IMPLEMENTASI API BRIDGING BPJS BERBASIS WEB MENGGUNAKAN METODE *WATERFALL*

IMPLEMENTATION OF BPJS WEB-BASED BRIDGING API USING THE WATERFALL METHOD

Putri Indah Cahyani[1], Danur Wijayanto[2]
E-mail: [1]*2211501021@student.unisayogya.ac.id, [2]danurwijayanto@unisayogya.ac.id
[1,2] Program Studi Teknologi Informasi, Fakultas Sains dan Teknologi, Universitas 'Aisyiyah Yogyakarta

Abstrak

Pemanfaatan teknologi informasi dalam sektor kesehatan, khususnya melalui Badan Penyelenggara Jaminan Sosial (BPJS), semakin penting untuk meningkatkan efisiensi layanan kesehatan. Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem bridging BPJS yang terpisah dari Sistem Informasi Manajemen Rumah Sakit (SIMRS) di RSU XYZ dengan menggunakan Metode *Waterfall*. Proses pengembangan mencakup analisis kebutuhan, desain sistem, implementasi, integrasi, dan pemeliharaan. Hasil pengujian menggunakan Metode *Black box* menunjukkan bahwa semua endpoint berfungsi dengan baik, dengan tingkat keberhasilan 100%, menandakan sistem ini siap digunakan secara operasional. Pemisahan sistem ini diharapkan dapat meningkatkan stabilitas dan efisiensi operasional, serta mengurangi risiko gangguan layanan yang dapat mempengaruhi pasien. Selain itu, sistem ini memungkinkan pencatatan log aktivitas yang lebih baik, yang mendukung analisis kinerja dan pengelolaan data pasien secara lebih akurat dan efisien. Penelitian ini memberikan kontribusi signifikan terhadap pengembangan sistem informasi kesehatan yang lebih handal dan efisien, serta menawarkan solusi untuk tantangan integrasi data antara SIMRS dan BPJS. Diharapkan, implementasi sistem ini dapat mempercepat proses pengelolaan klaim dan verifikasi, serta meningkatkan kualitas layanan kesehatan bagi peserta dan fasilitas kesehatan, sehingga memberikan manfaat yang lebih besar bagi semua pihak yang terlibat dalam sistem kesehatan secara keseluruhan dan berkelanjutan dalam jangka panjang untuk masyarakat.

Kata kunci: SIMRS, sistem *bridging*, *waterfall*, efisiensi layanan kesehatan.

Abstract

The use of information technology in the health sector, especially through the Social Security Administering Agency (BPJS), is increasingly important to improve the efficiency of health services. This study aims to design and implement a BPJS bridging system that is separate from the Hospital Management Information System (SIMRS) at RSU XYZ using the Waterfall method. The development process includes needs analysis, system design, implementation, integration, and maintenance. The results of testing using the Black box method show that all endpoints function well, with a 100% success rate, indicating that this system is ready for operational use. This system separation is expected to improve operational stability and efficiency, as well as reduce the risk of service disruptions that can affect patients. In addition, this system allows for better activity log recording, which supports performance analysis and more accurate and efficient patient data management. This study makes a significant contribution to the development of a more reliable and efficient health information system, and offers solutions to the challenges of data integration between SIMRS and BPJS. It is expected that the implementation of this system can accelerate the process of claim management and verification, as well as improve the quality of health services for participants and health facilities, thus providing greater benefits for all parties involved in the health system as a whole and sustainable in the long term for the community.

***Keywords**: SIMRS, bridging system, waterfall, health service efficiency.*

1. PENDAHULUAN

Pemanfaatan teknologi informasi dalam sektor kesehatan semakin penting untuk meningkatkan efisiensi layanan kesehatan, meskipun terdapat berbagai tantangan seperti keterbatasan sumber daya manusia, aspek keuangan, regulasi, dan keamanan data [1]. Salah satu pemanfaatan teknologi di sektor ini adalah Badan Penyelenggara Jaminan Sosial (BPJS) yang terus berupaya meningkatkan kualitas layanan kepada peserta dan fasilitas kesehatan, termasuk melalui pengembangan *bridging system*. *Bridging System* merupakan penggunaan aplikasi yang berbasis *web service* yang mengintergrasikan sistem pelayanan kesehatan menjadi satu kesatuan [2].

