

**RANCANG BANGUN SISTEM INFORMASI KEPERAWATAN
LUKA**

Skripsi

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



**Oleh:
Muhammad Insan Khamil
3145161580**

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

2023

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI

LEMBAR PERNYATAAN

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul "**Rancang Bangun Sistem Informasi 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 yang 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, 17 Agustus 2023

Muhammad Insan Khamil

HALAMAN PERSEMBAHAN

Untuk Bapak, Ibu, dan Adik-adikku

ABSTRAK

Muhammad Insan Khamil. Rancang Bangun Sistem Informasi Keperawatan Luka. Skripsi. Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta. 2023. Di bawah bimbingan Muhammad Eka Suryana, M. Kom dan Drs. Mulyono, M. Kom

Saat ini pencatatan data pengkajian luka dan pemeriksaan kesehatan masih dilakukan secara tradisional. Hal ini menyebabkan pemanfaatan informasi menjadi kurang maksimal, berjalan kurang efektif, lama dalam prosesnya dan menyulitkan proses pertukaran data medis antar divisi medis. Sehingga, diperlukannya aplikasi yang dapat mengarsipkan data pengkajian luka dan pemeriksaan kesehatan secara digital dan memudahkan dalam pertukaran data antar divisi medis. Adapun Penelitian ini bertujuan untuk merancang sistem informasi keperawatan luka di klinik *Moist Care*. Jenis Penelitian ini adalah Pengembangan/*Research and Development*. Data diambil melalui paparan presentasi bersama ibu Irma Puspita Arisanti selaku pemilik klinik *Moist Care*. Hasil penelitian yang didapatkan yaitu: (1) Terciptanya sistem informasi keperawatan luka yang sudah mengimplementasikan sebagian fitur-fitur pada *product backlog*. Diantaranya pembuatan akun pasien, *dashboard* klinik, pemeriksaan kesehatan dan sebagai proses pengobatan luka (*view* dan *web service* inventaris dan layanan). Adapun perancangannya dilakukan dengan metode *scrum* dengan tahapan penyusunan *product backlog*, *sprint backlog* dan terbagi menjadi empat *sprint*; (2) Terimplementasikannya *Web Service* yang berfungsi sebagai *Back-End* sistem informasi keperawatan luka; (3) Sistem informasi keperawatan luka dikembangkan menggunakan bahasa *python* dengan bantuan *framework flask*.

Kata kunci : Sistem informasi, *web service*, arsip data, pertukaran data.

ABSTRACT

Muhammad Insan Khamil. Wound Nursing Information System Design. Thesis. Faculty of Mathematics and Natural Sciences, State University of Jakarta. 2023. Under the guidance of Muhammad Eka Suryana, M. Kom and Drs. Mulyono, M. Kom

Currently, the recording of wound assessment data and medical examinations is still carried out in the traditional way. This causes the utilization of information to be less than optimal, runs less effectively, takes longer to process and complicates the process of exchanging medical data between medical divisions. Thus, an application is needed that can digitally archive wound assessment and medical examination data and facilitate the exchange of data between medical divisions. This study aims to design a wound nursing clinical information system at the Moist Care clinic. This type of research is Research and Development. The data was taken through a presentation with Mrs. Irma Puspita Arisanti as the owner of the Moist Care clinic. The research results obtained are: (1) The creation of a wound nursing information system that has implemented some of the features in the product backlog. Including creating patient accounts, clinical dashboards, health checks and part of the wound treatment process (view and web service of inventory and services). The design is done using the scrum method with the stages of preparing a product backlog, sprint backlog and divided into four sprints; (2) Implementation of a Web Service that functions as the Back-End of wound nursing information systems; (3) Wound nursing information systems were developed in python with the help of the flask framework.

Keywords: *Information system, web service, data archive, data exchange.*

KATA PENGANTAR

Atas berkat rahmat Allah Yang Maha Kuasa, penulis bisa menyelesaikan skripsi yang berjudul "**Rancang Bangun Sistem Informasi Keperawatan Luka**". Selain itu penulis juga berterima kasih kepada pihak-pihak pendukung dan mendoakan semoga kebaikan dan jasa-jasa yang sudah diberikan dibalas Allah Yang Maha Kuasa. Adapun pihak-pihak tersebut sebagai berikut:

1. Yth. Kedua orang tua dan adik-adik penulis yang selama ini telah senantiasa sabar membimbing, memberikan semangat, mengingatkan, dan mendo'akan penulis.
2. Yth. Ibu Dr. Ria Arafiah, M. Si. selaku Koordinator Program Studi S-1 Ilmu Komputer Universitas Negeri Jakarta yang sudah membimbing penulis dalam hal akademik,
3. Yth. Bapak Muhammad Eka Suryana, M. Kom. selaku dosen pembimbing I yang banyak memberikan bantuan, masukan, dan saran baik secara konten maupun penulisan,
4. Yth. Bapak Drs. Mulyono, M. Kom. selaku dosen pembimbing II sekaligus dosen pembimbing akademik yang banyak memberikan bantuan, masukan, dan saran baik secara konten maupun penulisan,
5. Yth. Seluruh dosen program studi S-1 Ilmu Komputer yang sudah memberikan banyak ilmu kepada penulis selama perkuliahan,
6. Yth. Agus Setiawan, Fandy Alifian, Dio Aryanto Dio Wicaksono dan teman-teman Ilmu Komputer angkatan 2016 yang senantiasa menemani, memberikan semangat, dan motivasi dari semenjak awal dunia perkuliahan,

7. Yth. Fauzan Rizky Ramadhan, Abdul Ghaffar Sidik, Sony Ariyanto, dan Imam Nuddudin yang senantiasa menemani saya saat suka dan duka dalam menyelesaikan skripsi,

Merupakan kekurangan dari pribadi penulis jika masih ditemukan banyak kesalahan di dalam ini. Penulis sangat berterima kasih jika terdapat saran-saran membangun terkait skripsi penulis.

Jakarta, 17 Agustus 2023

Muhammad Insan Khamil

DAFTAR ISI

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI	i
LEMBAR PERNYATAAN	ii
HALAMAN PERSEMBAHAN	iii
ABSTRAK	iv
ABSTRACT	v
KATA PENGANTAR	vi
DAFTAR ISI	xii
DAFTAR TABEL	xv
DAFTAR GAMBAR	xv
I PENDAHULUAN	1
1.1 Latar Belakang Masalah	1
1.2 Rumusan Masalah	5
1.3 Pembatasan Masalah	5
1.4 Tujuan Penelitian	6
1.5 Manfaat Penelitian	6
II KAJIAN PUSTAKA	7
2.1 Sistem Informasi	7
2.2 Unified Modeling Language (UML)	8
2.2.1 <i>Use Case Diagram</i>	8

2.2.2	<i>Class Diagram</i>	8
2.2.3	<i>Activity Diagram</i>	9
2.2.4	<i>Sequence Diagram</i>	9
2.3	Pengantar <i>Flask</i>	9
2.4	<i>MongoDB</i>	12
2.5	<i>Scrum</i>	16
III METODOLOGI PENELITIAN		22
3.1	Pengumpulan Data	22
3.2	Analisa Kebutuhan	23
3.3	Perancangan Sistem Menggunakan <i>Scrum</i>	23
3.3.1	<i>Product Backlog</i>	24
3.3.2	<i>Sprint Backlog</i>	25
3.3.3	<i>Sprint</i>	25
3.3.4	<i>Deploy</i>	25
IV HASIL DAN PEMBAHASAN		26
4.1	Pembahasan	26
4.1.1	<i>Sprint-1</i>	26
4.1.2	<i>Sprint-2</i>	54
4.1.3	<i>Sprint-3</i>	61
4.1.4	<i>Sprint-4</i>	65
4.1.5	Laporan akhir pengembangan sistem	65
V KESIMPULAN DAN SARAN		68
5.1	Kesimpulan	68
5.2	Saran	69

DAFTAR PUSTAKA	71
DAFTAR LAMPIRAN	71
A Dokumentasi <i>Screenshot</i> Presentasi Bersama <i>Owner Klinik Moist Care</i>	72
B Transkrip Paparan Presentasi Bersama Pemilik Klinik <i>Moist Care</i> 1	73
C Transkrip Paparan Presentasi Bersama Pemilik Klinik <i>Moist Care</i> 2	75
D Surat Pernyataan Kesediaan Kerjasama Dari Mitra	76
E <i>Code</i> Untuk <i>View Login</i> Admin dan Perawat	77
F <i>Code</i> Untuk <i>Web Service Login</i> Admin dan Perawat	80
G <i>Code</i> Untuk <i>Dashboard</i> Klinik	81
H <i>Code</i> Untuk <i>Web Service Login</i> Pasien	84
I <i>Code</i> Untuk <i>View Registrasi</i> Pasien Dari Klinik	85
J <i>Code</i> Untuk <i>Web Service Registrasi</i> Pasien Dari Klinik	91
K <i>Code</i> Untuk <i>View List</i> Permohonan Akun Pasien	93
L <i>Code</i> Untuk <i>Web Service List</i> Permohonan Akun Pasien	96
M <i>Code</i> Untuk <i>Web Wervice List</i> Verifikasi/Terima pasien	97
N <i>Code</i> Untuk <i>Web Service List</i> Tolak Verifikasi Pasien	98
O <i>Code</i> Untuk <i>View List</i> Pasien Terdaftar Klinik	99
P <i>Code</i> Untuk <i>Web Service List</i> Pasien Terdaftar Klinik	101

Q <i>Code Untuk View Detail Hasil Pemeriksaan Kesehatan</i>	102
R <i>Code Untuk Web Service Tambah Hasil Pemeriksaan Kesehatan</i>	105
S <i>Code Untuk Web Service Untuk Mendapatkan Detail Data Hasil Pemeriksaan Kesehatan Pasien Berdasar Tanggal Pemeriksaan</i>	106
T <i>Code Untuk View List Inventaris</i>	107
U <i>Code Untuk View Tambah Inventaris</i>	109
V <i>Code Untuk Web Service Tambah Inventaris</i>	112
W <i>Code Untuk View List Layanan</i>	113
X <i>Code Untuk View Tambah Layanan</i>	116
Y <i>Code Untuk Web Service Tambah Layanan</i>	119
DAFTAR RIWAYAT HIDUP	120

DAFTAR TABEL

Tabel 2.1	Fungsi-fungsi Metode HTTP	10
Tabel 3.1	<i>Product Backlog</i>	24
Tabel 4.1	<i>Sprint-1 backlog</i>	26
Tabel 4.2	<i>Sprint-1 backlog</i> lanjutan 1	27
Tabel 4.3	<i>Sprint-1 backlog</i> lanjutan 2	28
Tabel 4.4	<i>Routing table</i> pembuatan akun pasien	31
Tabel 4.5	<i>Routing table dashboard</i> klinik	33
Tabel 4.6	<i>Routing table</i> pemeriksaan kesehatan	33
Tabel 4.7	<i>Routing table</i> pemeriksaan kesehatan - lanjutan	34
Tabel 4.8	<i>Routing table</i> proses pengobatan	40
Tabel 4.9	<i>Routing table</i> proses pengobatan - lanjutan 1	41
Tabel 4.10	<i>Routing table</i> proses pengobatan - lanjutan 2	42
Tabel 4.11	<i>Routing table</i> pendaftaran pasien berobat	44
Tabel 4.12	<i>Routing table</i> pengelolaan antrian - lanjutan	48
Tabel 4.13	<i>Routing table</i> administrasi keuangan	51
Tabel 4.14	<i>Sprint-2 backlog</i>	54
Tabel 4.15	<i>Sprint-3 backlog</i>	61

DAFTAR GAMBAR

Gambar 2.1	Contoh registrasi Blueprint pada dokumen “auth.py”.	
Sumber:		
https://flask.palletsprojects.com/en/2.1.x/tutorial/views/ . . . 10		
Gambar 2.2	Dokumen <code>__init__.py</code> .	Sumber:
https://flask.palletsprojects.com/en/2.1.x/tutorial/views/ . . . 11		
Gambar 2.3	Contoh pembuatan <i>routing</i> dengan <i>Blueprint</i> .	Sumber:
https://flask.palletsprojects.com/en/2.1.x/tutorial/views/ . . . 11		
Gambar 2.4	Pemodelan RDBMS.	Sumber:
https://www.mongodb.com/nosql-explained/ 12		
Gambar 2.5	Pemodelan NoSQL.	Sumber:
https://www.mongodb.com/nosql-explained/ 13		
Gambar 2.6	Menambahkan Ekstensi PyMongo pada Flask.	Sumber:
Dokumentasi PyMongo, https://flask-pymongo.readthedocs.io/en/latest/ 14		
Gambar 3.1	Tahapan penelitian	22
Gambar 4.1	<i>Mock up</i> tampilan awal pembuatan akun pasien	28
Gambar 4.2	<i>Mock up</i> tampilan <i>list</i> permohonan pembuatan akun pasien . .	29
Gambar 4.3	<i>Mock up</i> pembuatan akun pasien	30
Gambar 4.4	Tabel Pasien	32
Gambar 4.5	<i>Mock up</i> tampilan awal dashboard klinik	32
Gambar 4.6	Tabel pemeriksaan kesehatan	34
Gambar 4.7	<i>Mock up</i> tampilan awal data medis pasien	35
Gambar 4.8	<i>Mock up</i> profil data medis pasien	36
Gambar 4.9	<i>Mock up</i> <i>list</i> histori kajian data medis pasien	36

Gambar 4.10 <i>Mock up</i> detail histori kajian data medis pasien	37
Gambar 4.11 <i>Mock up</i> galeri luka pasien	38
Gambar 4.12 <i>Mock up</i> tampilan awal inventaris	38
Gambar 4.13 <i>Mock up</i> tampilan awal layanan	39
Gambar 4.14 Tabel data kajian	42
Gambar 4.15 Tabel inventaris	43
Gambar 4.16 Tabel layanan	43
Gambar 4.17 <i>Mock up</i> melihat daftar dan urutan pasien yang akan dilayani	44
Gambar 4.18 Tabel pendaftaran berobat	45
Gambar 4.19 <i>Mock up</i> menentukan kuota pelayanan pasien yang mendaftar secara <i>offline</i> maupun <i>online</i> pada hari yang ditentukan	45
Gambar 4.20 <i>Mock up</i> tampilan awal melakukan pendaftaran pasien yang akan berobat secara <i>offline</i> maupun <i>online</i>	46
Gambar 4.21 <i>Mock up</i> mendaftarkan pasien yang berobat secara <i>offline</i> . .	47
Gambar 4.22 <i>database</i> pengelolaan antrian	49
Gambar 4.23 <i>Mock up</i> verifikasi dan validasi tagihan	49
Gambar 4.24 <i>Mock up</i> tambah barang/layanan pada verifikasi dan validasi tagihan	50
Gambar 4.25 Tabel tagihan dan tabel detail tagihan	51
Gambar 4.26 <i>Use case</i> diagram	52
Gambar 4.27 Desain <i>database</i> sistem	53
Gambar 4.28 Realisasi <i>view login</i> admin/perawat	55
Gambar 4.29 Realisasi <i>view dashboard</i> klinik	56
Gambar 4.30 Pemanggilan <i>web service</i> login pasien	57
Gambar 4.31 Realisasi <i>view registrasi</i> pasien dari klinik	57

Gambar 4.32 Realisasi <i>view list</i> permohonan akun pasien	58
Gambar 4.33 Realisasi <i>view list</i> pasien terdaftar klinik	59
Gambar 4.34 Realisasi <i>view</i> detail hasil pemeriksaan kesehatan seorang pasien berdasarkan tanggal pemeriksaan	62
Gambar 4.35 Realisasi <i>view</i> semua inventaris klinik	63
Gambar 4.36 Realisasi <i>view</i> tambah inventaris klinik	63
Gambar 4.37 Realisasi <i>view</i> semua layanan klinik	64
Gambar 4.38 Realisasi <i>view</i> tambah layanan klinik	65
Gambar 1.1 Dokumentasi presentasi bersama Ibu Irma Puspita Arisanti 1 .	72
Gambar 1.2 Dokumentasi presentasi bersama Ibu Irma Puspita Arisanti 2 .	72
Gambar 4.1 Surat Pernyataan Kesediaan Kerjasama Dari Mitra	76

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Luka kronis adalah masalah kritis dalam kesehatan. Di Amerika Serikat, sekitar 6,5 juta orang menderita luka kronis dan biaya perawatan luka kronis menghabiskan sekitar \$20 miliar per tahun. Bahkan di negara maju, sekitar 1-2% dari seluruh populasi terkena luka kronis selama hidup mereka. (Biswas et al., 2018). Luka kronis berdampak terhadap finansial dan penurunan kualitas hidup pasien. Kerusakan fisik, sosial, dan emosional seperti penurunan mobilitas, rasa sakit, ketidaknyamanan, membatasi kinerja aktivitas sehari-hari. Isolasi sosial, frustasi, dan reaksi psikologis lainnya yang menimbulkan dampak pada kehidupan pasien. (Vogt et al., 2020). Di Indonesia sendiri pengidap luka kronis berjumlah sekitar 24% dari 8,6% total populasi terhadap kasus diabetes. (Safitri et al., 2022).

Pada penelitian sebelumnya yang dilakukan oleh Salsa yang berjudul "Rancang Bangun Aplikasi dan *Web Service* Pengkajian Luka Kronis Khusus Modul Pengolahan Citra Berbasis Android". Salsa melakukan wawancara dengan Ratna Aryani, M.Kep., Dosen Politeknik Negeri Jakarta I, diperoleh bahwa saat melakukan penggantian balutan luka dan pengecekan awal kondisi luka dilakukan pengkajian luka. Berikut langkah-langkah pengkajian luka diawali dengan balutan luka dibuka, lalu luka dicuci, dan diakhiri dengan proses pengkajian luka. Instrumen yang dipilih saat melakukan pengkajian luka ialah Bates-Jensen *Wound Assesment Tools* (BWAT). Pada BWAT ada 13 kategori penilaian yakni beberapa di antaranya tepi luka, ukuran luka, epitalisasi dan jumlah eksudat (cairan tubuh yang keluar dari jaringan selama peradangan). Saat ini data pengkajian luka masih dilakukan secara

tradisional dicatat dalam arsip atau catatan kertas, maka dari itu salsa mengusulkan untuk mendigitalisasi pencatatan data luka yang sudah dikaji. (Rahmadati, 2023).

Penelitian lain yang terkait juga dilakukan oleh Ardiansyah, menjelaskan bahwa Rumah Sakit Umum Kambang Jambi masih memakai cara tradisional dalam pelayanannya seperti mendapatkan nomor antrian berobat, informasi mengenai jadwal dokter dan jumlah seluruh pasien. Hal ini menyebabkan pemanfaatan informasi menjadi kurang maksimal, berjalan kurang efektif dan lama dalam prosesnya. Dengan adanya permasalahan yang terjadi ardiansyah dan kawannya menyimpulkan bahwa dibutuhkannya Sistem Informasi Manajemen Rumah Sakit Berbasis *Website* untuk menyelesaikan permasalahan yang ada. Sehingga meminimalkan kekurangan dan ketidak efektifan dalam pelayanan. (Ardiansyah and Effiyaldi, 2021).

Dalam jurnal berjudul "Sistem Informasi Rekam Medis Pada Rumah Sakit Umum Daerah (RSUD) Pacitan Berbasis *Web Base*" oleh Gunawan Susanto. Pencatatan riwayat dan data rekam medis kesehatan milik pasien merupakan hal yang krusial dalam dunia medis karena data tersebut digunakan untuk pemeriksaan pasien selanjutnya. Sistem pencatatan yang dipakai memiliki kelemahan. Hal ini dikarenakan data rekam medis pasien hanya disimpan secara lokal di tempat pasien diperiksa dan dirawat serta pertukaran data langsung antara divisi medis tidak diperbolehkan. Maka dari itu dilakukan pengembangan sistem informasi rekam medis yang memiliki tujuan untuk menyelesaikan kelemahan yang dimiliki oleh sistem pencatatan rekam medis pasien yang sebelumnya, yaitu alternatif teknologi yang dapat diterapkan di masa yang akan datang untuk pencatatan dan penyampaian data rekam medis. (Susanto and Sukadi, 2021).

Pada penelitian yang dilaksanakan oleh Inah Carminah yang berjudul "Aplikasi Monitoring Perawatan Luka Diabetes Melitus Berbasis *Website*". Proses

elayanan yang masih menggunakan *paper base system* memiliki risiko kerusakan atau kehilangan data rekam medis pasien. Selain itu membuat perawat kewalahan ketika mencari data rekam medis pasien secara satu-persatu ketika dibutuhkan ketika pasien datang untuk berobat kembali. Berangkat dari permasalahan di atas memotivasi instansi untuk membuat aplikasi dengan tujuan untuk meningkatkan kualitas pelayanan pasien saat berobat. (Carminah et al., 2021).

