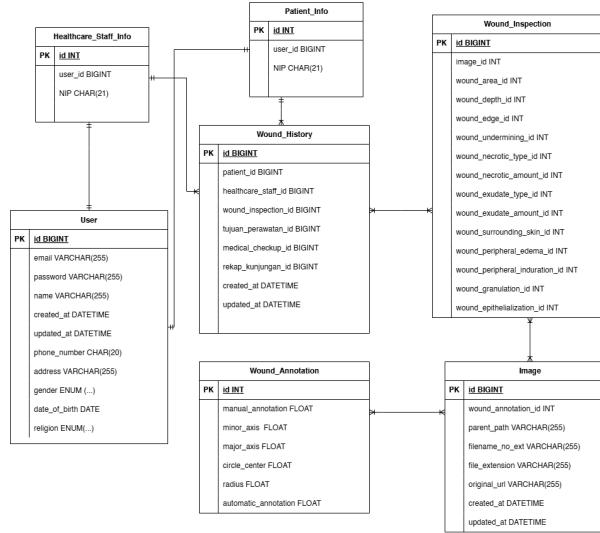
5. Menampilkan riwayat anotasi berupa tampilan aplikasi

Penulis mengalami kendala dalam menjalankan *task* ini dikarenakan waktu yang kurang cukup. Selain itu, akurasi dari program yang dibuat belum akurat. Maka dari itu pada bagian ini butuh pengembangan lebih lanjut.

6. Menentukan parameter utama yang mencerminkan perkembangan penyembuhan luka

Parameter utama yang digunakan pada penelitian ini yaitu berdasarkan BWAT terdapat pada lampiran B. Seluruh isi konten berdasarkan hal tersebut dan telah didiskusikan dengan *scrum master*.

7. Merancang skema database yang dapat menyimpan riwayat progres luka pasien



Gambar 4.17: ERD Sprint-5

6. Sprint 6

Sprint-6 ini terdapat 1 story yaitu membuat fitur *wound history*. Story tersebut diurai menjadi beberapa bagian sebagai berikut.

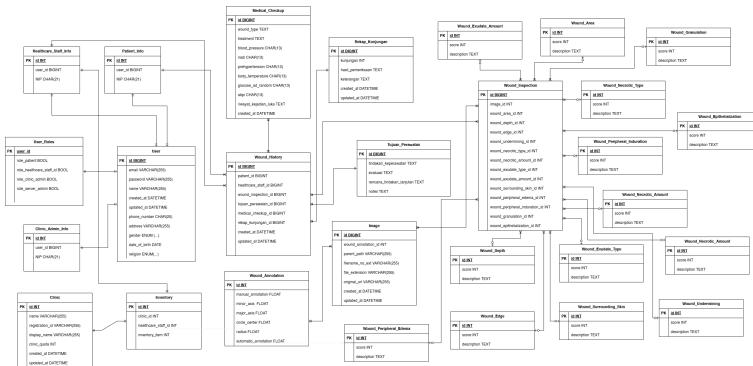
Tabel 4.6: Sprint 6

No.	User Story	Task	Status
1.	<i>Wound History</i>	Analisis dan Perancangan Wound History	selesai
		Implementasi Backend untuk Wound History	selesai
		Implementasi Frontend untuk Wound History	selesai

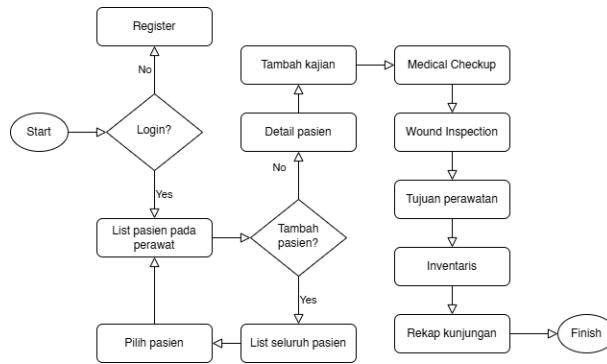
Pada sprint ini membuat fitur *wound history*. Tujuannya agar dapat memantau perkembangan penyembuhan dan menentukan tindakan perawatan yang tepat.

1. Analisis dan Perancangan Wound History

Menganalisis data yang diperlukan dalam riwayat luka. Mendesain struktur database untuk menyimpan data riwayat luka. Membuat alur riwayat luka pasien.



Gambar 4.18: Entity Relation Diagram Sistem Keseluruhan



Gambar 4.19: Flowchart Wound History

2. Implementasi Backend untuk Wound History

Membuat API untuk mengambil dan menyimpan data riwayat luka. Lalu menghubungkan API dengan database sistem. *Implementasi Backend*

```

1 @bp.route("v1/wound_history/", methods=["POST"])
2 def create_wound_history():
3     try:
4         return json.dumps({"message" : str(db_histori_kajian.
5                         create_wound_history(request)) })
6     except Exception as ex:
7         print(ex)
8         return Response(response = json.dumps({"message" : f"{ex}"}), mimetype
9                         ="application/json", status=500)
10
11 @bp.route("v1/wound_history/patient/<patient_id>", methods=["GET"])
12 def get_all_wound_history_by_patient_id(patient_id):
13     try:
14         return db_histori_kajian.get_all_wound_history_by_patient_id(
15             patient_id)
16     except Exception as ex:

```

```
14     print(ex)
15     return Response(response = json.dumps({"message" : f"{ex}"}), mimetype
="application/json", status=500)
```

3. Implementasi Frontend untuk Wound History

Mengembangkan tampilan riwayat luka pasien pada aplikasi, menampilkan daftar riwayat luka, serta mengintegrasikan frontend dengan API yang telah dibuat.

```
1 @GET("v1/wound_history/patient/{patient_id}")
2 suspend fun getAllWoundHistory(
3     @Path("patient_id") patient_id: String
4 ): Response<Collection<WoundHistoryItem>>
5
6 @GET("v1/wound_history/{wound_history_id}")
7 suspend fun getDetailWoundHistory(
8     @Path("wound_history_id") wound_history_id: String
9 ): Response<DetailWoundHistory>
```

7. *Sprint 7*

Pada *Sprint-7* ini terdapat 4 *story* yang berfokus pada fitur pengembangan pasien sebagai berikut.

Tabel 4.7: Sprint 7

No.	User Story	Task	Status
1.	Registrasi akun pasien	Analisis kebutuhan dan perancangan fitur registrasi.	selesai
		Implementasi backend API untuk menangani registrasi	selesai
		Implementasi frontend untuk formulir registrasi dengan validasi input	selesai
1.	Mengambil nomor antrean	Desain UI untuk tampilan pemilihan layanan dan nomor antrean.	selesai
		Integrasi web service backend API untuk pemrosesan antrean pasien	tidak selesai
		Pengembangan frontend untuk mengambil dan menampilkan nomor antrean	tidak selesai
1.	Pasien melihat rekap kunjungan	Merancang tampilan daftar kunjungan	selesai
		Implementasi backend API untuk mengambil data riwayat kunjungan pasien	selesai
		Pengembangan frontend untuk menampilkan riwayat kunjungan dalam format yang mudah dibaca	selesai
1.	Pasien melihat riwayat kajian luka	Desain tampilan daftar riwayat kajian luka untuk pasien.	selesai
		Pengembangan frontend untuk menampilkan riwayat kajian luka	selesai

Sprint ini berfokus pada pengalaman pasien dalam mengakses layanan digital klinik, dari registrasi, antrean, hingga melihat riwayat perawatan mereka. Kendala yang dialami oleh penulis adalah tidak bisanya integrasi dengan web service antrean klinik dikarenakan pada penelitian sebelumnya tidak selesai.

1. Analisis kebutuhan dan perancangan fitur registrasi

Membuat rancangan model fitur register pasien. Dilanjutkan dengan membuat tampilan yang telah dikembangkan berdasarkan rancangan model.



Gambar 4.20: *Flowchart Wound History*

Gambar 4.21: Desain tampilan registrasi pasien

2. Implementasi backend API untuk menangani registrasi

Implementasi yang dilakukan seperti halnya dengan register user baru dengan tambahan *roles* untuk membedakan pasien dan perawat.

```

1 @bp.route("v1/patient", methods=["POST"])
2 def create_patient():
3     try:
4         return json.dumps({"message" : db_user_new.create_patient(request) })
5     except Exception as ex:
6         print(ex)
7         return Response(response = json.dumps({"message" : f"{ex}"}), mimetype
8                     ="application/json", status=500)
9
10    def create_patient(request):
11        data = {
12            "email": request.form["email"],
13            "password": request.form["password"],
14            "updated_at": time.strftime("%d/%m/%Y %H:%M:%S"),
15            "created_at": time.strftime("%d/%m/%Y %H:%M:%S"),
16        }
17        if request.form.get("registration_id") != '' and request.form.get(""
18                           "registration_id") != "" and request.form.get("registration_id") != "''" and
19                           request.form.get("registration_id") != None:
20            pass
21        else:
22            raise Exception("id registrasi tidak ada")

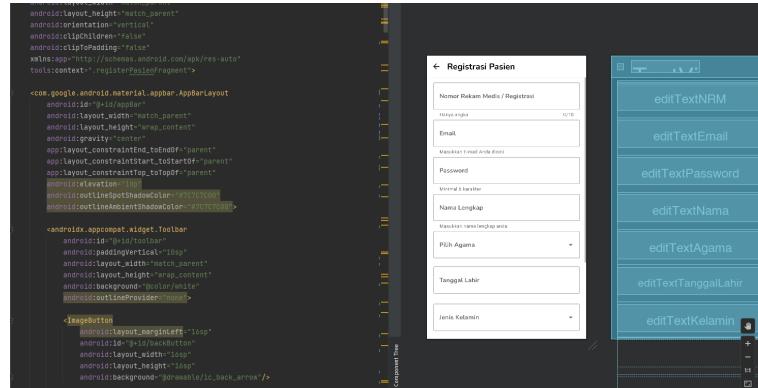
```

```

20     check = get_from_collection("user", {"email":request.form["email"]})
21     check = json.loads(bson.json_util.dumps(list(check)))
22     if len(check) !=0:
23         raise Exception("email yang dimasukkan telah digunakan")
24     nullable = {"name":"string","phone_number": "string","address":"string",
25                 "gender":"string","religion":"string","date_of_birth": "date"}
26     for param in nullable.keys():
27         if request.form.get(param) !='''' and request.form.get(param) !="" and
28             request.form.get(param) !=''' and request.form.get(param) !=None:
29             if nullable[param]==="string":
30                 data[param] = request.form[param]
31             elif nullable[param]==="int":
32                 data[param] = int(request.form[param])
33             elif nullable[param]==="date":
34                 data[param] = request.form[param]
35             else:
36                 data[param] = None
37     user_id = insert_to_collection("user",data).inserted_id
38     data = {
39         "user_id": user_id,
40         "role_patient" : True,
41         "role_healthcare_staff": False,
42         "role_clinic_admin": False,
43         "role_server_admin": False,
44     }
45     insert_to_collection("user_roles", data)
46     data = {
47         "user_id": user_id,
48         "registration_id": request.form["registration_id"],
49         "healthcare_staff_id": [],
50     }
51     insert_to_collection("patient_info", data)
52     return "Berhasil menambahkan pasien baru"

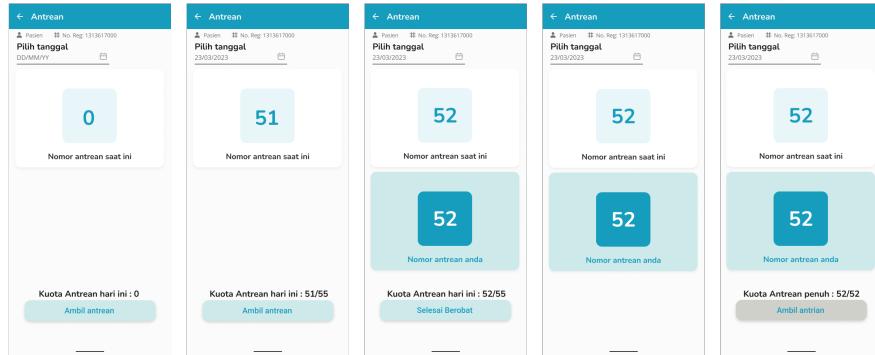
```

3. Implementasi frontend untuk formulir registrasi dengan validasi input



Gambar 4.22: Fragment registrasi pasien

4. Desain UI untuk tampilan mengambil nomor antrean



Gambar 4.23: Fragment registrasi pasien

5. Integrasi web service backend API untuk pemrosesan antrean pasien

Kendala yang dialami oleh penulis adalah belum selesainya web service klinik pada penelitian sebelumnya yang mengakibatkan tidak dapat dilanjutkannya integrasi web service untuk antrean.

6. Merancang tampilan daftar kunjungan

Desain tampilannya sama dengan yang ada pada fitur rekап kunjungan di perawat. Pasien juga dapat menggunakannya.

7. Implementasi backend API untuk mengambil data riwayat kunjungan pasien

Implementasi backend

```

1 @bp.route("v1/rekap_kunjungan/<id_rekap_kunjungan>", methods=["GET"])
2 def get_rekap_kunjungan_by_id(id_rekap_kunjungan):
3     try:
4         return db_rekap_kunjungan.get_rekap_kunjungan_by_id(id_rekap_kunjungan)
5     except Exception as ex:
6         print(ex)
7         return Response(response = json.dumps({"message" : f"{ex}"}), mimetype
8                     ="application/json", status=500)
9
10 @bp.route("v1/rekap_kunjungan/patient/<patient_id>", methods=["GET"])
11 def get_rekap_kunjungan_by_patient_id(patient_id):
12     try:
13         return db_rekap_kunjungan.get_rekap_kunjungan_by_patient_id(patient_id)
14     except Exception as ex:
15         print(ex)

```

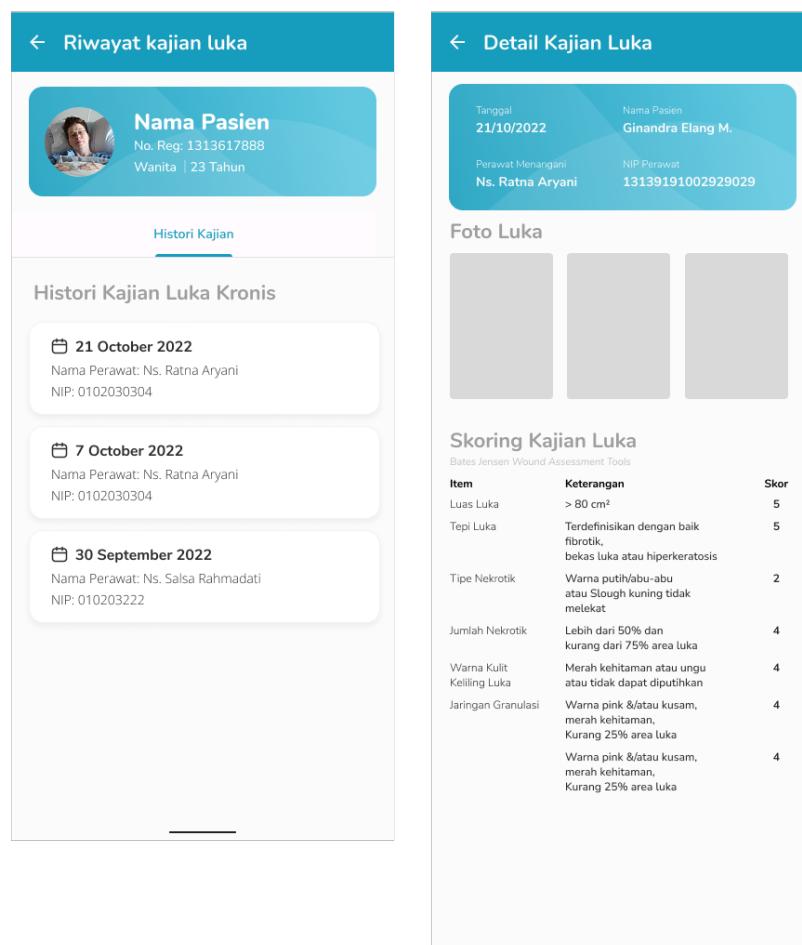
15

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

- Pengembangan frontend untuk menampilkan riwayat kunjungan dalam format yang mudah dibaca

Sama seperti tampilan yang terdapat dalam fitur rekap kunjungan pada perawat. Perbedaannya hanyalah pada beberapa spesifik *field* yang tidak ditampilkan.