Didalam buku berjudul “Rancang Bangun Aplikasi *Mobile Android* Sebagai Alat Deteksi Warna Dasar Luka Dalam Membantu Proses Pengkajian Luka Kronis Dengan Nekrosis”, Tehnik pengkajian luka berdasarkan warna luka yang umum digunakan salah satunya The RYB (*Red-Yellow-Black*) *wound classification system*. Metode ini digunakan dengan mengandalkan subyektifitas dari perawat luka. Hasil penelitian pada buku ini menunjukkan bahwa perawat mampu mengetahui perbedaan warna luka secara otomatis yang membantu proses pengkajian luka kronis dengan nekrosis. (Aryani et al., 2018). Ia juga meneliti dan menemukan bahwa perban basah membantu mempercepat proses penyembuhan luka. Perawat harus mempertimbangkan untuk menggunakan balutan basah daripada perawatan standar untuk meningkatkan penyembuhan. Namun, perawat harus melindungi luka dari kelembapan yang berlebihan karena dapat merusak kulit di sekitar luka atau di dalam luka. (Aryani, 2016).

Pada payung penelitian *medical imaging* yang sama dengan peneliti juga sudah pernah dilakukan penelitian mengenai Pengaruh Penggunaan *Color Model LAB* dalam Kalibrasi Warna Luka Menggunakan Metode Segmentasi *K-Means* dan *Mean Shift* oleh rekan sesama peneliti. (Khairunnisa, 2021). Dan Muhamad rizki juga melakukan penelitian deteksi tepi luka menggunakan metode *Active Contour* yang ditambah interpolasi. (Rizki, 2022). Kedua penelitian tersebut merupakan penelitian berdasarkan dua kategori pengkajian luka yaitu warna luka dan tepi luka,

algoritma yang dikembangkan pada penelitian tersebut direncanakan akan terintegrasi dalam satu ekosistem aplikasi, yakni sistem informasi keperawatan luka. Dimana pada penelitian Salsa Rahmadati melakukan perancangan aplikasi pengkajian luka kronis berbasis Android sesuai modul *image processing*. (Rahmadati, 2023).

Berdasarkan hal di atas, penulis tertarik untuk melakukan penelitian yang bertujuan untuk membuat sistem informasi keperawatan luka dengan dasar pengembangan menggunakan data paparan presentasi bersama ibu Irma Puspita Arisanti selaku pemilik klinik *moist care* dan sesuai dengan proposal PKM-PI dengan judul "Pengembangan Pelayanan Sistem Informasi Klinik Serta Fitur Keperawatan Luka Pada Aplikasi Untuk Mendukung Integrasi Data Kesehatan Dan Ketahanan Nasional Bidang Kesehatan" yang dibuat oleh Hafiz dan tim. Melanjutkan penelitian sebelumnya yang dilakukan oleh Salsa, dimana pengkajian luka masih dilakukan dengan cara manual atau arsip kertas sehingga Salsa membuat aplikasi untuk mengarsipkan data secara digital dan peneliti mengembangkan *web* aplikasi yang berkaitan dengan aplikasi sebelumnya untuk dapat diakses datanya oleh klinik dengan maksud seluruh staff klinik yang berkepentingan dapat dengan mudah mengaksesnya.

Sistem Informasi tersebut diharapakan dapat menambah opsi pendaftaran berobat secara *online* selain daripada pendaftaran secara *offline*, membantu pengelolaan antrian, membantu integrasi data pasien dan perawat secara digital, manajemen inventaris, beserta verifikasi dan validasi biaya tagihan sehingga dapat mempermudah pelayanan.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang yang diutarakan di atas, maka perumusan masalah pada penelitian ini adalah bagaimana membuat rancang bangun sistem informasi keperawatan luka?

1.3 Pembatasan Masalah

Batasan masalah dalam penelitian ini adalah:

1. Sistem informasi keperawatan luka dibuat dengan dasar instrumen pengkajian *Bates-Jensen Wound Assessment Tool (BWAT)*.
2. *User* aplikasi sistem informasi keperawatan luka adalah perawat dan admin klinik.
3. Sistem informasi keperawatan luka dibuat berbasis *Website*
4. Sistem informasi keperawatan luka dibuat berdasarkan paparan presentasi bersama pemilik klinik *Moist Care* yaitu ibu Irma Puspita Arisanti.
5. Model pengembangan yang digunakan untuk mengembangkan sistem informasi keperawatan luka adalah *scrum*.
6. Fitur-fitur yang diimplementasi pada sistem informasi keperawatan luka, diantaranya adalah pembuatan akun pasien, *dashboard* klinik, pemeriksaan kesehatan dan sebagian proses pengobatan luka (*view* dan *web service* inventaris dan layanan).

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk membuat rancang bangun sistem informasi keperawatan luka di klinik *Moist Care*.

1.5 Manfaat Penelitian

1. Bagi penulis

Penelitian yang dilakukan merupakan media penerapan dari berbagai ilmu pengetahuan, khususnya dalam perancangan sistem informasi keperawatan luka pada klinik *Moist Care*.

2. Bagi Program Studi Ilmu Komputer

Penelitian ini dapat menjadi pintu gerbang untuk penelitian selanjutnya di masa depan.

3. Bagi Universitas Negeri Jakarta

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

BAB II

KAJIAN PUSTAKA

2.1 Sistem Informasi

Sistem informasi tersusun dari dua kata yaitu sistem dan informasi. Menurut Jerry FitzGerald, Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu. Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, bekerja sama membentuk satu kesatuan. Komponen-komponen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem. Setiap sistem tidak perduli betapapun kecilnya, selalu mengandung komponen-komponen atau subsistem-subsistem. Setiap subsistem mempunyai sifat-sifat dari sistem untuk menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. (FitzGerald et al., 1981)

Informasi adalah data yang telah diproses menjadi bentuk yang memiliki arti bagi penerima dan dapat berupa fakta, suatu nilai yang bermanfaat. Jadi ada suatu proses transformasi data menjadi suatu informasi yaitu *input*, proses dan *output*. Menurut Robert A. Leitch, Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan. (A. Leitch and Davis, 2001)

2.2 Unified Modeling Language (UML)

Subbab ini ditulis berdasarkan (Suendri, 2019). *Unified Modeling Language* (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar dalam memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis OO (*Object-Oriented*). Siti Fatima mengatakan UML memberikan standar penulisan dalam sebuah sistem *blueprint* yang meliputi konsep bisnis proses, penulisan kelas-kelas yang spesifik dalam bahasa program, skema database, dan komponen-komponen yang dibutuhkan dalam sistem *software* Fatima (2013). Diagram *Unified Modelling Language* (UML) antara lain sebagai berikut :

2.2.1 Use Case Diagram

Use case menggambarkan *external view* dari sistem yang akan kita buat modelnya. Model *use case* dapat diartikan sebagai diagram *use case*, tetapi diagram ini tidak sama dengan model sebab model memiliki cakupan yang lebih luas dari diagram. *Use case* harus sanggup dalam menggambarkan susunan atau urutan aktor yang menghasilkan nilai terukur. (Widodo and Prabowo, 2011)

2.2.2 Class Diagram

Kelas merupakan suatu set objek dengan atribut dan perilaku yang sama. Kelas dapat disebut juga sebagai kelas objek. (Whitten et al., 2004)

Kelas memiliki tiga area pokok yaitu :

1. Nama, kelas haruslah mempunyai sebuah nama.
2. Atribut, merupakan kelengkapan yang melekat pada kelas. Suatu kelas memiliki nilai yang hanya bisa diproses sebatas pada atribut yang dimiliki.

3. Operasi, merupakan proses yang dilakukan oleh sebuah kelas kepada kelas itu sendiri ataupun kelas lainnya.

2.2.3 *Activity Diagram*

Diagram *activity* menunjukkan aktivitas sistem yang berbentuk kumpulan dari aksi-aksi, bagaimana masing-masing aksi tersebut dimulai, keputusan yang mungkin terjadi hingga berakhirnya aksi. Selain itu, *activity diagram* dapat menggambarkan lebih dari satu proses aksi dalam waktu yang bersamaan. “*Activity diagram* adalah aktifitas-aktifitas, objek, *state*, *transisi state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas”. (Haviluddin, 2011)

2.2.4 *Sequence Diagram*

“Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case diagram*.” (Haviluddin, 2011)

2.3 Pengantar Flask

Flask merupakan *web framework* yang memiliki dasar bahasa pemrograman *python*. *Web framework* adalah koleksi dari modul-modul dan *packages* yang membuat pengembang dapat membuat aplikasi *web* atau *web service* tanpa harus memikirkan detail-detail dasar seperti protokol, soket, atau manajemen proses.

Core yang dimiliki *flask* tergolong sederhana dan bersifat ringan, selain itu *flask* juga bersifat *simplicity* dan *flexibility* sehingga pengembangan dapat menyesuaikan dengan kebutuhan oleh penambahan ekstensi yang ada. Ekstensi yang dimiliki *flask* salah satunya adalah *blueprint*. *Blueprint* memiliki fungsi untuk

mempermudah dalam pembuatan pengaturan minimal RESTful APIs. RESTful APIs merupakan layanan atau metode yang berfungsi untuk mentransmisikan data dengan menggunakan protokol HTTP.

Routing pada *flask* diartikan sebagai bantuan ekstensi *blueprint* yang mempermudah akses kepada beberapa metode *Hypertext Transfer Protocol*(HTTP) hanya dengan mendefinisikan metode yang digunakan pada *routing* yang akan digunakan. Metode permintaan HTTP yang bisa gunakan antara lain sebagai berikut:

Tabel 2.1: Fungsi-fungsi Metode HTTP

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

```

import functools

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

from flaskr.db import get_db

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

```

Gambar 2.1: Contoh registrasi Blueprint pada dokumen “auth.py”.
Sumber: <https://flask.palletsprojects.com/en/2.1.x/tutorial/views/>

Gambar 2.1 berisi *code* pada halaman sebelumnya berfungsi untuk menterjemahkan penggunaan ekstensi *blueprint* pada suatu dokumen bernama ‘auth’. Agar *routing* dapat berjalan maka harus di registrasikan pada dokumen init.py yang merupakan tempat *flask* akan berjalan.

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

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

    return app
```

Gambar 2.2: Dokumen __init__.py.

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

Gambar 2.3 merupakan contoh dari pembuatan *routing login* dengan URL *routing* “/login” yang mendefinisikan metode *GET* dan *POST*. Saat proses *log in* sukses maka akan diarahkan ke URL “login.html”.

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

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

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

            flash(error)

    return render_template('auth/login.html')
```

Gambar 2.3: Contoh pembuatan *routing* dengan *Blueprint*.

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

2.4 MongoDB

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

Kunci perbedaan NoSQL dan *Relational Database Management System* (RDBMS) ialah bagaimana sebuah data dimodelkan pada *database*. RDBMS menggunakan pemodelan yang masih menggunakan tabel berstruktur dengan setiap kolom baris bersifat tetap antara satu dengan lainnya, sedangkan pemodelan data pada NoSQL, khususnya pada MongoDB, menggunakan dokumen dimana setiap barisnya mempunyai kolom yang dapat berbeda dengan baris yang lain. berikut merupakan contoh perbedaan basis data menggunakan konsep RDBMS dan NoSQL dapat dilihat pada **Gambar 2.4** (Pemodelan RDBMS) dan **Gambar 2.5** (Permodelan NoSQL).

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

Gambar 2.4: Pemodelan RDBMS.

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

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

Gambar 2.5: Pemodelan NoSQL.

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

Pada **Gambar 2.4** pada halaman sebelumnya, digunakan untuk menyimpan data *user* dan data *hobbies* dibutuhkan dua tabel terpisah dimana hal ini tidak dibutuhkan pada pemodelan NoSQL (**Gambar 2.5**) yang dapat menggabungkan dua data *user* dan *hobbies* pada satu dokumen serta baris yang sama. Dengan NoSQL ketika ingin memanggil dua data tersebut secara bersamaan hanya membutuhkan satu dokumen saja tanpa menggunakan *joins*, yang menghasilkan *queries* jauh lebih cepat dibandingkan dengan RDBMS.

1. Integrasi MongoDB dan *Flask*

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

```

from flask import Flask
from flask_pymongo import PyMongo

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

```

Gambar 2.6: Menambahkan Ekstensi PyMongo pada Flask.

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

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

(a) Membuat Dokumen

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

```

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

```

Ketika mendefinisikan lebih dari satu data yang sama *BulkWriteError*

akan muncul, yang berarti hanya ada satu data yang terekam dan data lainnya yang sama akan hilang. Untuk mencegah hal tersebut, parameter *ordered* pada fungsi *insert_many()* harus didefinisikan sebagai *false* kemudian menangkap eksepsi *BulkWriteError*.

(b) Membaca Dokumen

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

(c) Memperbarui dan Mengganti Dokumen

Metode yang digunakan dalam memperbarui data pada *database* adalah *update_one()* atau *replace_one()*. Metode *replace_one()* mempunyai beberapa argumen sebagai berikut:

- i. *Filter*: berupa *query* yang mendefinisikan data pada ID yang akan diganti,
- ii. *Replacement*: berupa data yang akan menggantikan data yang dihapus.
- iii. *Upsert*: adalah opsi *boolean* yang jika dijadikan sebagai *true* dapat membuat dokumen baru jika tidak terdapat target dokumen yang dimaksud.

(d) Menghapus Dokumen

PyMongo menyediakan dua metode untuk menghapus satu atau lebih

koleksi *database* yaitu, *delete_one()* untuk menghapus satu koleksi dan *delete_many()* untuk menghapus beberapa koleksi.

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

Contoh kode di atas ketika menjalankan *request* seperti http://localhost:5000/delete_todo/5 PyMongo akan mencari entri berdasarkan ID yang diberikan dan menghapusnya.

(e) Menyimpan dan Menerima *Files*

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

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

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

2.5 Scrum

Scrum merupakan salah satu struktur kerja yang digunakan untuk mengembangkan produk. *Scrum* diumumkan pertama kali oleh Ken Schwaber pada tahun 1995 pada konferensi Austin, namun fondasi metode *scrum* sudah ada sejak

tahun 1980 (Ozierańska et al., 2016). *Scrum* dibuat berdasarkan empirisme yang dicapai dengan beberapa kualitas. Hasil survei dari literatur, kualitas yang membangun empirisme *scrum* adalah kejelasan dari setiap proses, inspeksi untuk mendeteksi masalah dan adaptasi terhadap perubahan

Setiap produk dihantarkan dengan cara yang fleksibel dan iteratif dalam kerangka kerja *scrum* dimana setiap akhir *sprint* terdapat produk nyata yang dapat dihantarkan. *Requirement* yang dibutuhkan dalam suatu proyek berupa *product backlog* yang diperbarui secara berkala.

Scrum mempunyai tiga elemen, di antaranya:

1. *Roles*

Role dalam *scrum* terbagi menjadi empat *role* utama, yaitu:

(a) Tim *Scrum*

Tim *scrum* merupakan kelompok kecil yang terdiri dari satu *scrum master*, satu *product owner*, dan pengembang. Pada tim *scrum* tidak terdapat tim kecil ataupun hierarki. Seluruh anggota tim memiliki kemampuan penting untuk memberikan nilai ke dalam setiap *sprint* dan fokus dengan satu tujuan pada satu waktu, *product goal*.

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

(b) *Scrum Master*

Scrum master memiliki tanggung jawab dalam merealisasikan *scrum* yang terdefinisi pada panduan *scrum*. Setiap anggota tim dibantu *scrum master* untuk mengerti bagaimana teori dan praktik kerangka kerja pada metode *scrum*. Selain itu, menjaga efektivitas dari tim *scrum* juga menjadi tanggung jawab *scrum master*.

(c) *Product Owner*

Product owner memiliki tanggung jawab untuk meningkatkan nilai komersial produk yang dihasilkan oleh *development team* dan mengelola *product backlog* agar lebih maksimal. Hanya *product owner* yang memiliki tanggung jawab untuk mengelola *product backlog*. Adapun pengelolaan *product backlog*:

- i. Penyampaian isi *product backlog*.
- ii. Memastikan *development team* memahami *product backlog*.
- iii. Memastikan isi daripada *product backlog* transparan dan jelas bagi seluruh anggota tim.
- iv. Mengurutkan item pada *product backlog* untuk mencapai tujuan secara optimal.

(d) *Development Team*

Development team atau tim pengembang adalah profesional yang mengeksekusi isi yang tercantum di dalam *product backlog*. Tim Pengembang berkomitmen untuk membuat semua aspek *increment* yang dapat berfungsi pada setiap *sprint*. Namun, tim Pengembang juga selalu bertanggung jawab untuk:

- i. Membuat rancangan *sprint* atau dikenal dengan *sprint backlog*.

- ii. Membuat definisi penyelesaian sebuah *task*.
- iii. Mengadaptasikan semua *plan* setiap hari sampai *sprint goal*.
- iv. Mengurutkan *item* pada *product backlog* untuk mencapai tujuan secara optimal.

2. *Artifacts*

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

(a) *Product Backlog*

Product backlog atau umumnya disebut dengan *user stories* merupakan kumpulan fitur-fitur yang terdapat pada suatu produk. *User stories* dapat ditambahkan, dimodifikasi, atau dihilangkan dari *product backlog* selama proyek berjalan.

(b) *Sprint Backlog*

Sprint backlog adalah beberapa *user stories* yang diambil dari *product backlog* untuk dijalankan pada satu *sprint*. *Sprint backlog* mencakup seluruh kegiatan kerja yang diperlukan untuk mencapai *sprint goal*.

Pada satu *sprint* terdapat *increment* yang merupakan manifestasi dari *user stories* yang diselesaikan dan total *increment* dari seluruh *sprint* sebelumnya.

3. Events

Event merupakan wadah dari semua *event* yang terdapat pada *scrum*. *Event* dibuat sebagai perwujudan salah satu dari tiga pilar *scrum*. Seluruh *event* berjalan secara bersamaan untuk mengurangi kompleksitas.

(a) *Sprint*

Sprint adalah komponen utama kerangka kerja *scrum*, dimana sebuah ide menjadi sebuah nilai. Lama durasi *sprint* bersifat tetap yaitu satu hingga empat minggu untuk menjaga konsistensi.

Sprint berfokus untuk menghantarkan beberapa *user stories* pada *product backlog*. Setiap satu *sprint* memiliki beberapa kegiatan diantaranya *sprint planning*, *sprint review* dan *sprint retrospective*.

(b) *Sprint Planning*

Sprint backlog adalah beberapa *user stories* yang diambil dari *product backlog* untuk dijalankan pada satu *sprint*. *sprint backlog* mencakup semua kegiatan kerja yang dibutuhkan untuk mencapai *sprint goal*.

Sebelum memulai *sprint*, perencanaan apa yang akan dilaksanakan pada saat *sprint* dilakukan pada saat *sprint planning* oleh seluruh anggota tim *scrum*. Waktu untuk melaksanakan *sprint planning* terbatas dengan lama durasi hingga delapan jam. Fungsi dari *sprint planning* adalah memutuskan apa yang dapat dimasukkan ke dalam *increment* dari *sprint* dan bagaimana penyelesaian yang dibutuhkan untuk menghantarkan *increment*.

(c) *Daily Scrum*

Daily scrum adalah kegiatan 15 menit bagi para pengembang untuk memeriksa perkembangan menuju *sprint goal* dan menyesuaikan pekerjaan yang akan dikerjakan selama 24 jam ke depan. Pada kegiatan *daily scrum*, hal apa saja yang akan dikerjakan hari ini, apa yang telah dikerjakan kemarin dan hambatan yang telah dialami dalam mencapai *sprint goal* akan didiskusikan oleh tim pengembang.

(d) *Sprint Review*

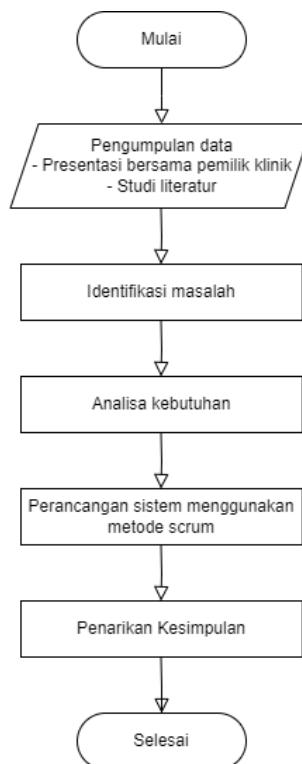
Tahap ini dilaksanakan pada akhir *sprint*, tujuannya untuk mengawasi apa yang telah diselesaikan di *sprint*. Menurut hasil tinjauan serta perubahan *product backlog*, tim *scrum* menentukan kembali pekerjaan selanjutnya yang dapat mengoptimalkan produk.