- Desain tampilan daftar riwayat kajian luka untuk pasien



Gambar 4.24: Tampilan riwayat luka

- Pengembangan frontend untuk menampilkan riwayat kajian luka

Sama seperti tampilan yang dilihat oleh fitur *wound history* perawat. Tetapi perbedaannya hanya beberapa *field* saja yang ditampilkan.

8. Sprint 8

Sprint-8 ini berisi 1 *story* yaitu membuat fitur inventaris. *Story* tersebut dapat diuraikan lagi menjadi sebagai berikut.

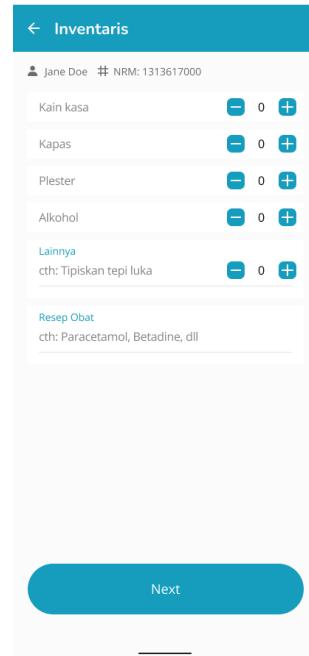
Tabel 4.8: *Sprint 8*

No.	User Story	Task	Status
1.	Inventaris	UI Design	selesai
		Pengembangan struktur database	selesai
		Endpoint API pada fitur inventaris	selesai
		Pengembangan frontend inventaris	selesai

Tujuan dari *sprint-8* adalah untuk mengelola inventaris alat medis dan obat-obatan agar ketersediaan peralatan yang dibutuhkan dalam perawatan pasien dapat dipastikan. Dalam hal ini pengembangan struktur database, membuat *Endpoint API* pada fitur inventaris, dan pengembangan frontend inventaris.

1. UI Design

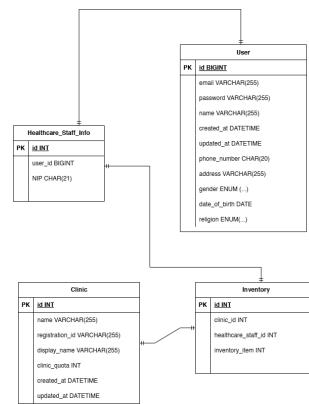
Membuat tampilan daftar inventaris yang mudah diakses oleh perawat.



Gambar 4.25: Tampilan inventaris

2. Pengembangan struktur database

Berikut adalah rancangan ERD terkait fitur inventaris yang sedang dikembangkan.



Gambar 4.26: ERD Sprint-8

3. Endpoint API pada fitur inventaris

Implementasi pada backend untuk membuat endpoint API menambahkan, memperbarui, dan menghapus data inventaris.

```

1 e@bp.route("v1/inventaris/", methods=["POST"])
2 def create_inventaris():
3     try:
4         return json.dumps({"inventaris_id" : str(db_inventaris.
5             create_inventaris(request)) })
6     except Exception as ex:
7         print(ex)
8         return Response(response = json.dumps({"message" : f"{ex}"}), mimetype
9             ="application/json", status=500)
10
11 @bp.route("v1/inventaris/<id_inventaris>", methods=["GET"])
12 def get_inventaris_by_id(id_inventaris):
13     try:
14         return db_inventaris.get_inventaris_by_id(id_inventaris)
15     except Exception as ex:
16         print(ex)
17         return Response(response = json.dumps({"message" : f"{ex}"}), mimetype
18             ="application/json", status=500)

```

4. Pengembangan frontend inventaris

Membuat tampilan daftar inventaris sebelumnya yang mudah diakses oleh perawat kedalam bentuk fragment.

```

1 class InventarisFragment : Fragment() {
2     private var _binding: FragmentInventarisBinding? = null
3     private val binding get() = _binding!!
4
5     override fun onCreateView(
6         inflater: LayoutInflater, container: ViewGroup?, savedInstanceState:
7         Bundle?
8     ): View {
9         _binding = FragmentInventarisBinding.inflate(inflater, container,
10             false)
11         return binding.root
12     }
13
14     override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
15         super.onViewCreated(view, savedInstanceState)
16
17         // Setup counter for each inventory item
18         setupCounter(binding.imageButtonAddKainKasa, binding.
19             imageButtonMinusKainKasa, binding.editTextNumberKainKasa)
20         setupCounter(binding.imageButtonAddKapas, binding.
21             imageButtonMinusKapas, binding.editTextNumberKapas)
22         setupCounter(binding.imageButtonAddPlester, binding.
23             imageButtonMinusPlester, binding.editTextNumberPlester)
24         setupCounter(binding.imageButtonAddAlkohol, binding.
25             imageButtonMinusAlkohol, binding.editTextNumberAlkohol)
26         setupCounter(binding.imageButtonAddInventarisLainnya, binding.
27             imageButtonMinusInventarisLainnya, binding.editTextNumberInventarisLainnya
28     )

```

```

21
22     binding.buttonSubmit.setOnClickListener {
23         sendDataToServer()
24     }
25 }
26
27 private fun setupCounter(addButton: ImageButton, minusButton: ImageButton,
28     editText: EditText) {
29     addButton.setOnClickListener {
30         val value = editText.text.toString().toIntOrNull() ?: 0
31         if (value < 99) editText.setText((value + 1).toString())
32     }
33
34     minusButton.setOnClickListener {
35         val value = editText.text.toString().toIntOrNull() ?: 0
36         if (value > 0) editText.setText((value - 1).toString())
37     }
38
39 private fun sendDataToServer() {
40     val kainKasa = binding.editTextNumberKainKasa.text.toString()
41     val kapas = binding.editTextNumberKapas.text.toString()
42     val plester = binding.editTextNumberPlester.text.toString()
43     val alkohol = binding.editTextNumberAlkohol.text.toString()
44     val lainnya = binding.editTextNumberInventarisLainnya.text.toString()
45     val resepObat = binding.editTextResepObat.text.toString()
46
47     val action = InventarisFragmentDirections.
48         actionInventarisFragmentToRekapKunjunganFragment()
49     findNavController().navigate(action)
50 }
51
52 override fun onDestroyView() {
53     super.onDestroyView()
54     _binding = null
55 }

```

9. Sprint 9

Pada *Sprint* ini terdapat 1 *story* yang berisi tentang fitur menambah pasien ke dalam akun perawat. *Story* tersebut diurai menjadi beberapa bagian sebagai berikut.

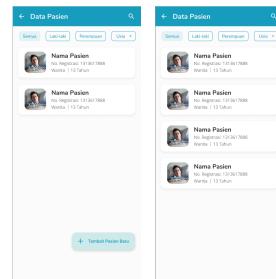
Tabel 4.9: Sprint 9

No.	User Story	Task	Status
1.	Menambah pasien ke dalam akun perawat	UI Design	selesai
		Pengembangan backend	selesai
		Pengembangan frontend berdasarkan desain	selesai

Tujuan dari *sprint* ini adalah dapat menambahkan pasien ke dalam akun perawat agar perawat bisa melihat daftar pasien yang pernah ditangani dan memantau perkembangan perawatan. Kendala yang penulis alami pada *sprint* kali ini adalah proses pengembangan backend.

1. UI Design

Mengumpulkan kebutuhan terkait pencatatan pasien yang pernah ditangani berdasar kartu kunjungan klinik pada lampiran C. Menentukan data pasien yang akan ditampilkan dalam akun perawat (nama, NRM, riwayat perawatan, dsb.).

**Gambar 4.27: UI Design Sprint-9**

2. Pengembangan backend

Membuat endpoint API untuk menambahkan pasien ke dalam daftar akun perawat. Mengembangkan fitur untuk menghubungkan pasien dengan akun perawat berdasarkan sesi perawatan. Menyediakan fitur untuk memperbarui daftar pasien dalam akun perawat jika diperlukan. *Endpoint API*

```

1 @bp.route("v1/healthcare_staff/patient", methods=["PUT"])
2 def insert_patient_to_healthcare_staff():
3     try:

```

```

4     return json.dumps({"message" : db_user_new.
5 insert_patient_to_healthcare_staff(request) })
6 except Exception as ex:
7     print(ex)
8     return Response(response = json.dumps({"message" : f"{ex}"}), mimetype
9 ="application/json", status=500)

```

Menghubungkan pasien dengan akun perawat

```

1 def insert_patient_to_healthcare_staff(reqeust):
2     patient = get_from_collection("user", {"_id": ObjectId(reqeust.form[
3         "patient_id"])})
4     patient = json.loads(bson.json_util.dumps(list(patient)))
5     if len(patient) == 0:
6         raise Exception(f"pasien dengan id pasien {rqeuest.form['patient_id']} tidak ditemukan")
7     patient = patient[0]
8     healthcare_staff = get_from_collection("user", {"_id": ObjectId(reqeust.
9         form["healthcare_staff_id"])})
10    healthcare_staff = json.loads(bson.json_util.dumps(list(healthcare_staff)))
11    if len(healthcare_staff) == 0:
12        raise Exception(f"perawat dengan id perawat {rqeuest.form['
13            healthcare_staff_id']} tidak ditemukan")
14    healthcare_staff = healthcare_staff[0]
15    patient_info = get_from_collection("patient_info", {"user_id": ObjectId(
16        patient["_id"]["$oid"])})
17    patient_info = json.loads(bson.json_util.dumps(list(patient_info)))
18    if len(patient_info) == 0:
19        raise Exception(f"user dengan id pasien {rqeust.form['patient_id']} bukan pasien")
20    patient_info = patient_info[0]
21    healthcare_staff_info = get_from_collection("healthcare_staff_info", {""
22        "user_id": ObjectId(healthcare_staff["_id"]["$oid"])})
23    healthcare_staff_info = json.loads(bson.json_util.dumps(list(
24        healthcare_staff_info)))
25    if len(healthcare_staff_info) == 0:
26        raise Exception(f"user dengan id perawat {rqeust.form['
27            healthcare_staff_id']} bukan perawat")
28    healthcare_staff_info = healthcare_staff_info[0]
29    patient_info["healthcare_staff_id"] = [
30        ObjectId(item["$oid"]) if isinstance(item, dict) and "$oid" in item
31        else ObjectId(item)
32        for item in patient_info["healthcare_staff_id"]
33    ]
34    patient_info["healthcare_staff_id"].append(ObjectId(rqeust.form["
35        healthcare_staff_id"]))
36    healthcare_staff_info["patient_id"] = [
37        ObjectId(item["$oid"]) if isinstance(item, dict) and "$oid" in item
38        else ObjectId(item)
39        for item in healthcare_staff_info["patient_id"]
40    ]
41    healthcare_staff_info["patient_id"].append(ObjectId(rqeust.form["

```

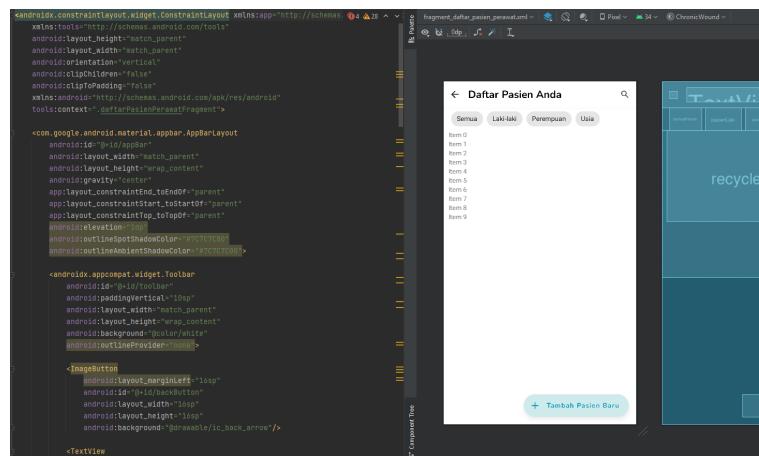
```

32     patient_id"))
33     update_from_collection("patient_info", patient_info["_id"]["$oid"], {"$oid": patient_info["healthcare_staff_id"]})
34     update_from_collection("healthcare_staff_info", healthcare_staff_info["_id"]["$oid"], {"patient_id": healthcare_staff_info["patient_id"]})
35
36     return "Berhasil menambah pasien ke list perawat"

```

3. Pengembangan frontend berdasarkan desain

Membuat tampilan fragment daftar pasien dalam akun perawat agar mempermudah akses perawat terhadap pasien yang pernah dirawat.



Gambar 4.28: Fragment daftar pasien

B. Hasil Pengujian

Pengujian aplikasi dilakukan melalui dua jenis metode, yaitu *Unit Testing* dan User Acceptance Test (UAT). Pengujian ini dilaksanakan setelah seluruh User Story dalam Product Backlog berhasil diimplementasikan. Unit Testing dilakukan oleh satu pengembang internal, sementara User Acceptance Test dilakukan oleh seorang penguji ahli atau perawat untuk memastikan kesesuaian aplikasi dengan kebutuhan pengguna.

1. Unit Testing

Hasil dari *Unit Testing* yang telah disiapkan untuk satu developer internal menunjukkan bahwa pengujian dilakukan setelah suatu fitur dinyatakan selesai. Namun, pengujian ini belum dilaksanakan secara langsung, melainkan hanya

menampilkan skenario pengujinya. Secara keseluruhan, *unit testing* dirancang untuk memastikan bahwa fitur yang dikembangkan dapat berfungsi dengan baik. Rincian skenario tersebut sebagai berikut.

Tabel 4.10: Unit Testing Halaman Awal.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat aplikasi dibuka akan muncul tampilan awal dengan pilihan "Login", "Buat Akun", dan "Buat Akun Pasien"	✓		Diterima
Jika tombol "Login" ditekan, guest akan masuk ke halaman Login	✓		Diterima
Jika tombol "Buat Akun" ditekan, pengguna akan masuk ke halaman register perawat	✓		Diterima
Jika tombol "Buat Akun Pasien" ditekan, pengguna akan masuk ke halaman register pasien	✓		Diterima

Tabel 4.11: Unit Testing Halaman Buat Akun.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika mengisi form registrasi dengan lengkap kemudian submit, maka akan kembali ke halaman Login	v		Diterima
Ketika mengisi form registrasi tidak lengkap kemudian submit, maka akan menampilkan pesan kesalahan untuk melengkapi data yang kosong	✓		Diterima

Tabel 4.12: Unit Testing Halaman Buat Akun Pasien.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika mengisi form registrasi dengan lengkap kemudian submit, maka akan kembali ke halaman Login	✓		Diterima
Ketika mengisi form registrasi tidak lengkap kemudian submit, maka akan menampilkan pesan kesalahan untuk melengkapi data yang kosong	✓		Diterima

Tabel 4.13: Unit Testing Halaman Login.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika mengisi form login dengan data yang tidak sesuai kemudian klik submit, maka akan menampilkan pesan kesalahan username atau password	v		Diterima
Ketika mengisi form login dengan data yang sesuai dengan role perawat, maka akan masuk ke halaman beranda perawat	✓		Diterima
Ketika mengisi form login dengan data yang sesuai dengan role pasien, maka akan masuk ke halaman beranda pasien	✓		Diterima

Tabel 4.14: Unit Testing Beranda Perawat.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat ikon profil pada kanan atas di-klik maka akan muncul halaman profil perawat	✓		Diterima
Saat tombol daftar pasien di-klik maka akan muncul halaman daftar pasien	✓		Diterima

Tabel 4.15: Unit Testing Profil Perawat.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat halaman profil perawat muncul maka akan tampil informasi umum perawat dan tombol logout	✓		Diterima
Saat tombol logout di-klik maka akan kembali ke halaman login	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman inventaris	✓		Diterima

Tabel 4.16: Unit Testing Halaman Daftar Pasien.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat daftar pasien dibuka, maka yang tampil hanya daftar pasien yang terdaftar pada perawat	✓		Diterima
Saat salah satu pasien dipilih maka akan muncul halaman detail pasien	✓		Diterima
Saat tombol tambah pasien ditekan maka akan muncul halaman daftar seluruh pasien	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman beranda perawat	✓		Diterima

Tabel 4.17: Unit Testing Halaman Daftar Seluruh Pasien.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat daftar seluruh pasien dibuka, maka seluruh akun pasien yang telah register akan tampil	✓		Diterima