Sprint review bersifat informal dan diselenggarakan dengan lama durasi empat jam untuk sprint dalam waktu satu bulan. Semakin singkat durasi *sprint*, maka semakin singkat juga durasi *sprint review*.

BAB III

METODOLOGI PENELITIAN

Melalui penelitian yang dilakukan oleh penulis, akan menghasilkan produk tertentu. Penelitian yang dilakukan oleh penulis juga termasuk dalam jenis penelitian dan pengembangan. Berikut adalah tahapan-tahapan penelitian yang penulis lakukan dalam perancangan sebuah aplikasi:



Gambar 3.1: Tahapan penelitian

3.1 Pengumpulan Data

Peneliti mengambil data dari presentasi bersama dengan pemilik klinik *moist care* dan klien dari penelitian ini yaitu ibu Irma Puspita Arisanti. Untuk dokumentasi foto pada saat presentasi dapat dilihat pada Lampiran A. Peneliti juga melakukan

studi literatur dengan membaca jurnal-jurnal yang berkaitan dengan topik penelitian serupa.

3.2 Analisa Kebutuhan

Berikut merupakan perangkat keras dan perangkat lunak yang penulis butuhkan dalam merancang sistem informasi keperawatan luka:

Perangkat keras berupa:

1. Laptop dengan spesifikasi Processor Intel Core i5 generasi ke-3 dan RAM 12 GB.

Perangkat lunak berupa:

1. Windows 10 *Operating System*.
2. Figma sebagai alat untuk mendesain tampilan UI/UX.
3. *Visual Studio Code* untuk pembuatan sistem informasi keperawatan luka.
4. *Python* sebagai bahasa pemrograman yang peneliti gunakan.
5. *Flask* sebagai web framework yang akan digunakan.
6. MongoDB sebagai basis data

3.3 Perancangan Sistem Menggunakan *Scrum*

Website aplikasi yang dibuat dalam penelitian ini dikembangkan dengan menggunakan metode *scrum*. Penjelasan rinci tentang metode *scrum* akan disajikan pada sub bab di bawah ini.

3.3.1 *Product Backlog*

Tahap *Product backlog* ini berfungsi untuk menterjemahkan seluruh fitur yang akan diimplementasikan pada aplikasi. Rincian *Product Backlog* yang akan diimplementasikan pada *website* aplikasi dapat dilihat pada tabel di bawah ini.

Tabel 3.1: *Product Backlog*

No	User Story	Priority	Sprint No.
1	Pembuatan akun pasien	<i>High</i>	1 & 2
2	Dashboard klinik	<i>High</i>	1 & 2
3	Pemeriksaan kesehatan	<i>High</i>	1 & 3
4	Proses pengobatan luka	<i>High</i>	1 & 3
5	Pendaftaran pasien berobat	<i>High</i>	1 & 3
6	Pengelolaan antrian	<i>High</i>	1 & 4
7	Administrasi keuangan	<i>High</i>	1 & 4

Product backlog yang dibuat memiliki 4 kolom yang di antaranya adalah sebagai berikut:

1. *User Story*

Kolom *user story* berisi fitur-fitur yang akan dibuat pada aplikasi.

2. *Priority*

Kolom *priority* berisi tingkat prioritas dari *user story*, dimana prioritas *high* merupakan fitur yang mempunyai peran penting pada penelitian ini.

3. *Sprint No.*

Kolom *sprint no.* berisi informasi tentang urutan pelaksanaan fitur *sprint* tersebut akan dibuat.

3.3.2 *Sprint Backlog*

Sebelum *sprint* dimulai dilakukan *sprint backlog*, *sprint backlog* berisikan daftar pekerjaan yang keputusannya diambil dari *product backlog*. Dengan adanya *sprint backlog* semua anggota tim bisa melihat perkembangan dari setiap pekerjaan. Pada penelitian peneliti menggunakan tiga status perkembangan yaitu harus dikerjakan, sedang dikerjakan, selesai, next *sprint* dan tidak selesai.

3.3.3 *Sprint*

Setelah perencanaan *sprint backlog* sudah dibuat, maka penggeraan *sprint* sudah bisa dimulai dan mengikuti jadwal penggeraan yang telah disepakati bersama tim. Interval *sprint* yang digunakan adalah dua minggu.

3.3.4 *Deploy*

Setelah semua pekerjaan *sprint* yang telah direncanakan pada *sprint backlog* selesai maka aplikasi akan di *deploy*.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pembahasan

Perancangan sistem informasi keperawatan luka dilakukan dengan menggunakan metode *scrum*. Pada metode *scrum*, proses pengembangan sistem dilakukan secara bertahap dikenal sebagai *sprint*. Penelitian ini memiliki empat *sprint* dimana satu putaran *sprint* berdurasi selama dua minggu. Setiap awal pekan, dilakukan perencanaan *sprint backlog* berdasarkan *product backlog* telah disepakati.

4.1.1 *Sprint-1*

Tabel 4.1: *Sprint-1 backlog*

No	User Story	Task
1	Pembuatan akun pasien	<ol style="list-style-type: none">1. Pembuatan <i>mock up</i> tampilan pembuatan akun pasien.2. Desain <i>routing table</i> pembuatan akun pasien.3. Membuat <i>database</i> pasien
2	<i>Dashboard</i> klinik	<ol style="list-style-type: none">1. Pembuatan <i>mock up</i> tampilan <i>dashboard</i> klinik.2. Desain <i>routing table dashboard</i> klinik.

Tabel 4.2: Sprint-1 backlog lanjutan 1

No	User Story	Task
3	Pemeriksaan kesehatan	<ul style="list-style-type: none"> 1. Desain <i>routing table</i> pemeriksaan kesehatan. 2. Membuat <i>database</i> pemeriksaan kesehatan.
4	Proses pengobatan luka	<ul style="list-style-type: none"> 1. Pembuatan <i>mock up</i> tampilan proses pengobatan. 2. Desain <i>routing table</i> proses pengobatan. 3. Membuat <i>database</i> proses pengobatan.
5	Pendaftaran pasien berobat	<ul style="list-style-type: none"> 1. Pembuatan <i>mock up</i> tampilan pendaftaran pasien berobat. 2. Desain <i>routing table</i> pendaftaran pasien berobat. 3. Membuat <i>database</i> pendaftaran pengobatan.
6	Pengelolaan antrian	<ul style="list-style-type: none"> 1. Pembuatan <i>mock up</i> tampilan pengelolaan antrian. 2. Desain <i>routing table</i> pengelolaan antrian. 3. Membuat <i>database</i> pengelolaan antrian.

Tabel 4.3: Sprint-1 backlog lanjutan 2

No	User Story	Task
7	Administrasi keuangan	<ol style="list-style-type: none"> 1. Pembuatan <i>mock up</i> tampilan administrasi keuangan. 2. Desain <i>routing table</i> administrasi keuangan. 3. Membuat <i>database</i> administrasi keuangan.

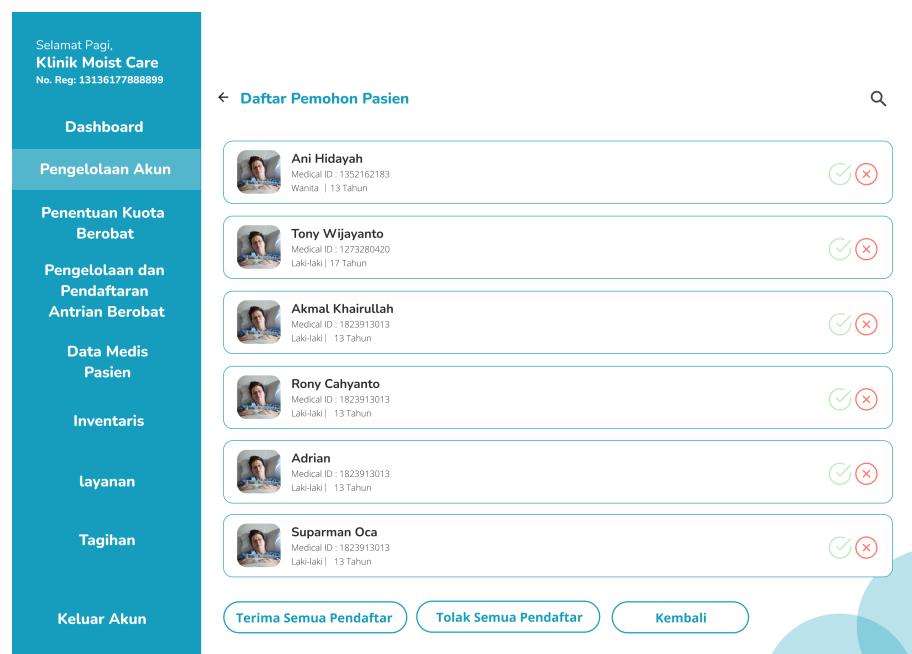
1. Pembuatan *mock up* pembuatan akun

**Gambar 4.1:** Mock up tampilan awal pembuatan akun pasien

Gambar 4.1 merupakan *mock up* tampilan awal membuat akun pasien. Pada tampilan ini di menu pasien terdapat tombol daftar permohonan pasien untuk melihat *list* akun pasien yang menunggu untuk disetujui pembuatannya. Lalu ada tombol tambah pasien untuk membuat akun pasien baru melalui

klinik. Selanjutnya ada tombol daftar pasien untuk melihat semua akun pasien yang terdaftar pada klinik. Di bawah menu pasien ada menu perawat yang berisi tombol tambah perawat untuk membuat akun perawat baru. Lalu tombol daftar perawat untuk melihat semua akun perawat yang terdaftar pada klinik. Selanjutnya di bawah menu perawat ada menu admin yang berisi tombol tambah admin untuk membuat akun admin baru dan ada tombol daftar admin untuk melihat seluruh akun admin yang terdaftar.

Berikutnya saat admin menekan tombol daftar pemohon pada **Gambar 4.1** maka akan diarahkan ke **Gambar 4.2**.



Gambar 4.2: Mock up tampilan *list* permohonan pembuatan akun pasien

Gambar 4.2 merupakan tampilan *list* permohonan pembuatan akun pasien. Berisi *list* akun yang menunggu untuk disetujui pembuatan akunnya oleh klinik. Di bawah terdapat tombol terima semua pendaftar untuk menyetujui semua akun yang ada pada *list* permohonan pasien. Lalu ada tombol tolak semua pendaftar untuk menolak semua akun yang ada pada *list* permohonan

pasien. Dan ada tombol kembali untuk kembali ke tampilan awal pembuatan akun pasien.

Berikutnya ketika admin menekan tombol tambah pasien pada **Gambar 4.1** maka akan diarahkan ke **Gambar 4.3**.

Gambar 4.3: Mock up pembuatan akun pasien

Gambar 4.3 merupakan tampilan pembuatan akun pasien. Saat membuat akun pasien data yang harus diisi adalah medical ID, password, email, nama lengkap, tanggal lahir, agama, Nomor HP yang bisa dihubungi, Nomor BPJS, alamat, foto dan jenis kelamin pasien. Lalu di bawah ada tombol tambah pasien untuk menyimpan akun pasien dan ada tombol batal untuk membatalkan tambah akun pasien yang mengarah kembali ke **Gambar 4.1**.

2. Desain *routing table* pembuatan akun pasien

Tabel 4.4 pada halaman selanjutnya merupakan *routing table* pembuatan akun pasien:

Tabel 4.4: Routing table pembuatan akun pasien

Group	Name	API Endpoint	HTTP Verb	Keterangan	Return Type
Pasien	<i>CREATE</i>	/add_new_patient	<i>POST</i>	menampilkan halaman tambah pasien oleh klinik	<i>view</i>
	<i>READ</i>	/list_patient	<i>GET</i>	Menampilkan halaman seluruh pasien yang sudah terverifikasi oleh klinik	<i>view</i>
	<i>READ</i>	/list_request_new_patient	<i>GET</i>	menampilkan halaman list pasien yang membuat akun mandiri dan belum terverifikasi oleh klinik	<i>view</i>
Pasien	<i>READ</i>	/profil_pasien /<_id>	<i>GET</i>	Menampilkan halaman data pasien berdasarkan id pasien	<i>view</i>

3. Membuat *database* pasien

Gambar 4.4 pada halaman selanjutnya merupakan tabel pasien yang memuat beberapa atribut yaitu *id_pasien*, *password*, *id_staff_klinik*, *nama* dan *agama*, *tanggal_lahir*, *usia*, *jenis_kelamin*, *alamat*, *no_hp*, *email*, *no_bpjs*, *verif*, *list_image_id*, *created_at* dan *update_at*. Data tersebut akan disimpan pada *database* MongoDB dan disimpan dalam format JSON. Tabel atau *collection* pasien akan dibuat bersamaan dengan pemanggilan REST API pembuatan

akun pasien.

User / Pasien	
ID_Pasien	U
Password	
ID_Staff_Klinik	U
Nama	
NIK	
Agama	
Tanggal_Lahir	
Usia	
Jenis_Kelamin	
Alamat	
No_HP	U
Email	U
No_BPJS	
Verif	
List_Image_ID	
Created_At	
Update_At	

Gambar 4.4: Tabel Pasien

4. Pembuatan *mock up dashboard* klinik



Gambar 4.5: *Mock up* tampilan awal dashboard klinik

Gambar 4.5 merupakan tampilan *dashboard* klinik. Admin dapat melihat

statistik kinerja perawat, *cost* perawatan pasien, *income* masuk dan *balance* keuangan dengan detail rata-rata harian, mingguan, dan bulanan.

5. Desain *routing table dashboard* klinik

Tabel 4.5 merupakan *routing table* untuk *dashboard* klinik:

Tabel 4.5: Routing table dashboard klinik

Group	Name	API Endpoint	HTTP Verb	Keterangan	Return Type
<i>Dashboard</i>	<i>READ</i>	/	<i>GET</i>	Menampilkan halaman <i>dashboard</i> klinik	<i>view</i>

6. Desain *routing table* pemeriksaan kesehatan

Tabel 4.6 dan **Tabel 4.7** merupakan *routing table* untuk pemeriksaan kesehatan:

Tabel 4.6: Routing table pemeriksaan kesehatan

Group	Name	API Endpoint	HTTP Verb	Keterangan	Return Type
Pemeriksaan Kesehatan	<i>READ</i>	<i>/list_patient_medical_check</i>	<i>GET</i>	Menampilkan halaman <i>list</i> pasien yang telah melakukan pemeriksaan kesehatan	<i>view</i>
	<i>READ</i>	<i>/list_medical_check_data/<nik></i>	<i>GET</i>	Menampilkan halaman <i>list</i> hasil pemeriksaan kesehatan 1 pasien	<i>view</i>

Tabel 4.7: Routing table pemeriksaan kesehatan - lanjutan

Group	Name	API Endpoint	HTTP Verb	Keterangan	Return Type
Pemeriksaan Kesehatan	<i>READ</i>	/detail_ _medical_ check_data /<_id>	GET	Menampilkan halaman detail hasil pemeriksaan kesehatan	<i>view</i>

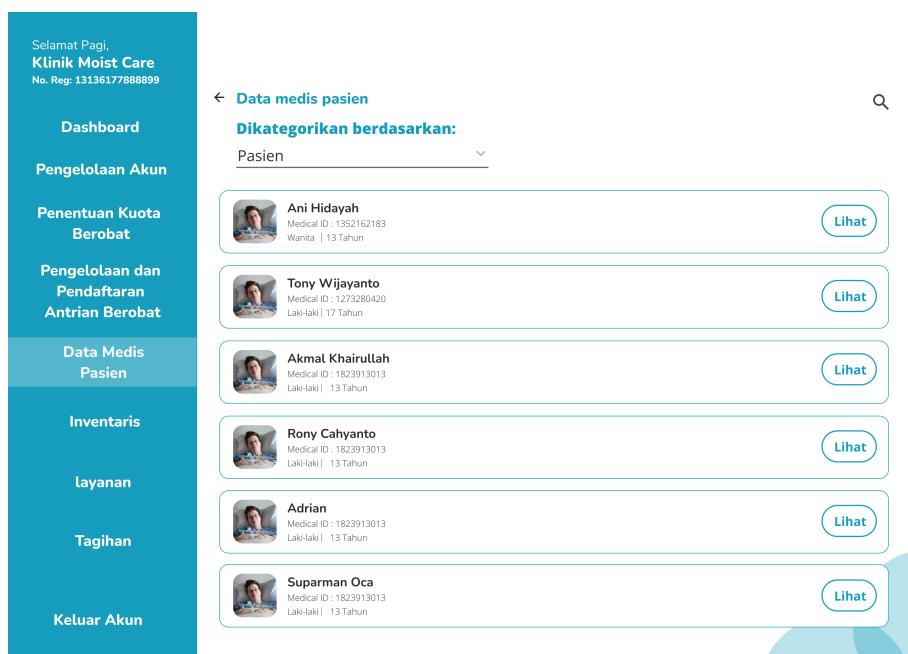
7. Membuat *database* pemeriksaan kesehatan

**Gambar 4.6:** Tabel pemeriksaan kesehatan

Gambar 4.6 merupakan tabel pemeriksaan kesehatan memuat beberapa atribut yaitu id_pk, NIP, tanggal, NIK pasien, tekanan_darah, nadi, jenis_kelamin, alamat, no_hp, email, no_bpjs, verif, suhu, GDS dan ABPI . Data tersebut akan disimpan pada *database* MongoDB dan disimpan dalam format JSON.

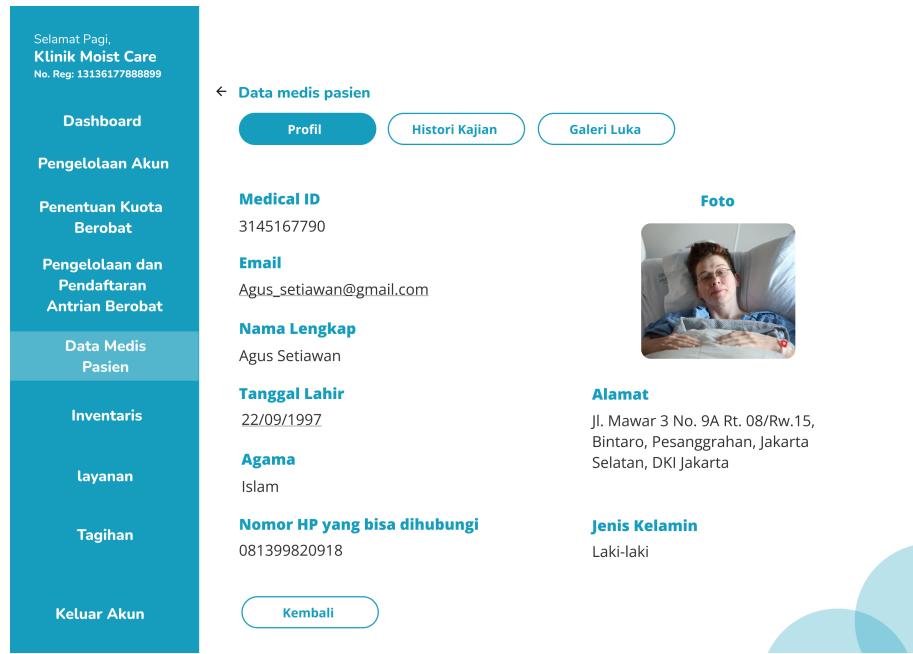
8. Pembuatan *mock up* proses pengobatan luka

Pada **Gambar 4.7** pada halaman selanjutnya merupakan tampilan awal data medis pasien. Terdapat *list* data medis pasien beserta tombol lihat. Admin maupun perawat dapat menyeleksi data berdasarkan pasien atau perawat pada *field* dikategorikan berdasarkan perawat. Dan terdapat juga *search bar* untuk mencari data medis pasien berdasarkan kata kunci *Medical ID* atau nomor BPJS.