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat salah satu pasien dipilih maka akan muncul halaman daftar pasien	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman daftar pasien	✓		Diterima

Tabel 4.18: Unit Testing Halaman Detail Pasien.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat masuk pertama kali ke halaman detail pasien, maka akan muncul tab profil pasien dan histori kajian	✓		Diterima
Saat tab Profil Pasien diklik maka akan muncul informasi umum pasien	✓		Diterima
Saat tab Histori Kajian diklik maka akan muncul histori kajian pasien	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman daftar pasien	✓		Diterima

Tabel 4.19: Unit Testing Halaman Histori Kajian.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat masuk pertama kali ke halaman histori kajian, maka akan muncul daftar riwayat kajian dan tombol tambah kajian	✓		Diterima
Saat salah satu riwayat kajian diklik maka akan muncul halaman detail histori kajian	✓		Diterima
Saat tambah kajian diklik maka akan muncul halaman periksa kesehatan	✓		Diterima

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman daftar pasien	✓		Diterima

Tabel 4.20: Unit Testing Halaman Periksa Kesehatan.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika mengisi form periksa kesehatan tidak lengkap kemudian submit, maka akan menampilkan pesan kesalahan untuk melengkapi data yang kosong	✓		Diterima
Ketika mengisi form periksa kesehatan dengan lengkap kemudian submit, maka akan pergi ke halaman tambah kajian	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman detail pasien	✓		Diterima

Tabel 4.21: Unit Testing Halaman Tambah Kajian.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika belum ada foto luka, foto anotasi tepi luka, dan foto diameter luka, maka muncul tombol ambil foto luka	✓		Diterima
Ketika tombol ambil foto luka ditekan, maka akan pergi ke activity camera dan mengambil foto lalu ke halaman anotasi tepi luka	✓		Diterima
Ketika sudah ada foto luka, foto anotasi tepi luka, dan foto diameter luka, maka muncul form penilaian kajian luka dan tombol submit	✓		Diterima

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika mengisi form penilaian kajian luka dengan lengkap kemudian submit, maka akan pergi ke halaman tujuan perawatan	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman periksa kesehatan	✓		Diterima

Tabel 4.22: Unit Testing Halaman Tujuan Perawatan.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika mengisi form tujuan perawatan kemudian submit, maka akan pergi ke halaman inventaris	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman tambah kajian	✓		Diterima

Tabel 4.23: Unit Testing Halaman inventaris.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika mengisi form inventaris kemudian submit, maka akan pergi ke halaman rekap kunjungan	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman tujuan perawatan	✓		Diterima

Tabel 4.24: Unit Testing Halaman Rekap Kunjungan.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika mengisi form rekap kunjungan kemudian submit, maka akan pergi ke halaman detail pasien	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman inventaris	✓		Diterima

Tabel 4.25: Unit Testing Beranda Pasien.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat ikon profil pada kanan atas di-klik maka akan muncul halaman profil pasien	✓		Diterima
Saat tombol riwayat kajian di-klik maka akan muncul halaman daftar riwayat kajian	✓		Diterima
Saat tombol rekap kunjungan di-klik maka akan muncul halaman daftar rekap kunjungan	✓		Diterima

Tabel 4.26: Unit Testing Profil Pasien.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat halaman profil pasien muncul maka akan tampil informasi umum pasien dan tombol logout	✓		Diterima
Saat tombol logout di-klik maka akan kembali ke halaman login	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman login	✓		Diterima

Tabel 4.27: Unit Testing Riwayat Kajian Pasien.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat halaman riwayat kajian pasien muncul maka akan tampil daftar seluruh riwayat kajian pasien tersebut	✓		Diterima
Ketika salah satu daftar riwayat kajian pasien tersebut diklik maka akan muncul halaman detail riwayat kajian pasien	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman beranda pasien	✓		Diterima

Tabel 4.28: Unit Testing Detail Riwayat Kajian Pasien.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat halaman detail riwayat kajian pasien muncul maka akan tampil informasi umum riwayat kajian pasien	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman riwayat kajian	✓		Diterima

Tabel 4.29: Unit Testing Rekap Kunjungan Pasien.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat halaman rekap kunjungan pasien muncul maka akan tampil daftar seluruh riwayat kunjungan pasien tersebut	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman beranda pasien	✓		Diterima

Tabel 4.30: Unit Testing Detail Rekap Kunjungan Pasien.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat halaman detail rekap kunjungan pasien muncul maka akan tampil informasi umum rekap kunjungan pasien	✓		Diterima
Ketika ikon panah ke arah kiri pada kiri atas ditekan, maka akan kembali ke halaman rekap kunjungan pasien	✓		Diterima

2. User Acceptance Testing

User Acceptance Test terhadap perawat dilaksanakan pada tanggal 7 Februari 2025 secara luring bertempat di Kabupaten Bogor. Adapun hasil UAT yang telah dilaksanakan dapat dilihat pada tabel dibawah ini.

Tabel 4.31: *User Acceptance Test*

User Acceptance Test				
No	Acceptance Requirements	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
1	Antarmuka aplikasi terasa nyaman, mudah dipahami, dan telah sesuai dengan kebutuhan pengguna	✓		Diterima
2	Fitur login sudah sesuai dengan kebutuhan pengguna	✓		Diterima
3	Fitur registrasi perawat/ buat akun sudah sesuai dengan kebutuhan perawat	✓		Diterima
4	Fitur registrasi pasien/ buat akun pasien sudah sesuai dengan kebutuhan pasien	✓		Diterima

User Acceptance Test				
No	Acceptance Requirements	Kesesuaian		Kesimpulan
		Sesuai	Tidak Sesuai	
5	Fitur profil perawat sudah sesuai dengan kebutuhan perawat	✓		Diterima
6	Fitur lihat daftar pasien sudah sesuai dengan kebutuhan perawat		✓	Butuh Revisi
7	Fitur tambah pasien ke dalam list daftar perawat sudah sesuai dengan kebutuhan perawat	✓		Diterima
8	Fitur lihat detail pasien sudah sesuai dengan kebutuhan perawat		✓	Butuh Revisi
9	Fitur lihat histori perawatan luka sudah sesuai dengan kebutuhan perawat	✓		Diterima
10	Fitur tambah kajian baru sudah sesuai dengan kebutuhan perawat	✓		Diterima
11	Fitur pencatatan kesehatan sebelum perawatan sudah sesuai dengan kebutuhan perawat	✓		Diterima
12	Fitur rencana tindakan lanjutan sudah sesuai dengan kebutuhan perawat	✓		Diterima
13	Fitur inventaris sudah sesuai dengan kebutuhan perawat		✓	Butuh Revisi
14	Fitur rekap kunjungan sudah sesuai dengan kebutuhan perawat	✓		Diterima
15	Fitur galeri luka pasien sudah sesuai dengan kebutuhan perawat	✓		Diterima
16	Fitur profil pasien sudah sesuai dengan kebutuhan pasien	✓		Diterima
17	Fitur lihat riwayat kajian luka pasien sesuai dengan kebutuhan pasien		✓	Butuh Revisi
18	Fitur lihat riwayat rekap kunjungan pasien sesuai dengan kebutuhan pasien	✓		Diterima

Berdasarkan tabel diatas, terdapat tiga fitur yang memerlukan revisi diantaranya:

1. Halaman daftar pasien

Pada halaman daftar pasien sebaiknya ditambahkan fitur untuk filter pasien untuk memudahkan perawat dalam melakukan pencarian pasien.

2. Halaman detail pasien

Pada halaman detail kajian pasien sebaiknya ditambahkan kolom jenis pembayaran dan tampilkan foto luka sebelumnya agar lebih informatif.

3. Halaman inventaris

Pada halaman inventaris sebaiknya list item pada halaman ini berbentuk paket yang disediakan oleh klinik untuk digunakan.

4. Halaman riwayat kajian luka pada pasien

Pada halaman riwayat kajian luka pada pasien sebaiknya tidak seluruh *medical record* yang ditampilkan melainkan cukup gambar luka saja yang ditampilkan.