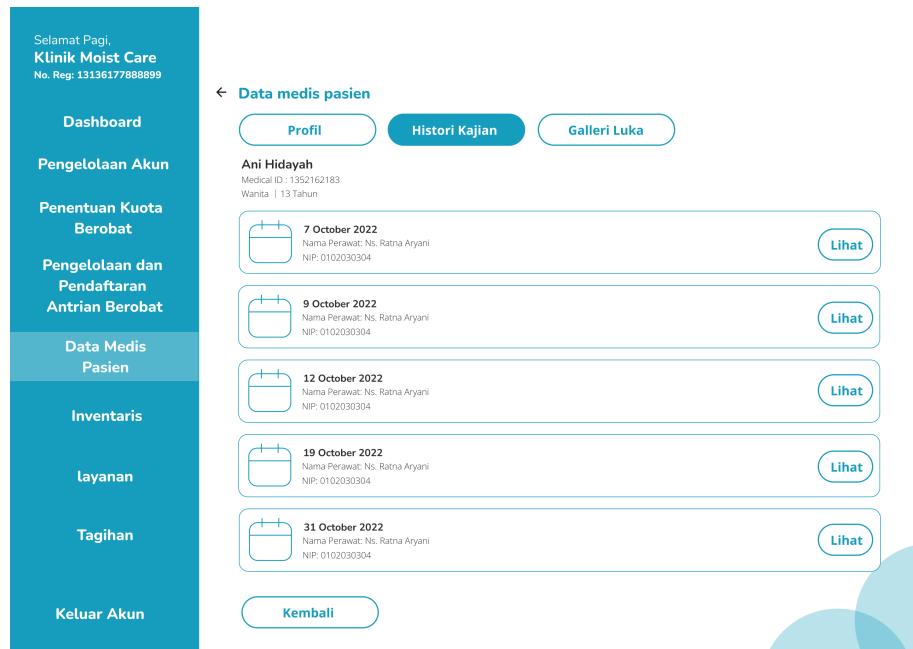
Gambar 4.7: *Mock up* tampilan awal data medis pasien

Ketika admin menekan tombol lihat pada data pasien yang dipilih pada **Gambar 4.7** maka akan diarahkan ke **Gambar 4.8** pada halaman selanjutnya yang merupakan profil data medis pasien dan berisikan data berupa *Medical ID*, email, nama lengkap, tanggal lahir, agama, nomor hp, alamat, jenis kelamin, foto, dan nomor BPJS. Selanjutnya terdapat tombol profil, histori kajian dan galeri luka. Dan terdapat tombol kembali untuk menuju ke halaman sebelumnya.



Gambar 4.8: Mock up profil data medis pasien

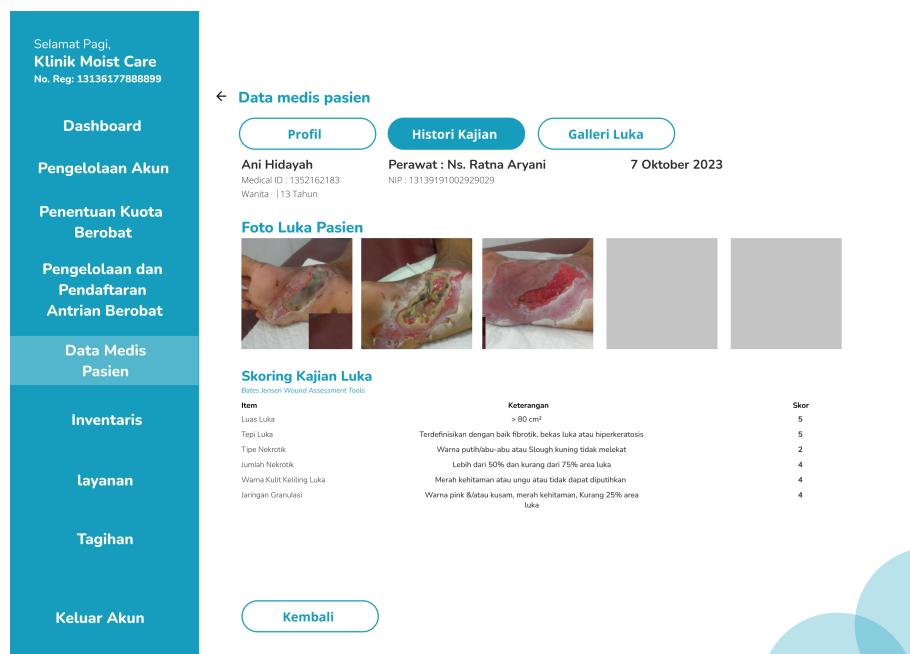
Ketika admin menekan tombol histori kajian pada **Gambar 4.8** maka akan diarahkan ke **Gambar 4.9**.



Gambar 4.9: Mock up list histori kajian data medis pasien

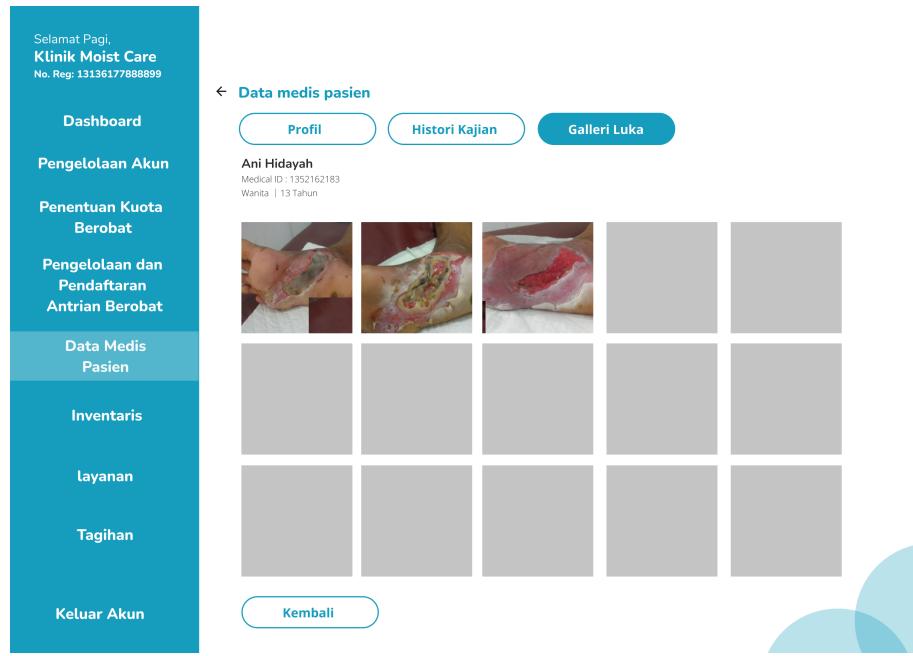
Gambar 4.9 merupakan histori kajian data medis pasien berisi *list* kajian yang telah dilaksanakan dan diurutkan berdasarkan tanggal dengan detail data nama dan nip perawat yang melakukan kajian beserta tombol lihat. Dan terdapat tombol kembali untuk menuju ke halaman sebelumnya.

Ketika admin menekan tombol lihat pada salah satu histori kajian pada **Gambar 4.9** maka akan diarahkan ke **Gambar 4.10** yang merupakan detail histori kajian data medis pasien berisi foto luka yang diambil saat kajian berlangsung dan skoring kajian luka pasien. Dan terdapat tombol kembali untuk kembali ke halaman sebelumnya.

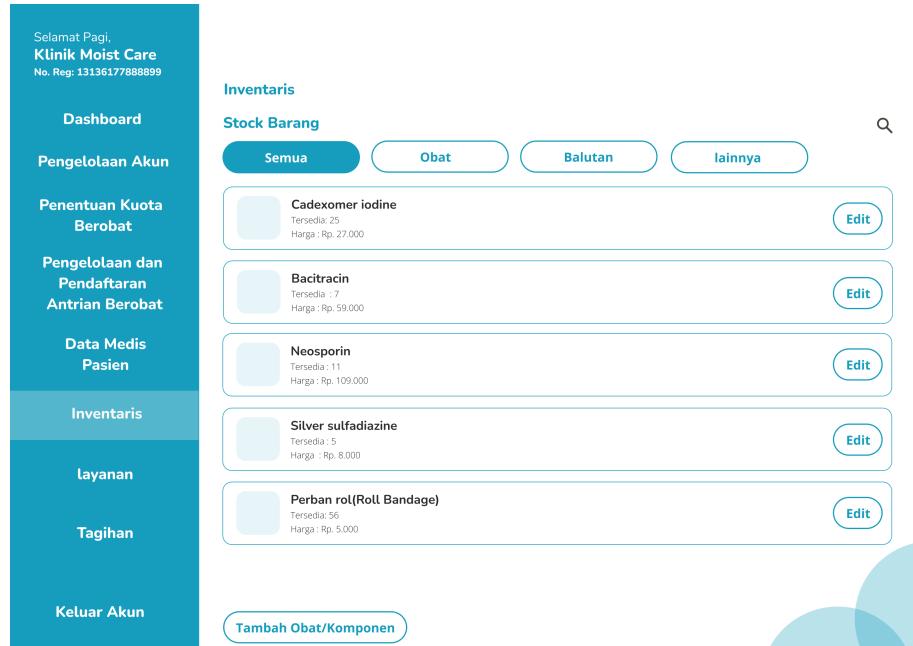


Gambar 4.10: Mock up detail histori kajian data medis pasien

Ketika admin menekan tombol galeri luka **Gambar 4.8** maka akan diarahkan ke **Gambar 4.11** pada halaman selanjutnya yang merupakan galeri luka pasien dari semua histori kajian. Dan terdapat tombol kembali untuk menuju halaman sebelumnya.



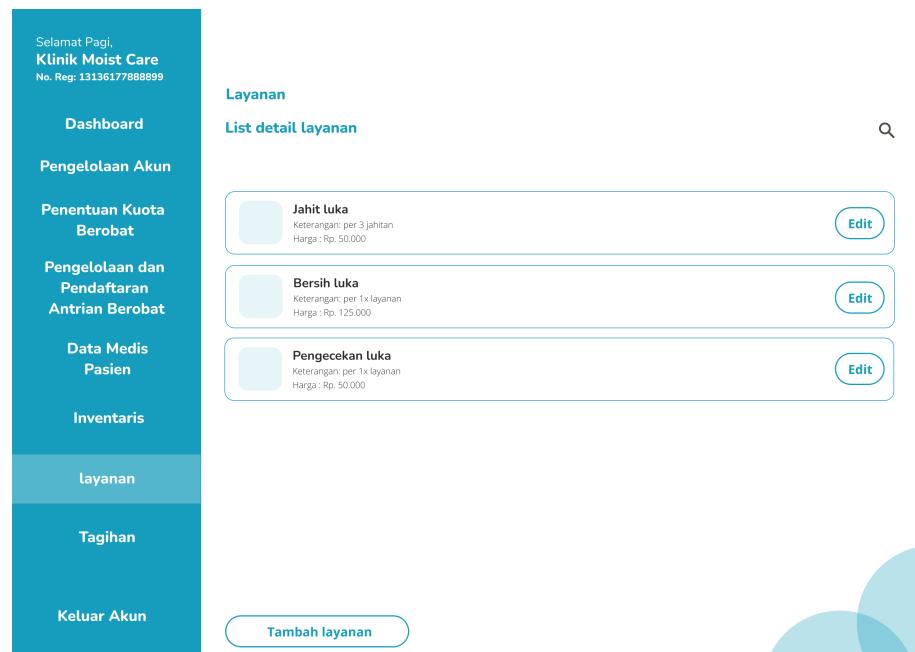
Gambar 4.11: Mock up galeri luka pasien



Gambar 4.12: Mock up tampilan awal inventaris

Gambar 4.12 merupakan tampilan awal inventaris didalamnya terdapat *list* barang berupa obat, balutan, dan komponen lainnya beserta detail stok,

harganya dan tombol *edit*. Terdapat juga tombol semua barang, obat, balutan, lainnya dan tambah obat/komponen.



Gambar 4.13: *Mock up* tampilan awal layanan

Gambar 4.13 merupakan tampilan awal layanan didalamnya terdapat *list* layanan beserta detail keterangan layanan, harganya dan tombol *edit*.

9. Desain *routing table* proses pengobatan

Tabel 4.8, Tabel 4.9 dan Tabel 4.10 merupakan *routing table* untuk proses pengobatan:

Tabel 4.8: *Routing table* proses pengobatan

Group	Name	API Endpoint	HTTP Verb	Keterangan	Return Type
Pengobatan	<i>READ</i>	/data_kajians	<i>GET</i>	Menampilkan halaman data kajian semua pasien berdasarkan nama pasien	<i>view</i>
	<i>READ</i>	/data_kajians /<nip>	<i>GET</i>	Menampilkan data kajian semua pasien berdasarkan nama perawat	{<nip>, <nama_pegawai>, <nrm>, <nama_pasien>}
	<i>READ</i>	/data_kajians /<nrm>	<i>GET</i>	Menampilkan data kajian semua pasien berdasarkan nomor rekam medis	{<nrm>, <nama_pasien>}
	<i>READ</i>	/profil_pasien /<nrm>	<i>GET</i>	Menampilkan halaman profil 1 pasien berdasarkan NRM	<i>view</i>

Tabel 4.9: Routing table proses pengobatan - lanjutan 1

Group	Name	API Endpoint	HTTP Verb	Keterangan	Return Type
Pengo- batan	<i>READ</i>	/list_data _kajian /<nrm>	<i>GET</i>	Menampilkan halaman <i>list</i> histori kajian 1 pasien berdasarkan NRM	<i>view</i>
	<i>READ</i>	/galeri_luka /<nrm>	<i>GET</i>	Menampilkan halaman galeri luka pasien berdasarkan NRM	<i>view</i>
Inven -taris	<i>READ</i>	/list_ inventaris	<i>GET</i>	Menampilkan halaman <i>list</i> semua inventaris	<i>view</i>
	<i>CREATE</i>	/add_ inventaris	<i>POST</i>	Menampilkan halaman tambah inventaris	<i>view</i>
	<i>UPDATE</i>	/edit_ inventaris /<_id>	<i>POST</i>	Menampilkan halaman <i>edit</i> inventaris	<i>view</i>
	<i>DESTROY</i>	/delete_ inventaris /<_id>	<i>DELETE</i>	Hapus data inventaris	<i>view</i>
Laya -nan	<i>READ</i>	/list_ layanan	<i>GET</i>	Menampilkan halaman <i>list</i> semua layanan	<i>view</i>

Tabel 4.10: Routing table proses pengobatan - lanjutan 2

Group	Name	API Endpoint	HTTP Verb	Keterangan	Return Type
Layan-	<i>CREATE</i>	/add_ layanan	<i>POST</i>	Menampilkan halaman tambah layanan	<i>view</i>
	<i>UPDATE</i>	/edit _layanan /<_id>	<i>POST</i>	Menampilkan halaman <i>edit</i> layanan	<i>view</i>
	<i>DESTROY</i>	/delete _layanan <_id>	<i>DELETE</i>	Hapus layanan	<i>view</i>

10. Membuat database proses pengobatan

**Gambar 4.14:** Tabel data kajian

Gambar 4.14 merupakan data kajian memuat beberapa atribut yaitu id_kajian, size, edges, necrotic_type, necrotic_amount, skincolor_surround,

granulation_tissue, *epithelization*, *raw_photo_id*, *tipe_image_id*, *diameter_image*, dan *created_at*. Data tersebut akan disimpan pada *database* MongoDB dan disimpan dalam format JSON.

Inventaris	
!	ID_Inventaris
!	ID_Image
!	Tipe_Inventaris
!	Nama_Inventaris
!	Jumlah
!	Harga
!	Keterangan
!	TagihanID_tagihan

Gambar 4.15: Tabel inventaris

Gambar 4.15 merupakan tabel inventaris memuat beberapa atribut yaitu *id_inventaris*, *id_image*, *tipe_inventaris*, *nama_inventaris*, *jumlah*, *harga*, dan *keterangan*. Data tersebut akan disimpan pada *database* MongoDB dan disimpan dalam format JSON.

Layanan	
!	ID_Layanan
!	ID_Image
!	Tipe_Layanan
!	Nama_Layanan
!	Keterangan
!	Harga

Gambar 4.16: Tabel layanan

Gambar 4.16 merupakan tabel layanan memuat beberapa atribut yaitu *id_layanan*, *id_image*, *tipe_layanan*, *nama_layanan*, *keterangan*, dan *harga*. Data tersebut akan disimpan pada *database* MongoDB dan disimpan dalam format JSON.

11. Pembuatan *mock up* pendaftaran pasien berobat

Pada **Gambar 4.17** di halaman selanjutnya merupakan tampilan *live* yang berisi antrian berobat lengkap beserta nama pasien dan nomor antrian.



Gambar 4.17: Mock up melihat daftar dan urutan pasien yang akan dilayani

- Desain *routing table* pendaftaran pasien berobat.

Tabel 4.11 merupakan *routing table* untuk pendaftaran pasien berobat:

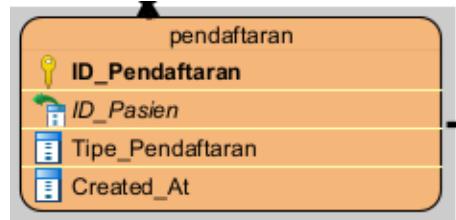
Tabel 4.11: *Routing table* pendaftaran pasien berobat

Group	Name	API Endpoint	HTTP Verb	Keterangan	Return Type
	<i>READ</i>	/antrian <i>_real_time</i>	<i>GET</i>	Menampilkan halaman daftar dan urutan pasien yang akan dilayani	<i>view</i>

- Membuat *database* pendaftaran pasien berobat

Gambar 4.18 pada halaman selanjutnya merupakan tabel detail tagihan memuat beberapa atribut yaitu *id_pendaftaran*, *tipe_pendaftaran*, *id_pasien* dan *created_at*. Data tersebut akan disimpan pada *database* MongoDB dan

disimpan dalam format JSON.



Gambar 4.18: Tabel pendaftaran berobat

14. Pembuatan *mock up* pengelolaan antrian

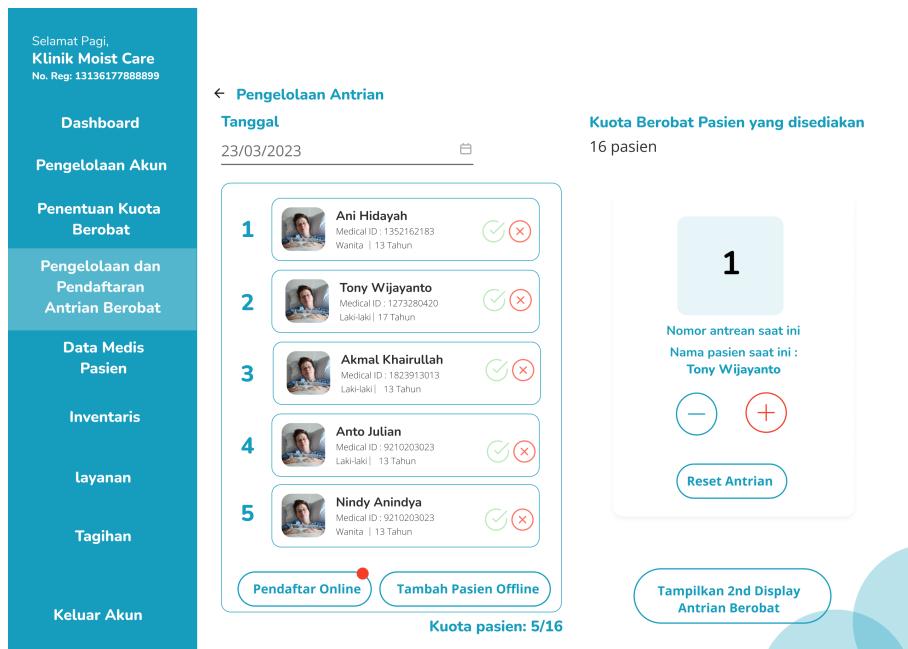
Gambar 4.19 merupakan tampilan menentukan kuota pelayanan pasien yang mendaftar secara *offline* maupun *online* pada hari yang ditentukan.

Gambar 4.19: *Mock up* menentukan kuota pelayanan pasien yang mendaftar secara *offline* maupun *online* pada hari yang ditentukan

Pada halaman ini admin harus memilih dan mengisi tanggal berobat beserta kuota pasien yang daftar berobat secara *online* dan kuota pasien yang mendaftar

secara *offline* atau daftar langsung di klinik. Dan terdapat tombol simpan untuk menyimpan data kuota berobat pada hari yang ditentukan yang telah diisi.

Gambar 4.20 merupakan tampilan awal melakukan pendaftaran pasien yang akan berobat secara *offline* maupun *online*. Terdapat *fields* memilih tanggal untuk mengambil data kuota berobat pada hari yang ditentukan yang telah di-set sebelumnya. Lalu terdapat *list* antrian pasien berobat yang sudah dikonfirmasi pendaftarannya beserta tombol ceklis untuk memvalidasi bahwa pasien sudah mendapatkan pelayanan berobat klinik dan tombol silang untuk membatalkan pasien yang sudah dikonfirmasi pendaftarannya apabila pasien tidak jadi berobat.