3. Kesimpulan Pengujian

Pengujian aplikasi dilakukan menggunakan dua metode, yaitu unit testing dan User Acceptance Test (UAT). Berdasarkan uraian di atas, seluruh skenario dalam unit testing terhadap satu developer internal berjalan dengan baik. Sedangkan dengan UAT yang dirancang untuk perawat atau penguji ahli, terdapat beberapa masukkan seperti penambahan fitur filter pasien pada daftar pasien, penambahan jenis pembayaran pada detail pasien, pengubahan bentuk konten dari satuan menjadi grup paket pada inventaris, dan hanya menampilkan gambar luka saja pada riwayat kajian luka bagian pasien. Dengan pengujian yang telah dilakukan maka dapat disimpulkan bahwa sistem yang dirancang telah lulus uji.

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil implementasi aplikasi yang telah dirancang, maka diperoleh kesimpulan sebagai berikut:

1. Aplikasi pengkajian luka kronis versi kedua berbasis Android berhasil dikembangkan dengan mengintegrasikan berbagai fitur dan hasil penelitian sebelumnya ke dalam Product Backlog. Proses perancangannya mengikuti metode Scrum, yang mencakup tahapan penyusunan Product Backlog, Sprint Backlog, serta pelaksanaan pengembangan dalam sembilan Sprint.
2. Dengan menerapkan Fragment, aplikasi dapat memiliki tampilan yang lebih dinamis dan fleksibel tanpa harus membuat banyak Activity, sehingga mengurangi overhead dan meningkatkan performa. Selain itu, Fragment memungkinkan penggunaan kembali komponen UI di berbagai bagian aplikasi, mendukung navigasi yang lebih lancar, serta memudahkan implementasi desain responsif, terutama dalam aplikasi berbasis Android modern.

B. Saran

Adapun saran untuk penelitian selanjutnya adalah:

1. Berdasarkan diskusi dengan product owner, harus dimulainya pengembangan sistem berikutnya yang memiliki fitur tagihan agar dapat membantu mempercepat antrean pasien.
2. Berdasarkan diskusi dengan scrum master, perlu ditambahkan feedback dari pelayanan yang dijalankan oleh klinik setelah pasien selesai berobat, apakah pelayanan yang dilakukan baik atau buruk.

DAFTAR PUSTAKA

- Aryani, R. (2016). Accelerating wound healing process by using moist dressing.
- DeveloperAndroid (2023).
- IndonesianWoundOstomyContinenceNursesAssociation (2023).
- Khairunnisa, A. (2021). Pengaruh penggunaan color model lab dalam kalibrasi warna luka menggunakan metode segmentasi k-means dan mean shift.
- Rahmadati, S. (2023). Rancang bangun aplikasi dan web service pengkajian luka kronis khususnya modul pengolahan citra berbasis android.
- Ratna Aryani, Muhammad Yusro, M. E. S. I. F. (2018). *Rancang Bangun Aplikasi Mobile Android Sebagai Alat Deteksi Warna Dasar Luka Dalam Membantu Proses Pengkajian Luka Kronis Dengan Nekrosis*. Jakarta.
- Ratna Aryani, U. N. (2017). Autolytic & conservative sharp wound debridement for granulation tissue on unstageable diabetic foot ulcer. *IJINNA*, 1(1):80–87.
- Rizki, M. (2022). Deteksi keliling luka menggunakan active contour.
- Setacci C, Benevento D, D. D. G. V. E. B. U. D. G. L. P. G. S. F. (2015). Wound assessment tools and nurses' needs: an evaluation study. *Int Wound J*, 12(3):293–301.
- Setacci C, Benevento D, D. D. G. V. E. B. U. D. G. L. P. G. S. F. (2020). Focusing on diabetic ulcers. *Transl Med UniSa*, 21(3):7–9.
- Square (2023). Retrofit.
- Sussman, C. dan Bates-Jensen, B. (2012). *Wound Care: A Collaborative Practice Manual for Health Professionals*. Wolters Kluwer Health/ Lippincott Williams & Wilkins, Philadelphia.
- Titus, A., Singerji, A., dan Raj, D. D. (2017). A smartphone based wound assessment system for patients with diabetes using accelerated mean shift algorithm. *International Journal of Information Technology & Decision Making*, 6(05).

- Wang, S., Zhang, Q., Huang, W., Tian, H., Hu, J., Cheng, Y., dan Peng, Y. (2018). A new smart mobile system for chronic wound care management. *IEEE Access*, 6:52355–52365.
- WorldHealthOrganization (2022). Leading causes of death and disability 2000-2019: A visual summary.

LAMPIRAN A

Transkrip Percakapan

Hari: Jumat

Tanggal: 24 Februari 2023

PL: Penulis

N: Narasumber

PL: Bagaimana cara kerja sistem aplikasi pengkajian luka ini?

N: Kita akan membuat aplikasi pengkajian luka yang berjalan di android dan membantu perawat dalam mengkaji luka kronis pasien.

PL: Untuk saat ini sejauh mana aplikasi ini berjalan?

N: Untuk saat ini, sistem ini sudah berjalan hingga mendapatkan data kajian berupa gambar.

PL: Apakah masih ada kelemahan pada aplikasi ini?

N: Karena aplikasi ini masih hanya menyimpan data kajian yang berupa gambar, maka data kajian yang lainnya belum lengkap. Dan oleh sebab itu, lebih baik dilanjutkan dengan menyimpan seluruh data kajian luka dan diletakkan pada medical record pasien agar perawat mudah dalam melihat riwayat kondisi pasien.

PL: Bagaimana prioritas dalam menentukan data kajian luka tersebut?

N: Saya menganjurkan untuk menggunakan Bates-Jensen karena lebih lengkap dalam melakukan kajian luka.

PL: Bagaimana manajemen keperawatan?

N: Pada saat pertama kali pasien datang langsung didata status pasien secara lengkap. Lalu dilanjutkan dengan pengkajian umum luka dan pengkajian lokal luka. Nah setelah itu dicatat perkembangan dan tujuan keperawatannya mau kemana. Apabila ada tindakan yang perlu dilakukan terhadap pasien nanti perawat meminta persetujuan pasien untuk dapat melakukan tindakan pengobatan. Dan diakhiri dengan mencatat rekap kunjungan secara ringkas.

PL: Mengapa pasien perlu mendapat data kajian miliknya pribadi?

N: Karena pasien harus melihat progres penyembuhan setelah mendapat pengobatan agar dapat menilai sendiri dan membantu perawat dalam memutuskan tindakan selanjutnya.

.

LAMPIRAN B

Bates-Jensen Wound Assessment Tools

BATES-JENSEN WOUND ASSESSMENT TOOL Instructions for use

General Guidelines:

Fill out the attached rating sheet to assess a wound's status after reading the definitions and methods of assessment described below. Evaluate once a week and whenever a change occurs in the wound. Rate according to each item by picking the response that best describes the wound and entering that score in the item score column for the appropriate date. When you have rated the wound on all items, determine the total score by adding together the 13-item scores. The HIGHER the total score, the more severe the wound status. Plot total score on the Wound Status Continuum to determine progress.

Specific Instructions:

1. **Size:** Use ruler to measure the longest and widest aspect of the wound surface in centimeters; multiply length x width.
2. **Depth:** Pick the depth, thickness, most appropriate to the wound using these additional descriptions:
1 = tissues damaged but no break in skin surface.
2 = superficial, abrasion, blister or shallow crater. Even with, &/or elevated above skin surface (e.g., hyperplasia).
3 = deep crater with or without undermining of adjacent tissue.
4 = visualization of tissue layers not possible due to necrosis.
5 = supporting structures include tendon, joint capsule.
3. **Edges:** Use this guide:

Indistinct, diffuse	=	unable to clearly distinguish wound outline.
Attached	=	even or flush with wound base, <u>no</u> sides or walls present; flat.
Not attached	=	sides or walls are present; floor or base of wound is deeper than edge.
Rolled under, thickened	=	soft to firm and flexible to touch.
Hyperkeratosis	=	callous-like tissue formation around wound & at edges.
Fibrotic, scarred	=	hard, rigid to touch.
4. **Undermining:** Assess by inserting a cotton tipped applicator under the wound edge; advance it as far as it will go without using undue force; raise the tip of the applicator so it may be seen or felt on the surface of the skin; mark the surface with a pen; measure the distance from the mark on the skin to the edge of the wound. Continue process around the wound. Then use a transparent metric measuring guide with concentric circles divided into 4 (25%) pie-shaped quadrants to help determine percent of wound involved.
5. **Necrotic Tissue Type:** Pick the type of necrotic tissue that is predominant in the wound according to color, consistency and adherence using this guide:

White/gray non-viable tissue	=	may appear prior to wound opening; skin surface is white or gray.
Non-adherent, yellow slough	=	thin, mucinous substance; scattered throughout wound bed; easily separated from wound tissue.
Loosely adherent, yellow slough	=	thick, stringy, clumps of debris; attached to wound tissue.
Adherent, soft, black eschar	=	soggy tissue; strongly attached to tissue in center or base of wound.
Firmly adherent, hard/black eschar	=	firm, crusty tissue; strongly attached to wound base and edges (like a hard scab).