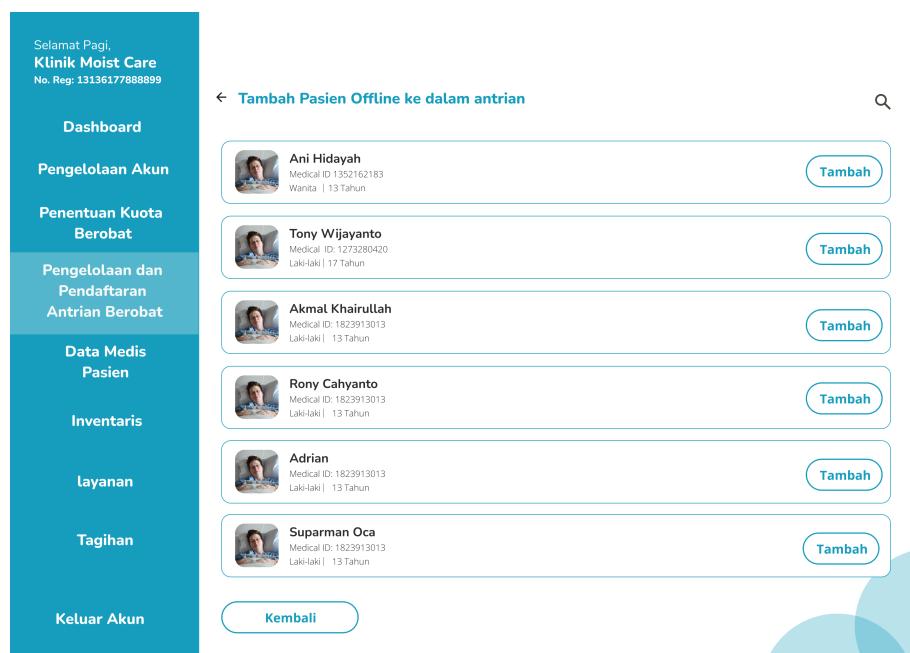


Gambar 4.20: *Mock up* tampilan awal melakukan pendaftaran pasien yang akan berobat secara *offline* maupun *online*

Selanjutnya terdapat tombol pendaftar *online* untuk melihat *list* pasien yang sudah mendaftar berobat secara *online* namun belum dikonfirmasi. Terdapat juga tombol tambah *offline* untuk tambah pasien berobat secara manual

apabila pasien daftar secara *offline* atau langsung, lalu tombol *plus icon* untuk memajukan antrian dan tombol *minus icon* untuk memundurkan antrian. Selanjutnya tombol *reset* antrian untuk me-*reset* antrian kembali ke nomor 1. Dan tombol tampilkan *2nd display* antrian berobat untuk tampilan tv *live* di ruang tunggu pasien.

Ketika admin menekan tombol tambah pasien *offline* pada **Gambar 4.20** pada halaman sebelumnya maka muncul *list* daftar pasien pada **Gambar 4.21** beserta tombol tambah untuk meng-*input* pasien ke dalam antrian berobat dan admin bisa mencari pasien yang ingin daftar berobat dengan mencari lewat *Medical ID* atau nomor BPJS. Dan ada tombol kembali untuk kembali ke tampilan **Gambar 4.15**



Gambar 4.21: Mock up mendaftarkan pasien yang berobat secara *offline*

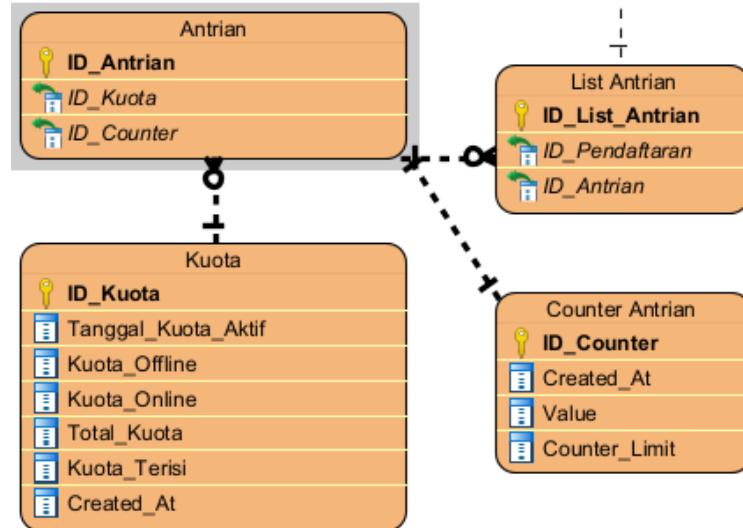
15. Desain *routing table* pengelolaan antrian.

Tabel 4.12 pada halaman selanjutnya merupakan *routing table* untuk pengelolaan antrian:

Tabel 4.12: Routing table pengelolaan antrian - lanjutan

Group	Name	API Endpoint	HTTP Verb	Keterangan	Return Type
Antrian	<i>CREATE</i>	/kuota	<i>POST</i>	Menampilkan halaman menentukan kuota jumlah pasien berobat	<i>view</i>
	<i>READ</i>	/antrian	<i>GET</i>	menampilkan halaman antrian pasien berobat	<i>view</i>
	<i>UPDATE</i>	/list_daftar _berobat _online	<i>POST</i>	Menampilkan halaman <i>list</i> pasien yang telah daftar berobat <i>online</i> dan dimasukkan ke dalam antrian berobat	<i>view</i>
	<i>READ</i>	/list_daftar _berobat _offline	<i>GET</i>	Menampilkan halaman seluruh pasien untuk didaftarkan berobat <i>offline</i> oleh klinik dan dimasukkan ke dalam antrian berobat	<i>view</i>

16. Membuat *database* pengelolaan antrian



Gambar 4.22: *database* pengelolaan antrian

17. Pembuatan *mock up* administrasi keuangan

The mock-up shows a user interface for financial administration, specifically for bill verification and validation. The left sidebar menu includes:

- Selamat Pagi,
- Klinik Moist Care
- No. Reg: 13136177888899
- Dashboard
- Pengelolaan Akun
- Penentuan Kuota Berobat
- Pengelolaan dan Pendaftaran Antrian Berobat
- Data Medis Pasien
- Inventaris
- Layanan
- Tagihan
- Keluar Akun

The main area displays bill information and a list of services with their details and prices:

Item	Keterangan	Qty	Rp
Jahit luka	Keterangan: per 3 jahitan Harga : Rp. 50.000	6	100.000
Bersih luka	Keterangan: per 1x layanan Harga : Rp. 125.000	1	125.000
Pengecekan luka	Keterangan: per 1x layanan Harga : Rp. 50.000	1	50.000
Betadien	Tersedia: 20 Harga : Rp. 50.000	2	100.000

Total bill calculations:

- Sub - Total Rp 375.000
- Pajak(10%) Rp 37.500
- Total Rp 412.500

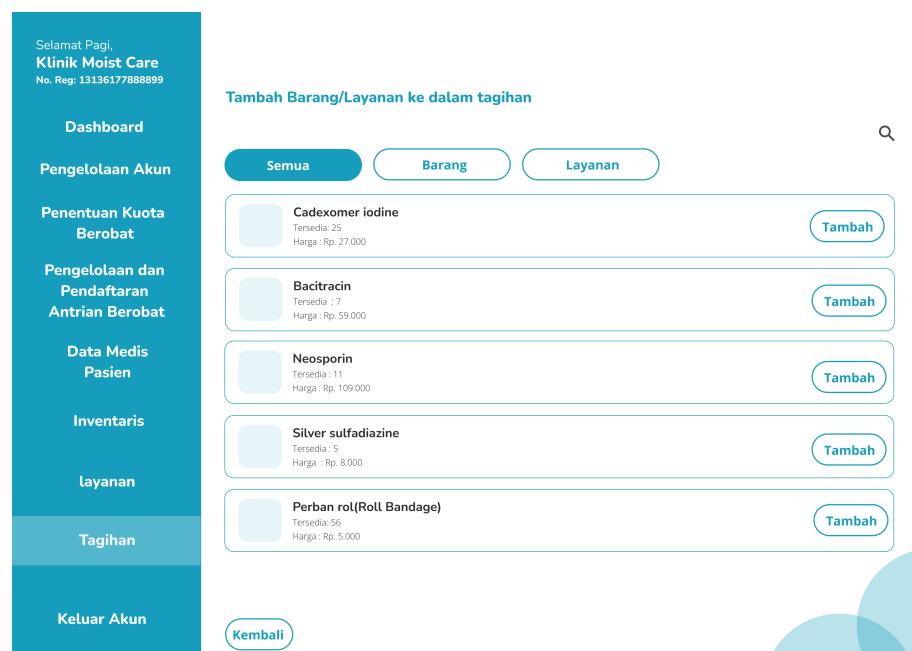
Buttons at the bottom include "Tambah Barang/Layanan", "Cetak dan Simpan", and a green checkmark icon.

Gambar 4.23: *Mock up* verifikasi dan validasi tagihan

Gambar 4.23 merupakan tampilan awal verifikasi dan validasi tagihan.

Terdapat *field Medical ID* dan Nomor BPJS untuk mencari nama pasien yang ingin membayar tagihan. Lalu ada *list* barang/layanan yang digunakan pasien selama berobat beserta detail kuantitas dan harga. Di bawah *list* juga terdapat keterangan harga sub-total, pajak, dan total harga dari semua barang/layanan yang digunakan pasien selama berobat. Selanjutnya terdapat cetak dan simpan untuk mencetak struk tagihan dan tombol tambah barang/layanan yang apabila admin klik akan mengarah pada **Gambar 4.21**.

Gambar 4.24 merupakan tampilan tambah barang/layanan ke dalam tagihan. Disini admin bisa mencari barang/layanan yang digunakan pasien dan menambahkannya ke dalam tagihan secara manual. Dan terdapat tombol kembali untuk kembali ke **Gambar 4.23**.



Gambar 4.24: Mock up tambah barang/layanan pada verifikasi dan validasi tagihan

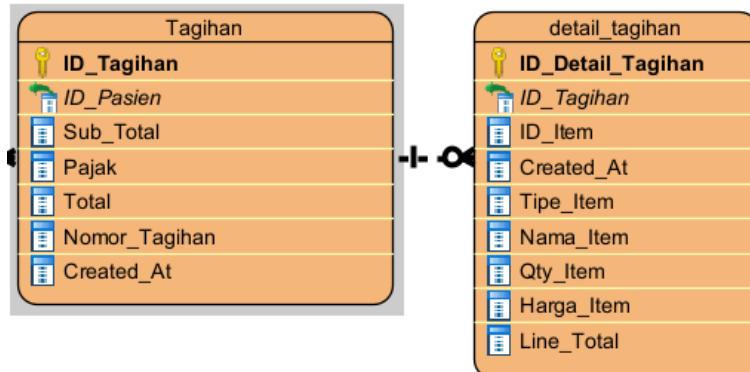
18. Desain *routing table* administrasi keuangan

Tabel 4.13 pada halaman selanjutnya merupakan *routing table* untuk administrasi keuangan:

Tabel 4.13: Routing table administrasi keuangan

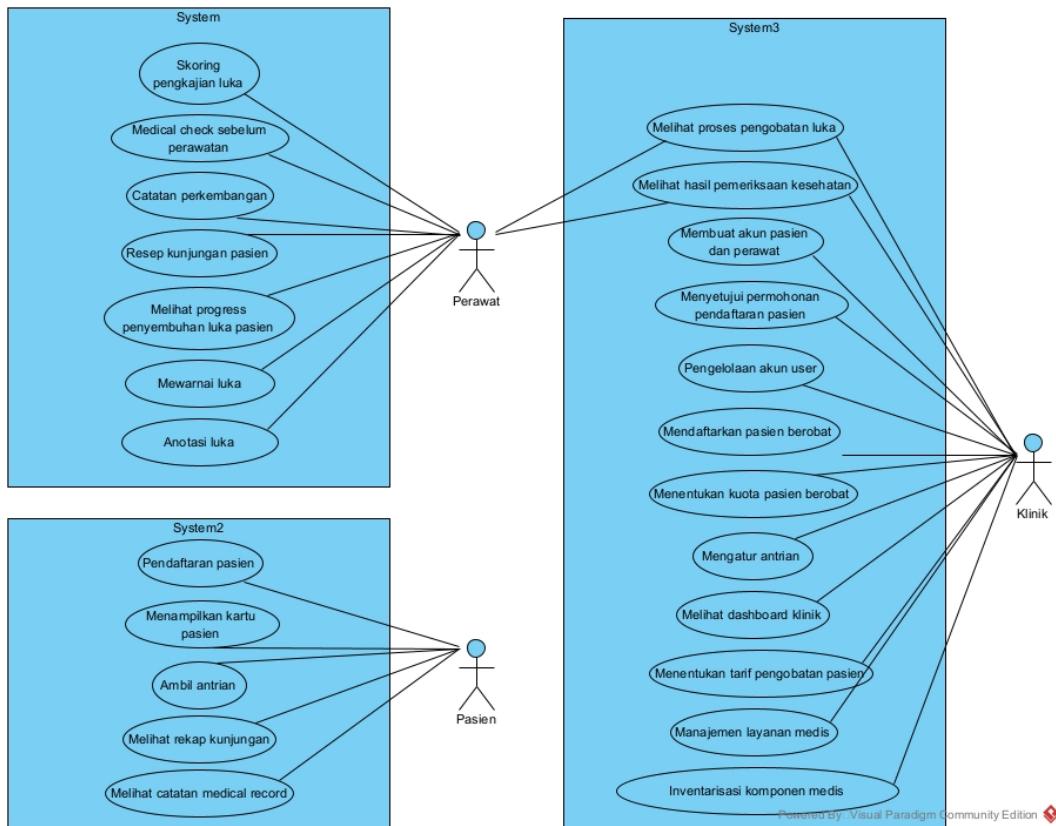
Group	Name	API Endpoint	HTTP Verb	Keterangan	Return Type
Tagihan	<i>CREATE</i>	/tagihan	<i>POST</i>	Menampilkan halaman membuat tagihan	<i>view</i>
	<i>READ</i>	/list_tagihan	<i>GET</i>	Menampilkan halaman <i>list</i> semua tagihan yang pernah dibuat	<i>view</i>

19. Membuat database administrasi keuangan

**Gambar 4.25:** Tabel tagihan dan tabel detail tagihan

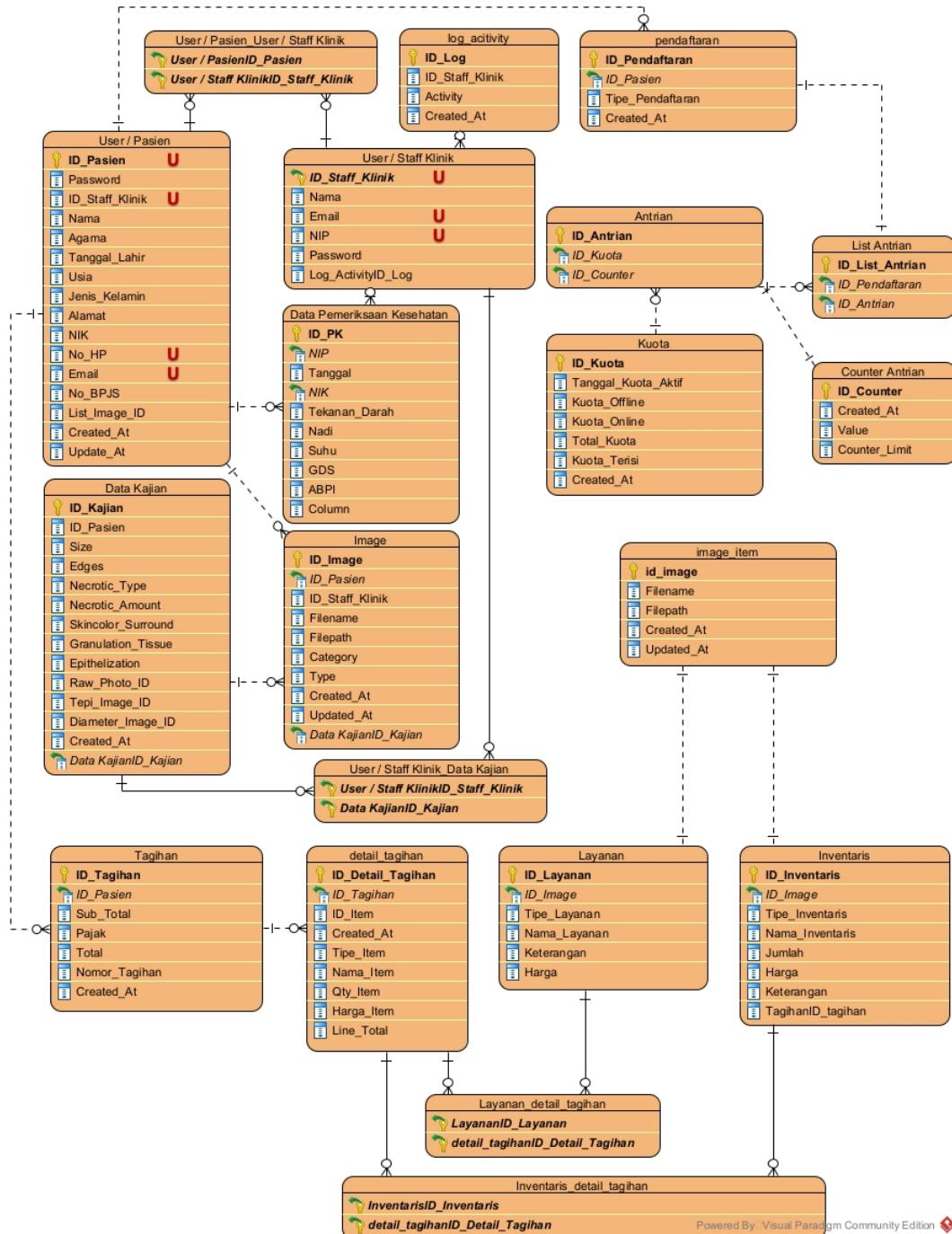
Gambar 4.25 merupakan database administrasi memiliki 2 tabel yang saling terhubung, yaitu tabel tagihan dan tabel detail tagihan. Tabel tagihan memuat beberapa atribut yaitu id_tagihan, id_pasien, sub_total, pajak, total, nomor_tagihan, dan created_at. Tabel detail tagihan memuat beberapa atribut yaitu id_detail_tagihan, id_tagihan, id_item, tipe_item, nama_item, qty_item, harga_item, line_total dan created_at. Data tersebut akan disimpan pada database MongoDB dan disimpan dalam format JSON.

20. Rancangan *use case* diagram



Gambar 4.26: *Use case* diagram

21. Rancangan desain database sistem



Gambar 4.27: Desain database sistem

4.1.2 Sprint-2

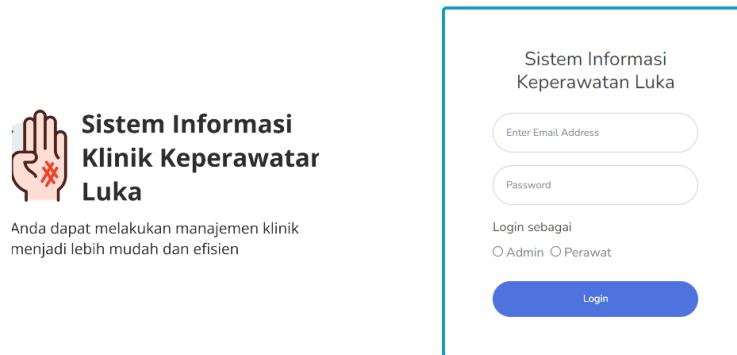
Tabel 4.14: Sprint-2 backlog

No	User Story	Task	Status
1	<i>Dashboard klinik</i>	1. implementasi <i>view login</i> perawat /admin 2. implementasi <i>web service login</i> perawat/admin klinik. 3. implementasi <i>dashboard</i> klinik.	selesai
2	Pembuatan akun pasien	1. implementasi <i>web service login</i> pasien 2. implementasi <i>view registrasi</i> pasien dari klinik. 3. implementasi <i>web service registrasi</i> pasien 4. implementasi <i>view list</i> permohonan akun pasien. 5. implementasi <i>web service list</i> permohonan akun pasien. 6. implementasi <i>view list</i> pasien terdaftar klinik. 7. implementasi <i>web service list</i> pasien terdaftar klinik.	selesai

1. Implementasi *view login* admin/perawat

Berikut merupakan implementasi *view login* admin/perawat menggunakan *Visual Studio Code*. *Code*-nya ada pada lampiran E dan menghasilkan

tampilan *view* seperti pada **Gambar 4.28**.



Gambar 4.28: Realisasi *view login* admin/perawat

2. Implementasi *web service login* perawat/admin klinik

Web service dibuat menggunakan framework *flask*. *flask* adalah *web service* berbasis *python*. URL *Routing* untuk melakukan *login* akun admin/perawat adalah `jft.web.id/woundapiv2/login_apps`.

Code untuk *web service login* admin/perawat ada pada lampiran F, metode yang digunakan adalah *POST*. Pada saat pemanggilan *web service*, terlebih dulu dilakukan pengecekan apakah sudah ada data yang sama berdasarkan *email*, *password* dan *role* atau belum, jika sudah ada maka akan dikembalikan ke halaman *view home* berisi halaman *dashboard* yang menandakan berhasil *login*, jika data tidak tersedia maka akan dikembalikan ke halaman *view login* yang menandakan bahwa *user* tidak ditemukan.

3. Implementasi *dashboard* klinik

Berikut merupakan implementasi *dashboard* klinik. *Code*-nya ada pada lampiran G dan menghasilkan tampilan *view* seperti pada **Gambar 4.29**.

Namun belum berjalan semestinya dan data yang ditampilkan hanya data statis dan masih bersifat *dummy*.

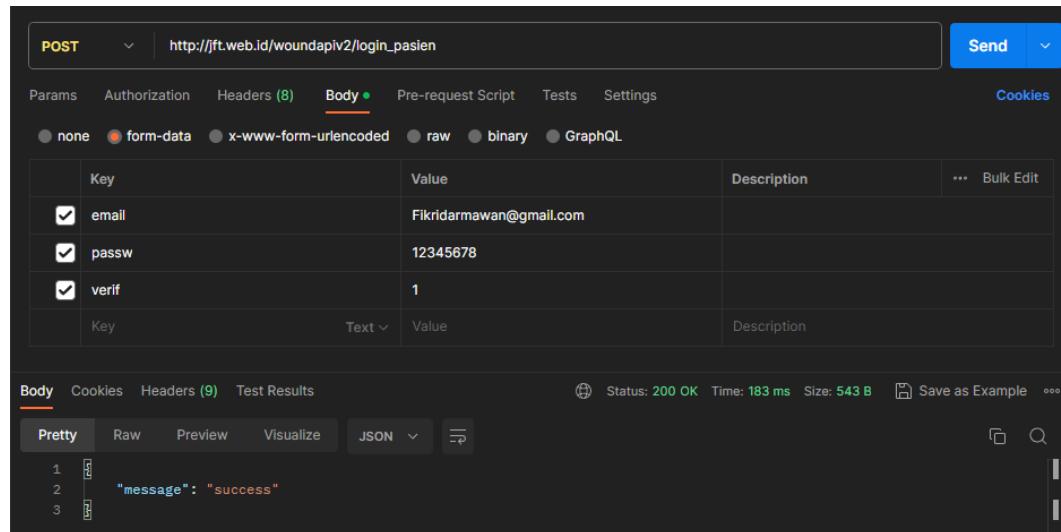


Gambar 4.29: Realisasi view dashboard klinik

4. Implementasi *web service* login pasien

URL *Routing* yang digunakan untuk melakukan *login* akun pasien adalah `jft.web.id/woundapiv2/login_pasien`.

Code untuk *web service* *login* pasien ada pada lampiran H, metode yang digunakan adalah *POST*. Pada saat pemanggilan *web service*, terlebih dulu dilakukan pengecekan apakah sudah ada data yang sama berdasarkan *email*, *password* dan status verifikasi pasien atau belum, jika sudah ada maka akan dikembalikan ke pesan *success* dengan status HTTP 200, jika data tidak sesuai maka akan dikembalikan ke pesan *failed* dengan status HTTP 400. **Gambar 4.30** pada halaman selanjutnya merupakan pengecekan *web service* *login* pasien yang sudah dapat digunakan dan dilakukan menggunakan POSTMAN.



Gambar 4.30: Pemanggilan *web service* login pasien

5. Implementasi *view* registrasi pasien dari klinik

Berikut merupakan implementasi *view* registrasi pasien dari klinik. *Code*-nya ada pada lampiran I dan menghasilkan tampilan *view* seperti pada **Gambar 4.31.**

The screenshot shows a web application for managing patients. The left sidebar has a blue background with white text and lists several menu items: Dashboard, Pengelolaan Akun, Pemeriksaan kesehatan, Proses Pengobatan, Inventaris, Layanan, and Keluar. The main content area is titled "Tambah Pasien Baru". It includes the following input fields:

- Email* (with placeholder "Enter Email Address")
- Password* (with placeholder "Enter Password")
- NIK* (with placeholder "Enter NIK")
- Nama* (with placeholder "Enter Name")
- Tanggal Lahir* (with placeholder "dd/mm/yyyy")
- Usia* (with placeholder "Enter Age")
- Jenis Kelamin* (radio buttons for Laki-laki and Perempuan)
- Agama* (radio buttons for Islam, Kristen, Hindu, Buddha, and Konghucu)
- Alamat* (with placeholder "Enter Address")
- Nomor Handphone* (with placeholder "Enter handphone number")

Gambar 4.31: Realisasi *view* registrasi pasien dari klinik

6. Implementasi *web service* registrasi pasien dari klinik

URL *Routing* yang digunakan untuk melakukan registrasi akun pasien dari klinik adalah `jft.web.id/woundapiv2/add_new_patient`.

Code untuk *web service* registrasi akun pasien ada pada lampiran J, metode yang digunakan adalah *POST*. Pada saat pemanggilan *web service*, admin terlebih dulu mengisi data email, *password*, NIK, nama, jenis kelamin, agama, tanggal lahir, usia, alamat, nomor HP lalu dilakukan pengecekan apakah sudah ada data yang sama atau belum, jika sudah ada maka akan dikembalikan ke halaman *view* registrasi pasien beserta pesan "gagal *input* pasien", jika data belum tersedia maka akan dikembalikan ke halaman *view* registrasi pasien beserta pesan "berhasil *input* pasien".

7. Implementasi *view list* permohonan akun pasien

Berikut merupakan implementasi *view list* permohonan akun pasien dari klinik.

Code-nya ada pada lampiran K dan menghasilkan tampilan *view* seperti pada **Gambar 4.32**.

NIK	Nama	Jenis Kelamin	Usia	Action
9928118291218	Ani Hidayah	perempuan	31	<button>lihat</button>
3152617228192	Kirana Mutiara	perempuan	31	<button>lihat</button>
315226261662	Yudo	laki-laki	28	<button>lihat</button>

Gambar 4.32: Realisasi *view list* permohonan akun pasien

8. Implementasi *web service list* permohonan akun pasien

URL *Routing* yang digunakan untuk *list* permohonan akun pasien untuk diverifikasi klinik adalah `jft.web.id/woundapiv2/list_request_new_patient`.

Code untuk *web service list* permohonan akun pasien untuk diverifikasi klinik ada pada lampiran L, metode yang digunakan adalah *GET*. Pada saat pemanggilan *web service* data berupa nama, jenis kelamin, usia dan NIK akan dipanggil untuk ditampilkan di tabel data *list* permohonan akun pasien.

Selanjutnya, *code* untuk verifikasi/terima pasien yang akan dikembalikan ke halaman *list* permohonan akun pasien, menampilkan pesan "berhasil verifikasi pasien", dan mengubah 1 pasien dari status belum terverifikasi klinik berubah menjadi terverifikasi klinik. *Code* bisa dilihat pada lampiran M.

Dan *code* untuk tolak verifikasi pasien yang akan dikembalikan ke halaman *list* permohonan akun pasien, menampilkan pesan "berhasil blokir pasien", dan mengubah 1 pasien dari status belum terverifikasi klinik berubah menjadi terblokir. *Code* bisa dilihat pada lampiran N.

9. Implementasi *view list* pasien terdaftar klinik

NIK	Nama	Jenis Kelamin	Usia	action
9981122811829521	Annisa	perempuan	28	<button>lihat</button>
998213446778	Afifah	perempuan	13	<button>lihat</button>
336298665321	Fikri Darmawan	laki-laki	19	<button>lihat</button>

Gambar 4.33: Realisasi *view list* pasien terdaftar klinik

Gambar 4.33 merupakan *view list* pasien terdaftar klinik. *Code*-nya ada pada lampiran O.

10. Implementasi *web service list* pasien terdaftar klinik

URL *Routing* yang digunakan untuk *list* pasien untuk terdaftar/terverifikasi klinik adalah `jft.web.id/woundapiv2/list_patient`.

Code untuk *web service list* pasien untuk terdaftar/terverifikasi klinik ada pada lampiran P, metode yang digunakan adalah *GET*. Pada saat pemanggilan *web service* data berupa nama, jenis kelamin, usia dan NIK akan dipanggil untuk ditampilkan di tabel data *list* pasien untuk terdaftar/terverifikasi klinik.

4.1.3 Sprint-3

Tabel 4.15: Sprint-3 backlog

No	User Story	Task	Status
1	Pemeriksaan kesehatan	1. implementasi <i>view</i> hasil pemeriksaan kesehatan. 2. implementasi <i>web service</i> hasil pemeriksaan kesehatan.	selesai
2	Proses pengobatan	1. implementasi <i>view</i> inventaris. 2. implementasi <i>web service</i> inventaris. 3. implementasi <i>view</i> layanan. 4. implementasi <i>web service</i> layanan. 5. implementasi <i>view</i> hasil pengkajian luka. 6. implementasi <i>web service</i> hasil pengkajian luka. 7. implementasi <i>view database</i> foto luka yang dikategorisasikan per perawat. 8. implementasi <i>web service database</i> foto luka yang dikategorisasikan per perawat.	selesai tidak selesai
3	Pendaftaran Pasien Berobat	1. implementasi <i>view</i> pendaftaran berobat. 2. implementasi <i>web service</i> pendaftaran berobat.	tidak selesai

1. Implementasi *view* hasil pemeriksaan kesehatan

Berikut merupakan implementasi *view* detail hasil pemeriksaan kesehatan

pasien sesuai tanggal yang admin pilih. *Code*-nya ada pada lampiran Q dan menghasilkan tampilan *view* seperti pada **Gambar 4.34**.

	Tanggal Pemeriksaan	Nadi
Tekanan Darah	140/95	95
Suhu Tubuh	34	Gula Darah Sewaktu
ABPI	119	99

Gambar 4.34: Realisasi *view* detail hasil pemeriksaan kesehatan seorang pasien berdasarkan tanggal pemeriksaan

2. Implementasi *web service* hasil pemeriksaan kesehatan

URL *Routing* yang digunakan untuk melakukan tambah hasil pemeriksaan kesehatan adalah `jft.web.id/woundapiv2/add_data_pemeriksaan_kesehatan`.

Code untuk *web service* tambah hasil pemeriksaan kesehatan ada pada lampiran R, metode yang digunakan adalah *POST*. Pada saat pemanggilan *web service*, perawat terlebih dulu mengisi tanggal, *username* atau NIP, NIK pasien, tekanan darah, nadi, suhu, gula darah sewaktu, dan ABPI lalu akan dikembalikan ke pesan berhasil input data pemeriksaan kesehatan dengan status HTTP 200, jika data tidak sesuai maka akan dikembalikan ke pesan gagal input data pemeriksaan kesehatan dengan status HTTP 500.

Dan *code* *web service* untuk mendapatkan detail data hasil pemeriksaan kesehatan pasien berdasar tanggal pemeriksaan, yang kemudian akan ditampilkan pada *view* detail hasil pemeriksaan kesehatan pasien. *Code*-nya

ada pada lampiran S.

3. Implementasi *view* inventaris

Berikut merupakan implementasi *view list* inventaris klinik. *Code*-nya ada pada lampiran T dan menghasilkan tampilan *view* seperti pada **Gambar 4.35**.

ID Inventaris	Nama Inventaris	Tipe Inventaris	Harga	Action
64d8169dc9c5f110be0a9ec9	Hansaplast	balutan	2000	<button>lihat</button>

[Tambah Inventaris](#)

Gambar 4.35: Realisasi *view* semua inventaris klinik

Gambar 4.36: Realisasi *view* tambah inventaris klinik

Jika admin menekan tombol aksi tambah inventaris maka akan diarahkan ke *view* tambah inventaris. *Code* -nya ada pada lampiran U dan menghasilkan

tampilan *view* seperti pada **Gambar 4.36.**

4. Implementasi *web service* inventaris

URL *Routing* yang digunakan untuk inventaris klinik adalah jft.web.id/woundapiv2/add_inventaris.

Code untuk *web service* tambah inventaris ada pada lampiran V, metode yang digunakan adalah *POST*. admin akan diminta untuk mengisi data yang dibutuhkan untuk menambah inventaris, akan menampilkan pesan "berhasil input inventaris", dan menambah 1 inventaris ke dalam *database* klinik. Jika gagal akan menampilkan pesan "gagal input inventaris".

5. Implementasi *view* layanan

Berikut merupakan implementasi *view list* layanan klinik. *Code*-nya ada pada lampiran W dan menghasilkan tampilan *view* seperti pada **Gambar 4.37.**

ID Layanan	Nama Layanan	Harga	Action
64d8173fe49aa807cd71dc7a	Bersih luka	3000	lihat

Gambar 4.37: Realisasi *view* semua layanan klinik

Jika admin menekan tombol aksi tambah layanan maka akan diarahkan ke *view* tambah layanan. Berikut merupakan *code* ada pada lampiran X dan menghasilkan tampilan *view* seperti pada **Gambar 4.38.**

The screenshot shows a web-based application interface for managing a clinic. On the left, a vertical sidebar has a blue header 'KLINIK MOIST CARE' and a list of navigation items: Dashboard, Pengelolaan Akun, Pemeriksaan kesehatan, Proses Pengobatan, Inventaris, Layanan (which is currently selected and highlighted in blue), and Keluar. The main content area has a white background and features a form titled 'Tambah Layanan Baru' (Add New Service). The form includes three input fields: 'Nama Layanan*' (Service Name*) with a placeholder 'Masukkan Nama Layanan...', 'Keterangan*' (Description*) with a placeholder 'Masukkan Keterangan...', and 'Harga*' (Price*) with a placeholder 'Masukkan Harga...'. Below these fields are two blue rounded rectangular buttons: 'Kembali' (Back) on the left and 'Submit' on the right.

Gambar 4.38: Realisasi *view* tambah layanan klinik

6. Implementasi *web service* layanan

URL *Routing* yang digunakan untuk layanan klinik adalah `jft.web.id/woundapiv2/add_layanan`.

Code untuk *web service* tambah layanan klinik ada pada lampiran Y, metode yang digunakan adalah *POST*. admin akan diminta untuk mengisi data yang dibutuhkan untuk menambah layanan, menampilkan pesan "berhasil input layanan", dan menambah 1 layanan ke dalam *database* klinik. Jika gagal akan menampilkan pesan "gagal input layanan".

4.1.4 *Sprint-4*

Sprint-4 tidak terencanakan.

4.1.5 Laporan akhir pengembangan sistem

Pada Penelitian ini, fitur pada sistem yang selesai peneliti buat adalah:

1. pembuatan akun pasien

2. *dashboard* klinik
3. pemeriksaan kesehatan
4. sebagian proses pengobatan luka (*view* dan *web service* inventaris dan layanan)

Dan fitur pada sistem yang tidak selesai peneliti buat adalah:

1. proses pengobatan luka
2. pendaftaran pasien berobat
3. pengelolaan antrian
4. administrasi keuangan
5. sebagian proses pengobatan luka (*view* dan *web service* hasil pengkajian luka dan *database* foto luka yang dikategorisasikan berdasarkan perawat)

Peneliti tidak dapat menuntaskan pengembangan sistem informasi keperawatan luka karena beberapa hal, yaitu:

1. *Lack of skill*

Pada saat pengembangan sistem berlangsung, peneliti juga mempelajari teknologi-teknologi yang diperlukan untuk pengembangan sistem karena peneliti tidak menguasai teknologi yang akan dipakai. Sehingga peneliti membutuhkan waktu yang lebih lama untuk pengembangannya, dimana deskripsi ini berkaitan dengan poin selanjutnya.

2. Waktu yang diberikan untuk merancang sistem yang kurang

Seperti yang sudah dijelaskan pada poin sebelumnya, dikarenakan peneliti membutuhkan waktu yang lebih lama untuk pengembangannya karena

disamping mengembangkan, peneliti juga harus belajar kembali teknologi yang belum dikuasai. sehingga waktu yang direncanakan dalam mengerjakan sistem dirasa sangat kurang untuk peneliti bisa menyelesaikan sistem yang ingin dibuat.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil implementasi sistem informasi keperawatan luka, maka dapat diambil kesimpulan sebagai berikut:

1. Terciptanya sistem informasi keperawatan luka yang sudah mengimplementasikan sebagian fitur-fitur pada *product backlog*. Adapun perancangannya dilakukan dengan metode *scrum* dengan tahapan penyusunan *product backlog*, *sprint backlog* dan terbagi menjadi empat *sprint*.
2. Fitur pada sistem yang selesai dibuat diantaranya adalah pembuatan akun pasien, *dashboard* klinik, pemeriksaan kesehatan dan sebagian proses pengobatan luka (*view* dan *web service* inventaris dan layanan).
3. Fitur pada sistem yang tidak selesai dibuat diantaranya adalah sebagian proses pengobatan luka (*view* dan *web service* hasil pengkajian luka dan *database* foto luka yang dikategorisasikan berdasarkan perawat), pendaftaran pasien berobat, pengelolaan antrian, dan administrasi keuangan.
4. Terimplementasikannya *web service* yang berfungsi sebagai *back-end* sistem informasi keperawatan luka.
5. Sistem informasi keperawatan luka dikembangkan menggunakan bahasa *python* dengan bantuan *framework flask*.

5.2 Saran

Adapun beberapa saran untuk penelitian selanjutnya adalah sistem informasi keperawatan luka dapat dilanjutkan pengembangannya untuk fitur-fitur yang belum terselesaikan atau menambahkan fitur-fitur lain yang dibutuhkan pada masa yang akan datang.

DAFTAR PUSTAKA

- A. Leitch, R. and Davis, K. R. (2001). *Sistem Informasi*. PT. Prenhallindo.
- Ardiansyah and Effiyaldi (2021). Analisis dan Perancangan Sistem Informasi Manajemen Rumah Sakit Berbasis Website Pada Rumah Sakit Umum Kambang Kota Jambi. *Jurnal Ilmiah Sistem Informasi, Teknologi Informasi dan Sistem Komputer*, 6(1):188–197.
- Aryani, R. (2016). Accelerating Wound Healing Process By Using Moist Dressing.
- Aryani, R., Yusro, M., Suryana, M. E., and Firmansyah, I. (2018). Rancang Bangun Aplikasi Mobile Android Sebagai Alat Deteksi Warna Dasar Luka Dalam Membantu Proses Pengkajian Luka Kronis Dengan Nekrosis.
- Biswas, T., Fauzi, M. F. A., Abas, F. S., and Nair, H. K. (2018). Superpixel Classification with Color and Texture Features for Automated Wound Area Segmentation. *2018 IEEE 16th Student Conference on Research and Development, SCOReD 2018*, pages 1–6.
- Carminah, I., Suheryadi, A., and Puspaningrum, A. (2021). Aplikasi Monitoring Perawatan Luka Aplikasi Monitoring Perawatan Luka Diabetes Mellitus Berbasis Website. *Seminar Nasional Teknologi Terapan (SEMITERA)*, pages 130–138.
- Fatima, S. (2013). Perancangan Sistem Informasi Penjualan Mebel Online pada UD. Melindo Jaya. *Kisaran: AMIK Royal Kisaran*.
- FitzGerald, J., Stallings, W. D., and Fitzgerald, A. F. (1981). *Fundamentals of systems analysis*. John Wiley & Sons, Inc.
- Haviluddin, H. (2011). Memahami Penggunaan UML (Unified Modelling Language). 6.
- Khairunnisa, A. (2021). Pengaruh Penggunaan Color Model Lab Dalam Kalibrasi Warna Luka Menggunakan Metode Segmentasi K-Means dan Mean Shift.
- Ozierańska, A., Skomra, A., Kuchta, D., and Rola, P. (2016). The critical factors of Scrum implementation in IT project– the case study. *Journal of Economics and Management*, 25(3):79–96.
- Rahmadati, S. (2023). Rancang Bangun Aplikasi dan Web Service Pengkajian Luka Kronis Khususnya Modul Pengolahan Citra Berbasis Android.
- Rizki, M. (2022). Deteksi Keliling Luka Kronis Menggunakan Active Contour (Snake) dan Active Contour yang Ditambahkan Interpolasi.

- Safitri, N. A. N., Purwanti, L. E., and Andayani, S. (2022). Hubungan Perilaku Perawatan Kaki Dengan Kualitas Hidup Pasien Diabetes Melitus Di RSU Muhammadiyah Dan Klinik Rulia Medika Ponorogo. *Health Sciences Journal*, 6(1):67–74.
- Suendri (2019). Implementasi Diagram UML (Unified Modelling Language) Pada Perancangan Sistem Informasi Remunerasi Dosen Dengan Database Oracle (Studi Kasus: UIN Sumatera Utara Medan). *Jurnal Ilmu Komputer dan Informatika*, 3(1):1–9.
- Susanto, G. and Sukadi (2021). Sistem Informasi Rekam Medis Pada Rumah Sakit Umum Daerah (RSUD) Pacitan Berbasis Web Base. *Journal Speed Sentra Penelitian Engineering dan Edukasi*, 3(4):18–24.
- Vogt, N. T., Koller, F. J., Santos, P. N. D., Lenhani, B. E., Guimarães, P. R. B., and Kalinke, L. P. (2020). Quality of life assessment in chronic wound patients using the Wound-QoL and FLQA-Wk instruments. *Investigacion y Educacion en Enfermeria*, 38(3):1–12.
- Whitten, J. L., Bentley, L. D., and Dittman, K. C. (2004). *Metode desain dan analisis sistem*. Andi Offset, Yogyakarta.
- Widodo, P. P. and Prabowo, H. (2011). *Menggunakan UML*. Informatika Bandung.