© 2001Barbara Bates-Jensen

6. **Necrotic Tissue Amount:** Use a transparent metric measuring guide with concentric circles divided into 4 (25%) pie-shaped quadrants to help determine percent of wound involved.
7. **Exudate Type:** Some dressings interact with wound drainage to produce a gel or trap liquid. Before assessing exudate type, gently cleanse wound with normal saline or water. Pick the exudate type that is predominant in the wound according to color and consistency, using this guide:

Bloody	=	thin, bright red
Serosanguineous	=	thin, watery pale red to pink
Serous	=	thin, watery, clear
Purulent	=	thin or thick, opaque tan to yellow
Foul purulent	=	thick, opaque yellow to green with offensive odor
8. **Exudate Amount:** Use a transparent metric measuring guide with concentric circles divided into 4 (25%) pie-shaped quadrants to determine percent of dressing involved with exudate. Use this guide:

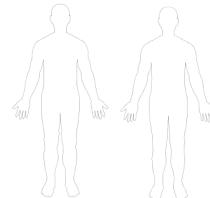
None	=	wound tissues dry.
Scant	=	wound tissues moist; no measurable exudate.
Small	=	wound tissues wet; moisture evenly distributed in wound; drainage involves $\leq 25\%$ dressing.
Moderate	=	wound tissues saturated; drainage may or may not be evenly distributed in wound; drainage involves $> 25\% \text{ to } \leq 75\%$ dressing.
Large	=	wound tissues bathed in fluid; drainage freely expressed; may or may not be evenly distributed in wound; drainage involves $> 75\%$ of dressing.
9. **Skin Color Surrounding Wound:** Assess tissues within 4cm of wound edge. Dark-skinned persons show the colors "bright red" and "dark red" as a deepening of normal ethnic skin color or a purple hue. As healing occurs in dark-skinned persons, the new skin is pink and may never darken.
10. **Peripheral Tissue Edema & Induration:** Assess tissues within 4cm of wound edge. Non-pitting edema appears as skin that is shiny and taut. Identify pitting edema by firmly pressing a finger down into the tissues and waiting for 5 seconds, on release of pressure, tissues fail to resume previous position and an indentation appears. Induration is abnormal firmness of tissues with margins. Assess by gently pinching the tissues. Induration results in an inability to pinch the tissues. Use a transparent metric measuring guide to determine how far edema or induration extends beyond wound.
11. **Granulation Tissue:** Granulation tissue is the growth of small blood vessels and connective tissue to fill in full thickness wounds. Tissue is healthy when bright, beefy red, shiny and granular with a velvety appearance. Poor vascular supply appears as pale pink or blanched to dull, dusky red color.
12. **Epithelialization:** Epithelialization is the process of epidermal resurfacing and appears as pink or red skin. In partial thickness wounds it can occur throughout the wound bed as well as from the wound edges. In full thickness wounds it occurs from the edges only. Use a transparent metric measuring guide with concentric circles divided into 4 (25%) pie-shaped quadrants to help determine percent of wound involved and to measure the distance the epithelial tissue extends into the wound.

BATES-JENSEN WOUND ASSESSMENT TOOL NAME _____

Complete the rating sheet to assess wound status. Evaluate each item by picking the response that best describes the wound and entering the score in the item score column for the appropriate date.

Location: Anatomic site. Circle, identify right (**R**) or left (**L**) and use "X" to mark site on body diagrams:

- | | | | |
|--------------------------|--------------------|--------------------------|---------------|
| <input type="checkbox"/> | Sacrum & coccyx | <input type="checkbox"/> | Lateral ankle |
| <input type="checkbox"/> | Trochanter | <input type="checkbox"/> | Medial ankle |
| <input type="checkbox"/> | Ischial tuberosity | <input type="checkbox"/> | Heel |
| | | | Other Site |



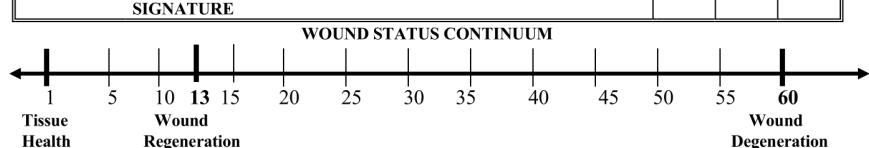
Shape: Overall wound pattern; assess by observing perimeter and depth.

Circle and date appropriate description:

- | | | | |
|--------------------------|------------------|--------------------------|---------------------|
| <input type="checkbox"/> | Irregular | <input type="checkbox"/> | Linear or elongated |
| <input type="checkbox"/> | Round/oval | <input type="checkbox"/> | Bowl/boat |
| <input type="checkbox"/> | Square/rectangle | <input type="checkbox"/> | Butterfly |
| | | | Other Shape |

Item	Assessment	Date Score	Date Score	Date Score
1. Size	1 = Length x width <4 sq cm 2 = Length x width 4-<16 sq cm 3 = Length x width 16.1-<36 sq cm 4 = Length x width 36.1-<80 sq cm 5 = Length x width >80 sq cm			
2. Depth	1 = Non-blanchable erythema on intact skin 2 = Partial thickness skin loss involving epidermis &/or dermis 3 = Full thickness skin loss involving damage or necrosis of subcutaneous tissue; may extend down to but not through underlying fascia; &/or mixed partial & full thickness &/or tissue layers obscured by granulation tissue 4 = Obscured by necrosis 5 = Full thickness skin loss with extensive destruction, tissue necrosis or damage to muscle, bone or supporting structures			
3. Edges	1 = Indistinct, diffuse, none clearly visible 2 = Distinct, outline clearly visible, attached, even with wound base 3 = Well-defined, not attached to wound base 4 = Well-defined, not attached to base, rolled under, thickened 5 = Well-defined, fibrotic, scarred or hyperkeratotic			
4. Under-mining	1 = None present 2 = Undermining < 2 cm in any area 3 = Undermining 2-4 cm involving < 50% wound margins 4 = Undermining 2-4 cm involving > 50% wound margins 5 = Undermining > 4 cm or Tunneling in any area			
5. Necrotic Tissue Type	1 = None visible 2 = White/grey non-viable tissue &/or non-adherent yellow slough 3 = Loosely adherent yellow slough 4 = Adherent, soft, black eschar 5 = Firmly adherent, hard, black eschar			
6. Necrotic Tissue Amount	1 = None visible 2 = < 25% of wound bed covered 3 = 25% to 50% of wound covered 4 = > 50% and < 75% of wound covered 5 = 75% to 100% of wound covered			
7. Exudate Type	1 = None			

Item	Assessment	Date Score	Date Score	Date Score
	2 = Bloody 3 = Serosanguineous: thin, watery, pale red/pink 4 = Serous: thin, watery, clear 5 = Purulent: thin or thick, opaque, tan/yellow, with or without odor			
8. Exudate Amount	1 = None, dry wound 2 = Scant, wound moist but no observable exudate 3 = Small 4 = Moderate 5 = Large			
9. Skin Color Surrounding Wound	1 = Pink or normal for ethnic group 2 = Bright red &/or blanches to touch 3 = White or grey pallor or hypopigmented 4 = Dark red or purple &/or non-blanchable 5 = Black or hyperpigmented			
10. Peripheral Tissue Edema	1 = No swelling or edema 2 = Non-pitting edema extends <4 cm around wound 3 = Non-pitting edema extends ≥4 cm around wound 4 = Pitting edema extends < 4 cm around wound 5 = Crepitus and/or pitting edema extends ≥4 cm around wound			
11. Peripheral Tissue Induration	1 = None present 2 = Induration, < 2 cm around wound 3 = Induration 2-4 cm extending < 50% around wound 4 = Induration 2-4 cm extending ≥ 50% around wound 5 = Induration > 4 cm in any area around wound			
12. Granulation Tissue	1 = Skin intact or partial thickness wound 2 = Bright, beefy red; 75% to 100% of wound filled &/or tissue overgrowth 3 = Bright, beefy red; < 75% & > 25% of wound filled 4 = Pink, &/or dull, dusky red &/or fills ≤ 25% of wound 5 = No granulation tissue present			
13. Epithelialization	1 = 100% wound covered, surface intact 2 = 75% to <100% wound covered &/or epithelial tissue extends >0.5cm into wound bed 3 = 50% to <75% wound covered &/or epithelial tissue extends to <0.5cm into wound bed 4 = 25% to < 50% wound covered 5 = < 25% wound covered			
TOTAL SCORE				
SIGNATURE				



Plot the total score on the Wound Status Continuum by putting an "X" on the line and the date beneath the line. Plot multiple scores with their dates to see-at-a-glance regeneration or degeneration of the wound.

LAMPIRAN C

Format Pengkajian Luka Klinik Moist Care

MOIST CARE PUSAT PENYEMBUHAN LUCA PERAWATAN KULIT, STOMA DAN KONTINENSIA

www.klinikmoist.com
www.stopamputasi.com

REKAP KUNJUNGAN KLIEN MOIST

NAMA PASIEN :
NO. REG :
DX KEPERAWATAN :



Rekam ASKEP MOIST Care
Raskep.MC-008/17

MOIST CARE PUSAT PENYEMBUHAN LUCA PERAWATAN KULIT, STOMA DAN KONTINENSIA

www.klinikmoist.com
www.stopamputasi.com

INFORM CONCENT

(SURAT PERNYATAAN PERSETUJUAN TINDAKAN)



Saya yang bertanda tangan dibawah ini :

Nama Lengkap :
No. KTP :
Tgl Lahir/Usia :
Alamat Tinggal :
Telp/HP :

Bertindak sebagai wakil/ wali keluarga pasien :

Nama Pasien	:
Usia	:
Diagnosa	:

Menyatakan SETUJU untuk dilakukan tindakan atas pasien
Tersebut diatas. Dengan telah diinformasikan terlebih dahulu tentang kondisi pasien tersebut dan
saya mengerti serta memahami konsekuensi dari tindakan tersebut, sesuai dengan penjelasan
yang telah diberikan tenaga kesehatan yang melakukan perawatan.

Demikian surat ini saya buat dengan penuh kesadaran dan tanpa tekanan dari pihak manapun.

Jakarta ,..... 20

(.....) (.....)

Saksi-Saksi

(.....) (.....)

Rekam ASKEP MOIST Care
Raskep.MC-007/17

MOIST CARE PUSAT PENYEMBUHAN LUKA
PERAWATAN KULIT, STOMA DAN KONTINENSIA

www.klinikmoist.com
www.stopamputasi.com

STATUS KLIEN



LOKASI :.....
Klinik / Homecare)*

NO. REG	:	
Nama Lengkap	:	
Agama/ Keyakinan	:	
TTL / Usia	:	
Jenis kelamin	:	
Alamat / No. Telp	:	
HP/ Email	:	
Diagnosa Keperawatan	:	
Therapist)* utama	:	
Tim Therapist)*	1	
	2	
	3	
	4	
Alergi	:	

* Therapist : Certified Nurse

Rekam ASKEP MOIST Care
Raskep.MC-001/17

MOIST CARE PUSAT PENYEMBUHAN LUKA
PERAWATAN KULIT, STOMA DAN KONTINENSIA

www.klinikmoist.com
www.stopamputasi.com



PENGKAJIAN UMUM

Riwayat kejadian luka:

--

Riwayat perawatan sebelumnya:

--

Faktor penyulit proses penyembuhan luka:

Status nutrisi	Mobilisasi	
Penyakit penyerta	Obat-obatan yang digunakan	
Vaskularisasi	Lokasi	
Status psikologis	Merokok	
Lainya		

Rekam ASKEP MOIST Care
Raskep.MC-002/17

MOIST CARE PUSAT PENYEMBUHAN LUKA
PERAWATAN KULIT, STOMA DAN KONTINENSIA

www.klinikmoist.com
www.stopamputasi.com

PENGKAJIAN LOKAL LUKA



Tipe Luka	:
Tipe Penyembuhan	:
Gambar Luka,	
(tgl.....) kunjungan ke :	(tgl.....) kunjungan ke :
Pemeriksaan tanda-tanda vital (TTV) dan penunjang	
TD :	TD.....
N :	N.....
P :	P.....
S :	S.....
GDS :	GDS.....
ABPI :	ABPI :
Pemeriksaan fisik dan penunjang lainnya (yang direncanakan maupun dilakukan, misal : kultur, dll)	

Rekam ASKEP MOIST Care
Raskep.MC-003/17

PENGKAJIAN LOKAL LUKA



(tgl.....)			(tgl.....)		
LUKA 1	LUKA 2	LUKA 3	LUKA 1	LUKA 2	LUKA 3
Stadium Luka (1-Unstage)					
Ukuran Luka (P x L x D/T)					
Goa / Undermining (goa di jam a-b, X cm, di jam Y)					
Exudate/ cairan luka (Tipe, jumlah, malodour)					
Warna dasar luka (x% merah, y% kuning, z% hitam)					
Dasar luka (menyatu dengan dasar luka)					
TEPI LUKA (tebal, tipis, halus, kaku, edema, epibole, dll)					
Kulit sekitar luka (iritasi, maserasi, kemerahan, edema, dll)					
Tanda infeksi (ada / tidak, local / sistemik)					
Nyeri (1 - 10)					

Rekam ASKEP MOIST Care
Raskep.MC-004/17

MOIST CARE PUSAT PENYEMBUHAN LUCA PERAWATAN KULIT, STOMA DAN KONTINENSIA

www.klinikmoist.com
www.stopamputasi.com

CATATAN PERKEMBANGAN



Rekam ASKEP MOIST Care
Raskep.MC-005/17

LAMPIRAN D

Source Code

Link *github* *full* *source* *code* :
https://github.com/ardani77/chronic-wound-backend

RIWAYAT HIDUP



MUHAMMAD ARDANI, Lahir di Palembang, 24 Oktober 1999. Anak tunggal dari pasangan Bapak Ruslan Abdulgani dan ibu Nurhayati. Saat ini penulis tinggal di Jl. Bunga Belakang No. 12 RT 010/009 Kelurahan Palmeriam, Kecamatan Matraman, Kota Jakarta Timur, Provinsi DKI Jakarta.

Riwayat Hidup: Penulis mengawali pendidikan di TK Aisyiyah 27 pada tahun 2003-2005. Kemudian melanjutkan pendidikan di SDN Palmeriam 03 Pagi Jakarta pada tahun 2005-2011. Selanjutnya penulis melanjutkan ke SMPN 7 Jakarta pada tahun 2011-2014. Kemudian melanjutkan pendidikan di SMAN 31 Jakarta pada tahun 2014-2017. Pada tahun 2018 penulis melanjutkan ke Universitas Negeri Jakarta (UNJ) di program studi Ilmu Komputer melalui jalur SBMPTN.

Riwayat Organisasi: Selama di bangku perkuliahan, penulis merupakan anggota Badan Eksekutif Mahasiswa Program Studi (BEMP) Ilmu Komputer. Penulis juga berpartisipasi dalam kegiatan COMPARE (Computer Academic Care) yaitu kegiatan workshop dan seminar online serta offline yang diadakan oleh BEM sebagai ketua pelaksana. Penulis juga berpartisipasi dalam kegiatan PKKMB Program Studi Ilmu Komputer sebagai koordinator seksi acara.

Riwayat Prestasi: Pada saat duduk di bangku SMA, penulis sering mengikuti kegiatan perlombaan bola voli diikuti dengan Juara I Putra Kompetisi Bola Voli SMA-SMK se-DKI Jakarta di SMAN 80. Selain itu, penulis juga pernah menjadi Juara I PKM-PI pada lomba Program Kreativitas Mahasiswa (PKM) yang diselenggarakan oleh Asosiasi MIPA LPTK Indonesia (AMLI) dan FMIPA Universitas Negeri Medan pada bulan April s/d November 2023.