LAMPIRAN A

Dokumentasi *Screenshot* Presentasi Bersama *Owner Klinik Moist Care*



Gambar 1.1: Dokumentasi presentasi bersama Ibu Irma Puspita Arisanti 1



Gambar 1.2: Dokumentasi presentasi bersama Ibu Irma Puspita Arisanti 2

LAMPIRAN B

Transkrip Paparan Presentasi Bersama Pemilik Klinik *Moist Care*

1

- Narasumber : Ibu Irma Puspita Arisanti
- Narasumber : Terkait penelitiannya kalau untuk saya bisa kasih masukan ini kan pak eka kan tadi gambarannya di PPT itu masih yang terlalu umum ya, saya gak lihat isinya, kalau mau saya lihat isinya begitu jadi nanti saya tahu ini ke sini, yang kemarin saya kasih gambaran kan kaya misalnya intinya ini buat siapa aplikasinya, buat pasien atau buat perawatnya atau buat instansi dimana perawat itu bekerja. Kalau misalnya untuk pasien berarti kan nanti kita informasinya seperti yang kemarin saya bilang itu, pasien itu paling enggak dapet informasi terkait dengan jadwal, cara dia bikin janji, kemudian melihat jadwal, melihat perkembangan lukanya, lalu kemudian lihat lukanya. Lihat perkembangan itu kan ini kemaren pakai versi insan ya, berarti kan ada skoring, nah skoring itu supaya melihat dia baik atau enggak kan ada grafik, kalau bisa tambahan ada grafiknya, itu yang di lihat pasien itu saja empat poin
- Scrum Master* : Kalau pengembangan sistem itu di bagi dua, pertama itu adalah daftar tertulis dari kebutuhan yang mau didevelop, kedua itu setelah dapat tertulis, biasanya kan ini tanda tangan, setelah butuh tanda tangan nanti ad kami ada dalam bentuk rancangan tatap muka. tapi kan kalau misalkan mau ada teks nanti kan gak kebayang kan, jadi nanti gini saja deh kita buat semua user interface, saya tampilkan semuanya
- Narasumber : Ya kan ini maksudnya apa ada tambahan atau ada yang mau dikurangi, nah kalau saya mau melihat yang di tambah dan apa yang di kurangi kalau dari PPT itu kemaren gak cukup, saya bingung itu loh, ya terserahlah mekanismenya bagaimana, yang terpenting tergambar alurnya, proses yang sudah terjadi dari awal sampai akhir, kalau itu kan satu slide sendiri, bicara tentang itu dan terlalu kecil-kecil dan saya tidak cukup memahami dari poin-poin itu. Terus kemudian bisa gak sih si perawatnya melihat rekapan kunjungan datanya itu misalnya muncul satu nama

kemudian dari satu nama itu ada muncul kontrol tanggal sekian tanggal sekian sampai sembuh, itu bisa apa enggak? Harusnya sih bisa, kebayang ya? Jadi kalau misalnya data pasien itu di klik pasien anu itu misalnya di klik itu rekapan kunjungannya dari awal sampai ke medical history perpasien. Jadi gak sekedar kita upload data, abis itu kita bingung ini ngeliat dari kunjungan pertama sampai terakhir itu gak bisa kita baca, ya kan itu history kan perawatan. Jadi satu pasien itu muncul ketika kita klik nama pasien muncul deh historynya mulai dari kunjungan satu sampai kunjungan sembuh, kalau perlu sampai ada pemeriksaan labnya, lampirannya di situ ada, termasuk foto lukanya juga ada di situ

Scrum Master

: Kalau lab saya belum kebayang ya soalnya selama ini belum ada.

Narasumber

: Kalau begitu gak usah lah kalau belum mah, kalau bisa, berarti Cuma foto saja

Scrum Master

: Yang penting kami butuh data sih

Narasumber

: Kalau data kita gak bisa kasih mas, karena kan itu kan ini bersifat privasi ya

Scrum Master

: Maksudnya bukan data bu, melainkan proses alur klinik

Narasumber

: Data proses ya tadi yang saya sampaikan, data proses. Itu paling tambahan tambahannya kalau mau memasukkan lagi nanti bisa dilihat dari apa yang sudah ada yang perlu nanti bisa di tambahkan atau di kurangi

LAMPIRAN C

Transkrip Paparan Presentasi Bersama Pemilik Klinik *Moist Care*

2

- Narasumber : Ibu Irma Puspita Arisanti
- Narasumber : untuk penelitian saya oke-oke saja, nah ini untuk yang pra-penelitian kalau saya pikir prosedurnya sama kaya penelitian saja pak, kaya misalkan kaya alur penelitian kan membicarakan misalnya harga, validasi, prototipe, dua kali seminggu, dua pekan misalnya, itu kan masuk dalam model penelitian nanti kan, nanti mungkin yang ke kita penelitian itu surat penelitian, formal, kemudian proposal penelitiannya segala macam dilampirkan termasuk uji etic, karena kan kaitannya kita ngambil data pasien ya kan, nanti eticnya bagaimana, biasanya dari kampus kan, bapak konsultasi dulu ya karena kan terkait data-data pasien kan, begitu saja mungkin untuk yang penelitian prosedurnya, sampai ke alur kan, nanti kan di alur kan jelas tu step by step dari pihak peneliti, kemudian dari pihak kita apa. Paling itu sih ya pak sigit
- Scrum master* : jadi nanti kita minta izin ke bu irma untuk mengirim online ini nanti kita butuh validasi untuk pengembangan itu apakah cukup valuable di klinik atau enggak, mumpung kita masih awal development. Kalau misalkan sudah jalan itu nanti jadi sulit lagi
- Narasumber : jadi kaya kalau bahasanya modul ya, main modul, istilahnya modul kan?
- Scrum master* : saya kirim dokumen saja deh ke bu irma atau nanti tolong dipelajari dulu lalu kalau butuh presentasi, nanti rekan-rekan paparkan lagi secara online gak papa kok
- Ibu Irma : ini untuk yang penelitian
- Scrum master* : penelitian mau kita belokin ke bisnis bu
- Ibu Irma : yang bahasan pertama ini yang penelitian yang di kirim ke saya kan?
- Scrum master* : tapi fiturenya yang sudah berbasis klinik, jadi klinik butuh apa itu harus di ceklist bu, minimal kerjanya ceklist saja deh

LAMPIRAN D

Surat Pernyataan Kesediaan Kerjasama Dari Mitra

SURAT PERNYATAAN KESEDIAAN KERJASAMA DARI MITRA

Yang bertanda tangan di bawah ini:

Nama	:	Klinik Moist
Pimpinan Mitra	:	Irma Puspita Arisanty
Bidang Kegiatan	:	Kesehatan
Alamat	:	BCC green AVENUE blok DD 2 no 17 kel. Sukadamai kec. Tanah Sereal kota Bogor

Dengan ini menyatakan Bersedia untuk Bekerjasama dengan Pelaksana Kegiatan PKM- Penerapan Iptek/Pengabdian Kepada Masyarakat) dengan judul:

“Integrasi Pelayanan Kesehatan dan Ketahanan Nasional: Otomasi Sistem Informasi Klinik dan Keperawatan Pasien dengan Fitur Pengkajian Luka Kronis”

Nama Ketua Tim	:	Muhammad Hafiz Hisbullah
Nomor Induk Mahasiswa	:	1313619019
Program Studi	:	Ilmu Komputer
Nama Dosen Pendamping	:	Muhammad Eka Suryana M.kom
Perguruan Tinggi	:	Universitas Negeri jakarta

Guna mengembangkan dan/atau menerapkan iptek pada tempat kami.

Bersama ini pula kami nyatakan dengan sebenarnya bahwa diantara pihak Mitra dan Pelaksana Program tidak terdapat ikatan kekeluargaan dan/atau ikatan usaha dalam wujud apapun juga.

Demikian Surat Pernyataan ini dibuat dengan penuh kesadaran dan tanggungjawab tanpa ada unsur pemaksaan di dalam pembuatannya untuk dapat digunakan sebagaimana mestinya.

Jakarta, 18 Juli 2023

Yang menyatakan,



(Irma Puspita Arisanty)

Gambar 4.1: Surat Pernyataan Kesediaan Kerjasama Dari Mitra

LAMPIRAN E

Code Untuk View Login Admin dan Perawat

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,
initial-scale=1, shrink-to-fit=no">
<meta name="description" content="">
<meta name="author" content="">
<title>Sistem Informasi Keperawatan Luka –
Login</title>
<link
href="https://fonts.googleapis.com/css?family=Nunito:2
0,200i,300,300i,400,400i,600,600i,700,700i,800,800i,90
,900i"
rel="stylesheet">
<!-- Custom styles for this template-->
<link href="..//static/css/sb-admin-2.min.css"
rel="stylesheet">
</head>
<body>
<div class="container">
<!-- Outer Row -->
```

```
<div class="row justify-content-center">
<div class="col-xl-10 col-lg-12 col-md-9">
<div class="border-0 my-7">
<div class="card-body p-0">
<!-- Nested Row within Card Body -->
<div class="row">
<div class="card col-lg-5 d-none d-lg-block border-0
bg-login-image"></div>
<div class="card col-lg-2 border-0"></div>
<div class="card col-lg-5 border-1">
<div class="p-5">
<div class="text-center">
<h1 class="h4 text-gray-900 mb-4">Sistem Informasi
Keperawatan Luka</h1>
</div>
{%
  with messages = get_flashed_messages() %}
{%
  if messages %}
{%
    for message in messages %}
<p>{{ message }}</p>
{%
    endfor %}
{%
    endif %}
{%
    endwith %}
<form class="user" method="POST" action="/login">
<div class="form-group">
<input type="email" class="form-control
form-control-user" name="email"
id="username" placeholder="Enter Email Address" required>
</div>
```

```
<div class="form-group">
<input type="password" class="form-control
form-control-user" name="passw"
id="passw" placeholder="Password" required>
</div>

<div class="form-group">
<label class="text-gray-900" for="role">Login
sebagai</label>
<br>
<input type="radio" name="role" id="admin"
value="admin" required>
<label for="html">Admin</label>&nbsp;
<input type="radio" name="role" id="perawat"
value="perawat" required>
<label for="html">Perawat</label>
</div>

<button type="submit" class="btn btn-primary btn-user
btn-block">
Login
</button>
</form>
</div>
</div>
</div>
</body>
</html>
```

LAMPIRAN F

Code Untuk Web Service Login Admin dan Perawat

```
@bp.route('/login', methods=['POST'])

def login():
    try:
        data = {
            "email" : request.form['email'],
            "passw" : request.form['passw'],
            "role" : request.form['role']}
        a = db.get_user(data)
        if a == None:
            print("User tidak ditemukan")
            flash("User tidak ditemukan")
            return redirect(url_for('login'))
        else:
            session['user_info'] = {
                "email" : data["email"],
                "role" : data["role"]}
            print("Berhasil Login")
            flash("Berhasil Login")
            return redirect(url_for('home'))
        except Exception as ex:
            print("Gagal login")
            flash("Gagal login")
            return redirect(url_for('login'))
```

LAMPIRAN G

Code Untuk Dashboard Klinik

```
{% extends "layout/Layout.html" %}

{% block title %} Dashboard {% endblock %}

{% block content %}

<!-- Content Wrapper -->

<div id="content-wrapper" class="d-flex flex-column
my-0">

<!-- Begin Page Content -->

<div class="container">

<!-- Outer Row -->

<div class="row justify-content-center">

<div class="card-body">

<div class="text-left">

<h1 class="h4 text-gray-900 my-3">Dashboard Klinik</h1>

</div>

{% with messages = get_flashed_messages() %}

{% if messages %}

{% for message in messages %}

<p>{{ message }}</p>

{% endfor %}

{% endif %}

{% endwith %}

<!-- Nested Row within Card Body -->

<div class="row">

<div class="col-lg-12">
```

```
<div class="p-5">  
    <!-- Content Row -->  
    <div class="row">  
        <div class="col-xl-10 col-md-6 mb-4">  
            <a href="#" class="d-none d-sm-inline-block btn btn-sm  
                btn-primary shadow-sm"><i  
                class="fas fa-sm text-white-50"></i> Kinerja  
                Perawat</a>  
            <a href="#" class="d-none d-sm-inline-block btn btn-sm  
                btn-primary shadow-sm"><i  
                class="fas fa-sm text-white-50"></i> Cost Perawatan  
                Pasien</a>  
            <a href="#" class="d-none d-sm-inline-block btn btn-sm  
                btn-primary shadow-sm"><i  
                class="fas fa-sm text-white-50"></i> Income Masuk</a>  
            <a href="#" class="d-none d-sm-inline-block btn btn-sm  
                btn-primary shadow-sm"><i  
                class="fas fa-sm text-white-50"></i> Balance  
                Keuangan</a>  
        </div>  
    </div>  
    <div class="row">  
        <!-- Area Chart -->  
        <div class="col-xl-8 col-lg-7">  
            <div class="card shadow mb-4">  
                <!-- Card Header - Dropdown -->  
                <div  
                    class="card-header py-3 d-flex flex-row"
```

```
align-items-center justify-content-between">

<h6 class="m-0 font-weight-bold text-primary">Cost
Perawatan Pasien</h6>

<div class="dropdown no-arrow">
<a class="dropdown-toggle" href="#" role="button"
id="dropdownMenuLink"
data-toggle="dropdown" aria-haspopup="true"
aria-expanded="false">
</a>
</div>
</div>

<!-- Card Body -->

<div class="card-body">
<div class="chart-area">
<canvas id="myAreaChart"></canvas>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
<!-- /.container-fluid -->
</div>

<!-- End of Main Content -->
</div>
</div>

{ % endblock content %}
```

LAMPIRAN H

Code Untuk Web Service Login Pasien

```
#login pasien

@bp.route('/login_pasien', methods=['POST'])

def login_pasien():

    try:

        data = {

            "email" : request.form['email'],

            "password" : request.form['passw'],

            "verif" : '1'

        }

        a = db.get_pasien_login(data)

        if a == None:

            print("Pasien tidak ditemukan")

            return Response(response = json.dumps({"message" :

                "failed"}), mimetype="application/json", status=400)

        else:

            print("Berhasil login")

            return Response(response = json.dumps({"message" :

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

    except Exception as ex:

        print (ex)

        return Response(response = json.dumps({"message" :

                "exe"}), mimetype="application/json", status=500)
```

LAMPIRAN I

Code Untuk View Registrasi Pasien Dari Klinik

```
{% extends "layout/Layout.html" %}

{% block title %} Tambah Pasien Baru {% endblock %}

{% block content %}

<!-- Content Wrapper -->
<div id="content-wrapper" class="d-flex flex-column my-0">

<!-- Begin Page Content -->
<div class="container">

<!-- Outer Row -->
<div class="row justify-content-center">

<div class="card-body">
<div class="text-left">
<h1 class="h4 text-gray-900 my-3">Tambah Pasien
Baru</h1>
</div>
{% with messages = get_flashed_messages() %}
{% if messages %}
{% for message in messages %}
<p>{{ message }}</p>
```

```
{% endfor %}

{% endif %}

{% endwith %}

<!-- Nested Row within Card Body -->



<form class="user" method="POST" action="/pasien">
<div class="form-group row">
<div class="col-sm-6">
<label class="text-gray-900" for="email">Email*</label>
<input type="email" class="form-control form-control-user" name="email" id="email" aria-describedby="emailHelp" placeholder="Enter Email Address" required>
</div>
<div class="col-sm-6">
<label class="text-gray-900" for="passw">Password*</label>
<input type="password" class="form-control form-control-user" name="passw" minlength="8" maxlength="16" id="passw" placeholder="Enter Password" required>
</div>
</div>
<div class="form-group row">
<div class="col-sm-6">
<label class="text-gray-900" for="nik">NIK*</label>


```

```
<input type="number" class="form-control  
form-control-user" minlength="16" name="nik"  
id="nik" placeholder="Enter NIK" required>  
</div>  
    <label class="text-gray-900" for="nama">Nama*</label>  
    <input type="text" class="form-control  
form-control-user" name="nama"  
id="nama" placeholder="Enter Name" required>  
</div>  
</div>  
    <div class="col-sm-6">  
        <label class="text-gray-900" for="born_date">Tanggal  
Lahir*</label>  
        <input type="date" class="form-control  
form-control-user" name="born_date"  
id="born_date" required>  
</div>  
    <div class="col-sm-6">  
        <label class="text-gray-900" for="usia">Usia*</label>  
        <input type="number" class="form-control  
form-control-user" name="usia"  
id="usia" placeholder="Enter Age" required>  
</div>  
</div>  
    <div class="col-sm-6">
```

```
<label class="text-gray-900" for="kelamin">Jenis  
Kelamin*</label>  
  
<br>  
  
<input type="radio" name="kelamin" id="laki-laki"  
value="laki-laki" required>  
  
<label for="html">Laki-laki</label><br>  
  
<input type="radio" name="kelamin" id="perempuan"  
value="perempuan" required>  
  
<label for="html">Perempuan</label><br>  
  
</div>  
  
<div class="col-sm-6">  
  
<label class="text-gray-900"  
for="kelamin">Agama*</label>  
  
<br>  
  
<input type="radio" name="agama" id="islam"  
value="islam" required>  
  
<label for="html">Islam</label><br>  
  
<input type="radio" name="agama" id="kristen"  
value="kristen" required>  
  
<label for="html">Kristen</label><br>  
  
<input type="radio" name="agama" id="hindu"  
value="hindu" required>  
  
<label for="html">Hindu</label><br>  
  
<input type="radio" name="agama" id="buddha"  
value="buddha" required>  
  
<label for="html">Buddha</label><br>  
  
<input type="radio" name="agama" id="konghucu"  
value="konghucu" required>
```

```
<label for="html">Konghucu</label><br>
</div>
</div>

<div class="form-group row">
<div class="col-sm-6">
<label class="text-gray-900"
for="alamat">Alamat*</label>
<input type="text" class="form-control
form-control-user" name="alamat"
id="alamat" placeholder="Enter Address" required>
</div>
<div class="col-sm-6">
<label class="text-gray-900" for="no_hp">Nomor
Handphone*</label>
<input type="number" class="form-control
form-control-user" name="no_hp"
id="no_hp" placeholder="Enter handphone number" required>
</div>
</div>

<div class="form-group row">
<div class="col-sm-6">
<label class="text-gray-900" for="no_bpjs">Nomor
BPJS</label>
<input type="number" class="form-control
form-control-user" name="no_bpjs"
id="no_bpjs" placeholder="Enter BPJS Number">
</div>
</div>
```

```
<div class=" row">
  <div class="col-sm-3">
    <a class="btn btn-primary btn-user btn-block"
      href="{{url_for('manage_accounts')}}">
      Kembali</button>
    </a>
  </div>

  <div class="col-sm-3"></div>
  <div class="col-sm-3"></div>
  <div class="col-sm-3">
    <button type="submit" class="btn btn-primary btn-user
      btn-block">
      Submit</button>
  </div>
  </div>
</form>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
<!-- /.container-fluid -->
</div>
<!-- End of Main Content -->
{% endblock content %}
```

LAMPIRAN J

Code Untuk Web Service Registrasi Pasien Dari Klinik

```
@bp.route('/pasien', methods =['POST'])

def addpasien():
    try:
        data = {"email": request.form['email'],
                "password": request.form['passw'],
                "nik": request.form['nik'],
                "nama":request.form['nama'],
                "kelamin": request.form['kelamin'],
                "agama":request.form['agama'],
                "born_date":request.form['born_date'],
                "usia":request.form['usia'],
                "alamat": request.form['alamat'],
                "no_hp": request.form['no_hp'],
                "created_at" : time.strftime("%d/%m/%Y %H:%M:%S"),
                "updated_at" : time.strftime("%d/%m/%Y %H:%M:%S"),
                "list_image_id": [],
                "verif": '1'}

        cek = get_pasien(data)

        if cek == None:
            row = insert_pasien(data)
            print("Berhasil input pasien baru")
            flash("Berhasil input pasien baru")
            return redirect(url_for('add_new_patient'))
        else:
```

```
print("Gagal input pasien baru")
flash("Gagal input pasien baru")
return redirect(url_for('add_new_patient'))
except Exception as ex:
    print("Gagal input pasien baru")
    flash("Gagal input pasien baru")
    return redirect(url_for('add_new_patient'))
```

LAMPIRAN K

Code Untuk View List Permohonan Akun Pasien

```
{% extends "layout/Layout.html" %}

{% block title %} Daftar Permohonan Pasien Baru
{% endblock %}

{% block content %}

<!-- Content Wrapper -->

<div id="content-wrapper" class="d-flex flex-column
my-0">

<!-- Begin Page Content -->

<div class="container-fluid">

<!-- Outer Row -->

<div class="row justify-content-center">

<div class="card-body">

<div class="text-left">

<h1 class="h4 text-gray-900 my-3">Daftar Permohonan
Pasien Baru</h1>

</div>

{% with messages = get_flashed_messages() %}

{% if messages %}

{% for message in messages %}

<p>{{ message }}</p>

{% endfor %}

{% endif %}

{% endwith %}

<!-- DataTales Example -->
```

```
<div class="card mb-4">
<div class="card-body">
<div class="table-responsive">
<table class="table table-bordered" id="dataTable"
width="100%" cellspacing="0">
<thead>
<tr>
<th>NIK</th>
<th>>Nama</th>
<th>Jenis Kelamin</th>
<th>Usia</th>
<th>Action</th>
</tr>
</thead>
<tbody>
{%
  for item in data %}
<tr>
<td>{{ item.nik }}</td>
<td>{{ item.nama }}</td>
<td>{{ item.kelamin }}</td>
<td>{{ item.usia }}</td>
<td><a class="btn btn-primary btn-user btn-block"
href="{{url_for('profil_pemohon_pasien_baru', _id =
item._id )}}">
lihat</a>
</td>
</tr>
{%
  endfor %}
```

```
</tbody>
</table>
</div>
<div class=" row">
<div class="col-sm-3">
<a class="btn btn-primary btn-user btn-block"
 href="{{url_for('manage_accounts')}}">
Kembali</button>
</a>
</div>
<div class="col-sm-3"></div>
<div class="col-sm-3"></div>
<div class="col-sm-3"></div>
</div>
</div>
</div>
</div>
<!-- /.container-fluid -->
</div>
<!-- End of Main Content -->
{ % endblock content %}
```

LAMPIRAN L

Code Untuk Web Service List Permohonan Akun Pasien

```
#akses semua pasien yang belum terverifikasi klinik
@bp.route('/pasien_unverify', methods =['GET'])

def get_pasiens_unverify():
    a = db.get_pasiens_unverify()
    a_serializable = [ {'_id': str(patient['_id']),
                        'email':patient['email'], 'nama':patient['nama'],
                        'born_date':patient['born_date'],
                        'usia':patient['usia'], 'kelamin':patient['kelamin'],
                        'agama':patient['agama'], 'alamat':patient['alamat'],
                        'no_hp':patient['no_hp'], 'nik':patient['nik']} for
    patient in a]
    print("pass")
    return Response(response =
                    json.dumps(list(a_serializable)),
                    mimetype="application/json", status=200)
```

LAMPIRAN M

Code Untuk Web Wervice List Verifikasi/Terima pasien

```
#akses merubah pasien belum terverifikasi menjadi pasien  
terverifikasi  
  
@bp.route('/accept_verif_pasien/<_id>', methods =['GET' ])  
  
def accept_unverify_patient(_id):  
  
    a = db.verify_pasien(_id)  
  
    print("pass")  
  
    flash("Berhasil verifikasi pasien")  
  
    return redirect(url_for('list_request_new_patient'))
```

LAMPIRAN N

Code Untuk Web Service List Tolak Verifikasi Pasien

```
#akses merubah pasien belum terverifikasi menjadi  
pasien terblokir  
@bp.route('/block_verif_pasien/<_id>', methods=  
['GET'])  
  
def block_unverify_patient(_id):  
    a = db.block_pasien(_id)  
    print("pass")  
    flash("Berhasil blokir pasien")  
    return redirect(url_for('list_request_new_patient'))
```

LAMPIRAN O

Code Untuk View List Pasien Terdaftar Klinik

```
{% extends "layout/Layout.html" %}

{% block title %} Daftar Pasien {% endblock %}

{% block content %}

<div id="content-wrapper" class="d-flex flex-column
my-0">

<div class="container-fluid">

<div class="row justify-content-center">

<div class="card-body">

<div class="text-left">

<h1 class="h4 text-gray-900 my-3">Daftar Pasien</h1>

</div>

<div class="card-body">

<div class="table-responsive">

<table class="table table-bordered" id="dataTable"
width="100%" cellspacing="0">

<thead>

<tr>

<th>NIK</th>

<th>Nama</th>

<th>Jenis Kelamin</th>

<th>Usia</th>

<th>action</th>

</tr>

</thead>
```

```
<tbody>
{%
  for item in data %
}
<tr>
<td>{{ item.nik }}</td>
<td>{{ item.nama }}</td>
<td>{{ item.kelamin }}</td>
<td>{{ item.usia }}</td>
<td><a class="btn btn-primary btn-user btn-block"
      href="{{url_for('profil_pasien', _id = item._id
) }}">lihat</a></td>
</tr>
{%
  endfor %
}
</tbody>
</table>
</div>
<div class="row">
<div class="col-sm-3">
<a class="btn btn-primary btn-user btn-block"
      href="{{url_for('manage_accounts')}}">
Kembali</button>
</a>
</div>
<div class="col-sm-3"></div>
<div class="col-sm-3"></div>
</div>
{%
  endblock content %
}
```

LAMPIRAN P

Code Untuk Web Service List Pasien Terdaftar Klinik

```
#akses semua pasien yang telah terverifikasi klinik
@bp.route('/pasien', methods =['GET'])

def get_pasiens():
    a = db.get_pasiens()
    a_serializable = [ {'_id': str(patient['_id']),
                        'email':patient['email'], 'nama':patient['nama'],
                        'born_date':patient['born_date'],
                        'usia':patient['usia'], 'kelamin':patient['kelamin'],
                        'agama':patient['agama'], 'alamat':patient['alamat'],
                        'no_hp':patient['no_hp'], 'nik':patient['nik']} for
    patient in a]
    return Response(response =
                    json.dumps(list(a_serializable)),
                    mimetype="application/json", status=200)
```

LAMPIRAN Q

Code Untuk View Detail Hasil Pemeriksaan Kesehatan

```
{% extends "layout/Layout.html" %}

{% block title %} Hasil Pemeriksaan
Kesehatan {% endblock %}

{% block content %}

<div id="content-wrapper" class="d-flex flex-
column my-0">

<div class="container">

<div class="row justify-content-center">

<div class="card-body">

<div class="text-left">

<h1 class="h4 text-gray-900 my-3">Hasil
Pemeriksaan Kesehatan</h1>

</div>

<div class="row">

<div class="col-lg-12">

<div class="p-5">

{% for item in data %}

<div class="row">

<div class="col-sm-6">

<label class="text-gray-900" for="tanggal">
Tanggal Pemeriksaan</label>

<br>

<label for="email">{{ item.tanggal }}</label>

</div>
```

```
</div>

<br>

<div class=" row">

<div class="col-sm-6">

<label class="text-gray-900" for="tekanan_>
darah">Tekanan Darah</label>

<br>

<label for="born_date">{{ item.tekanan_>
darah }}</label>

</div>

<div class="col-sm-6">

<label class="text-gray-900" for="nadi">
Nadi</label>

<br>

<label for="alamat">{{ item.nadi }}</label>

</div>

</div>

<div class=" row">

<div class="col-sm-6">

<label class="text-gray-900" for="suhu">Suhu<
Tubuh</label>

<br>

<label for="born_date">{{ item.suhu }}</label>

</div>

<div class="col-sm-6">

<label class="text-gray-900" for="gula_darah_>
_sewaktu">Gula Darah Sewaktu</label>

<br>
```

```
<label for="alamat">{{ item.gula_darah_
    sewaktu }}</label>
</div>
</div>
<div class=" row">
<div class="col-sm-6">
<label class="text-gray-900" for="ABPI">
ABPI</label>
<br>
<label for="born_date">{{ item.
    ABPI }}</label>
</div>
</div>
<div class=" row">
<div class="col-sm-3">
<a type="submit" class="btn btn-primary
btn-user btn-block" href="{{url_for('list
    _medical_check_data', nik=item.nik )}}">
Back</a>
</div>
<div class="col-sm-3"></div>
<div class="col-sm-3"></div>
</div>
{%
    endfor %}
</div>
{%
    endblock content %}
```

LAMPIRAN R

Code Untuk Web Service Tambah Hasil Pemeriksaan Kesehatan

```
@bp.route('/add_data_pemeriksaan_kesehatan',
methods =['POST'])
def post_data_pk():
    try:
        a = list(get_data_pemeriksaan_kesehatan())
        data = {"_id": 100000000 + len(a) + 1,
                "tanggal": request.form['tanggal'],
                "username": request.form['username'],
                "nik": request.form['nik'],
                "tekanan_darah": request.form['tekanan_darah'],
                "nadi": request.form['nadi'],
                "suhu": request.form['suhu'],
                "gula_darah_sewaktu":request.form['gula_darah_sewaktu'],
                "ABPI" : request.form['ABPI']}
        insert_data_pemeriksaan_kesehatan(data)
        print("Berhasil input data pemeriksaan kesehatan")
        return Response(response = json.dumps(data), status=200)
    except Exception as ex:
        print("Gagal input data pemeriksaan kesehatan")
        return Response(response = json.dumps({"message" : "error
encountered"}), mimetype="application/json", status=500)
```

LAMPIRAN S

Code Untuk Web Service Untuk Mendapatkan Detail Data Hasil Pemeriksaan Kesehatan Pasien Berdasar Tanggal Pemeriksaan

```
@bp.route('/detail_data_pk_pasien/<_id>',
methods =['GET'])

def detail_data_pk(_id):
    a = get_detail_data_pemeriksaan_kesehatan
    _one_patient(_id)
    print(a)
    a_serializable = [a]
    return Response(response = json.dumps(list
(a_serializable)), mimetype="application/json",
status=200)
```

LAMPIRAN T

Code Untuk View List Inventaris

```
{% extends "layout/Layout.html" %}

{% block title %} Inventaris

{% endblock %}

{% block content %}

<div id="content-wrapper" class="d-flex flex-column
my-0">

<h1 class="h4 text-gray-900 my-3">Inventaris</h1>

</div>

{% with messages = get_flashed_messages() %}

{% if messages %}

{% for message in messages %}

<p>{{ message }}</p>

{% endfor %}

{% endif %}

{% endwith %}

<div class="card mb-4">

<div class="card-body">

<div class="table-responsive">

<table class="table table-bordered" id="dataTable"
width="100%" cellspacing="0">

<thead>

<tr>

<th>ID Inventaris</th>

<th>Nama Inventaris</th>
```

```

<th>Tipe Inventaris</th>
<th>Harga</th>
<th>Action</th>
</tr>
</thead>
<tbody>
{%
  for item in data %}
<tr>
<td>{{ item._id }}</td>
<td>{{ item.nama_inventaris }}</td>
<td>{{ item.tipe_inventaris }}</td>
<td>{{ item.harga }}</td>
<td><a class="btn btn-primary btn-user btn-block"
      href="{{ url_for('detail_inventaris',
      _id = item._id )}}>lihat</a></td>
</tr>
{%
  endfor %}
</tbody>
</table>
</div>
<div class="row">
<div class="col-sm-3">
<a type="submit" class="btn btn-primary btn-user
      btn-block" href="{{ url_for('inventaris') }}">
      Tambah Inventaris</a>
{%
  endblock content %}

```

LAMPIRAN U

Code Untuk View Tambah Inventaris

```
{% extends "layout/Layout.html" %}

{% block title %} Tambah Inventaris

{% endblock %}

{% block content %}

<div id="content-wrapper" class="d-flex flex-column
my-0">

<h1 class="h4 text-gray-900 my-3">Tambah Inventaris
Baru</h1>

</div>

{% with messages = get_flashed_messages() %}

{% if messages %}

{% for message in messages %}

<p>{{ message }}</p>

{% endfor %}

{% endif %}

{% endwith %}

<div class="row">

<form class="user" method="POST" action="
/inventaris">

<div class="form-group row">

<div class="col-sm-6">

<label class="text-gray-900" for="nama_inventaris">
Nama Inventaris*</label>

<input type="text" class="form-control"
placeholder="Masukkan Nama Inventaris" name="nama_inventaris">

<small class="text-gray-900 form-text">Format: Nama
Inventaris</small>

<div class="d-grid gap-2 d-md-block">
<button type="submit" class="btn btn-primary">Simpan</button>
<button type="button" class="btn btn-secondary">Batal</button>
</div>

</div>
</div>
</form>
</div>
```

```
form-control-user" name="nama_inventaris"
id="nama_inventaris" required>
</div>

<div class="col-sm-6">
<label class="text-gray-900" for="keterangan">Keterangan*</label>
<input type="text" class="form-control form-control-user"
name="keterangan" id="keterangan" required>
</div>
</div>

<div class="form-group row">
<div class="col-sm-6">
<label class="text-gray-900" for="harga">
Harga*</label>
<input type="number" class="form-control form-
control-user" name="harga" id="harga" required>
</div>
<div class="col-sm-6">
<label class="text-gray-900" for="harga">
Jumlah*</label>
<input type="number" class="form-control form-
control-user" name="jumlah" id="jumlah" required>
</div>
</div>

<div class="form-group row">
<div class="col-sm-6">
<label class="text-gray-900" for="tipe_inventaris">
Tipe Inventaris*</label>
<br>
```

```
<input type="radio" name="tipe_inventaris"
id="obat" value="obat" required>
<label for="html">Obat</label><br>
<input type="radio" name="tipe_inventaris"
id="balutan" value="balutan" required>
<label for="html">Balutan</label><br>
</div>
</div>
<div class=" row">
<div class="col-sm-3">
<a class="btn btn-primary btn-user btn-block"
href="{{url_for('list_inventaris')}}">
Kembali</button>
</a>
</div>
<div class="col-sm-3"></div>
<div class="col-sm-3">
<div class="col-sm-3">
<button type="submit" class="btn btn-primary
btn-user btn-block">
Submit</button>
</div>
</div>
</form>
</div>
</div>
{ % endblock content %}
```

LAMPIRAN V

Code Untuk Web Service Tambah Inventaris

```
@bp.route('/inventaris', methods =['POST'])

def addinventory():
    try:
        data = {"nama_inventaris": request.form['nama_inventaris'],
                "tipe_inventaris": request.form['tipe_inventaris'],
                "harga": request.form['harga'],
                "keterangan":request.form['keterangan'],
                "jumlah" : request.form['jumlah']}
        cek = get_inventaris(data)

        if cek == None:
            row = insert_inventaris(data)
            print("berhasil input inventaris")
            flash("berhasil input inventaris")
            return redirect(url_for('inventaris'))
        else:
            #jika sudah ada data yang sama maka tidak
            bisa daftar lagi
            print("gagal input inventaris")
            flash("gagal input inventaris")
            return redirect(url_for('inventaris'))
    except Exception as ex:
        print(ex)
        flash("gagal input inventaris")
        return redirect(url_for('inventaris'))
```

LAMPIRAN W

Code Untuk View List Layanan

```
{% extends "layout/Layout.html" %}

{% block title %} Layanan

{% endblock %}

{% block content %}

<div id="content-wrapper" class="d-flex flex-column my-0">

<div class="container-fluid">

<div class="row justify-content-center">

<div class="card-body">

<div class="text-left">

<h1 class="h4 text-gray-900 my-3">Inventaris</h1>

</div>

{% with messages = get_flashed_messages() %}

{% if messages %}

{% for message in messages %}

<p>{{ message }}</p>

{% endfor %}

{% endif %}

{% endwith %}

<div class="card mb-4">

<div class="card-body">

<div class="table-responsive">

<table class="table table-bordered" id="dataTable"

width="100%" cellspacing="0">

<thead>
```

```
<tr>
<th>ID Layanan</th>
<th>Nama Layanan</th>
<th>Harga</th>
<th>Action</th>
</tr>
</thead>
<tbody>
{%
  for item in data
%}
<tr>
<td>{{ item._id }}</td>
<td>{{ item.nama_layanan }}</td>
<td>{{ item.harga }}</td>
<td><a class="btn btn-primary btn-user btn-block"
      href="{{url_for('detail_layanan', _id = item._id
)} }">lihat</a></td>
</tr>
{%
  endfor
%}
</tbody>
</table>
</div>
<div class="row">
<div class="col-sm-3"></div>
<div class="col-sm-3"></div>
<div class="col-sm-3"></div>
<div class="col-sm-3">
<a type="submit" class="btn btn-primary btn-user btn-block"
      href="{{url_for('layanan')}}">
```


LAMPIRAN X

Code Untuk View Tambah Layanan

```
{% extends "layout/Layout.html" %}

{% block title %} Tambah Layanan

{% endblock %}

{% block content %}

<div id="content-wrapper" class="d-flex flex-column
my-0">

<div class="container">

<div class="row justify-content-center">

<div class="card-body">

<div class="text-left">

<h1 class="h4 text-gray-900 my-3">Tambah Layanan
Baru</h1>

</div>

{% with messages = get_flashed_messages() %}

{% if messages %}

{% for message in messages %}

<p>{{ message }}</p>

{% endfor %}

{% endif %}

{% endwith %}

<div class="row">

<div class="col-lg-12">

<div class="p-5">

<form class="user" method="POST" action="/layanan">
```

```
<div class="form-group row">
<div class="col-sm-6">
<label class="text-gray-900" for="nama_layanan">
Nama Layanan*</label>
<input type="text" class="form-control
form-control-user" name="nama_layanan"
id="nama_layanan" required>
</div>
<div class="col-sm-6">
<label class="text-gray-900"
for="keterangan">Keterangan*</label>
<input type="text" class="form-control
form-control-user" name="keterangan" id="keterangan"
required>
</div>
</div>
<div class="form-group row">
<div class="col-sm-6">
<label class="text-gray-900" for="harga">
Harga*</label>
<input type="number" class="form-control
form-control-user" name="harga" id="harga"
required>
</div>
</div>
<div class="row">
<div class="col-sm-3">
<a class="btn btn-primary btn-user btn-

```


LAMPIRAN Y

Code Untuk Web Service Tambah Layanan

```
@bp.route('/layanan', methods =['POST'])

def addservice():
    try:
        data = {
            "nama_layanan": request.form['nama_layanan'],
            "keterangan": request.form['keterangan'],
            "harga":request.form['harga'],
        }
        cek = get_layanan(data)
        if cek == None:
            row = insert_layanan(data)
            print("Berhasil input layanan")
            flash("Berhasil input layanan")
            return redirect(url_for('layanan'))
        else:
            print("Gagal input layanan")
            flash("Gagal input layanan")
            return redirect(url_for('layanan'))
    except Exception as ex:
        print("Gagal input layanan")
        flash("Gagal input layanan")
        return redirect(url_for('layanan'))
```

DAFTAR RIWAYAT HIDUP



MUHAMMAD INSAN KHAMIL. Lahir di Boyolali, 21 Februari 1998. Anak pertama dari pasangan Bapak Sriyono dan Ibu Inong Martini. Saat ini beralamatkan di Jl. Mawar 2 no. 5C RT 08 RW 13 kel. Bintaro kec. Pesanggrahan, Kota Jakarta Selatan.

No. Ponsel : 081293402756

Email : khami_insan0@yahoo.com

Riwayat Pendidikan : Penulis mengawali pendidikan di MIN 15 Bintaro pada tahun 2004 - 2010. Setelah itu, penulis melanjutkan studi ke SMPN 178 Jakarta hingga tahun 2013. Kemudian melanjutkan ke SMAN 87 Jakarta hingga tahun 2016. Di Tahun 2016 penulis melanjutkan ke Universitas Negeri Jakarta (UNJ), Program Studi Ilmu Komputer, melalui jalur SBMPTN dengan beasiswa BIDIKMISI dan kemudian lulus di tahun 2023.

Riwayat Organisasi : Selama di bangku perkuliahan, penulis tergabung dengan organisasi kemahasiswaan seperti, Masjid Ulul Albaab yang sekarang Lembaga Dakwah Ulul Albab sebagai staff Huda periode 2017 dan 2018. Penulis juga berpartisipasi dalam organisasi Badan Eksekutif Mahasiswa di Program Studi Ilmu Komputer periode 2018, dimana penulis tergabung sebagai kepala departemen rohani islam. Penulis juga kerap mengikuti kepanitiaan kegiatan yang diadakan oleh lembaga dakwah fakultas (MUA), dan komunitas atau *underbow* lembaga (Default).

Tentang Skripsi : Skripsi ini merupakan karya terbaik penulis yang diberikan untuk civitas akademika Universitas Negeri Jakarta. Besar harapan penulis agar skripsi ini dapat bermanfaat bagi semua yang membaca dan menggunakan hasil penelitian ini.