Sistem Informasi Manajemen Rumah Sakit (SIMRS) penting untuk pengelolaan operasional rumah sakit, termasuk pendaftaran pasien, pemeriksaan medis, layanan farmasi, dan kasir. SIMRS perlu terintegrasi dengan instansi eksternal seperti BPJS Kesehatan dan Kementerian Kesehatan. *Fitur* seperti *VClaim* dan Antrian *Online* mempermudah akses pasien. Namun, penggabungan SIMRS dengan Antrian *Online* di RSU XYZ menghadapi tantangan, yaitu potensi gangguan pada kinerja sistem utama akibat beban pengelolaan antrian yang tinggi.

Untuk mengatasi masalah ini, dilakukan implementasi API *bridging* dengan pendekatan model *Waterfall*. Model *Waterfall* adalah Metode pengembangan perangkat lunak yang bersifat linier dan berurutan, di mana setiap fase harus diselesaikan sepenuhnya sebelum melanjutkan ke fase berikutnya. Tahapan dalam model ini meliputi analisis kebutuhan, desain sistem, implementasi, pengujian, dan pemeliharaan [3]. Pemilihan Metode *Waterfall* dalam pengembangan sistem didasarkan pada beberapa alasan utama. Pertama, model ini cocok untuk proyek dengan persyaratan yang jelas dan spesifik, mengurangi risiko perubahan yang tidak terduga. *Waterfall* ideal untuk pembangunan sistem secara menyeluruh, dari pengumpulan persyaratan hingga pengujian produk. Struktur terorganisir di setiap fase memudahkan pengendalian dan pemantauan proyek, memungkinkan manajer untuk mengawasi kemajuan dan mengidentifikasi masalah [4]

Salah satu elemen utama dalam sistem ini adalah pengelolaan dua jenis data pengiriman, yaitu *kode booking* dan *task ID*. Kedua data ini memiliki alur yang sama, yaitu dimulai dengan pengambilan data dari *endpoint* API, dilanjutkan dengan proses validasi untuk memastikan akurasi data, hingga akhirnya data dikirimkan ke BPJS untuk diproses lebih lanjut. Proses ini dirancang untuk memastikan data yang dikirimkan benar-benar *valid* dan sesuai dengan persyaratan.

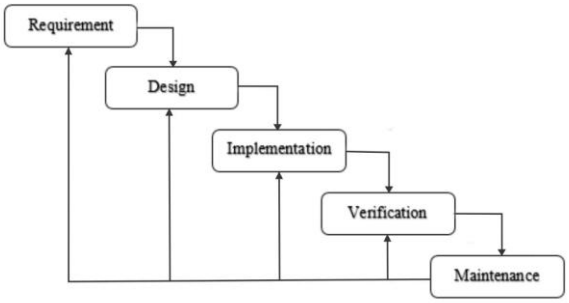
Tujuan implementasi sistem ini adalah memisahkan SIMRS pusat dari sistem *bridging* BPJS untuk mencegah gangguan antar sistem. Pemisahan ini diharapkan meningkatkan stabilitas dan efisiensi operasional di RSU XYZ, serta mengurangi risiko gangguan layanan. Sistem yang dibangun berbasis web, terintegrasi, dan modular, memberikan fleksibilitas dalam pengelolaan dan pengembangan di masa depan, serta memastikan kelancaran layanan.

2. METODOLOGI

Pendekatan yang digunakan dalam implementasi sistem *bridging* ini adalah Metode *Waterfall*. Dalam Metode ini, pembuatan sistem dilakukan secara teratur dan terstruktur pada setiap tahapannya [5]. Tahapan Metode *waterfall* dapat dilihat pada Gambar 1.

2.1 Tahap *Requirement* Analisis

Tahapan *requirement* analisis dimulai dengan diskusi bersama tim IT RSU XYZ untuk memahami kebutuhan dan kendala dalam sistem antrian BPJS. Dari hasil diskusi menetapkan tujuan utama, yaitu membangun sistem *bridging* BPJS terpisah dari SIMRS untuk meningkatkan efisiensi, keamanan, dan kecepatan pengelolaan data.



Gambar 1. Metode *Waterfall* [6]

2.2 Tahapan *System Design*

Pada tahapan design, membuat rancangan sistem *bridging* BPJS dengan menggunakan pemodelan berorientasi objek yaitu merancang *Flowchart*, *Entity Relationship Diagram* (ERD), *Use case Diagram* dan *Sequence Diagram*.

2.2.1 *Flowchart*

Flowchart adalah visualisasi yang menampilkan rangkaian proses atau tahapan dalam suatu sistem secara berurutan. Analis sistem menggunakan *flowchart* sebagai dokumentasi untuk menyampaikan struktur logis sistem kepada *programmer*, membantu menyelesaikan kendala selama pengembangan. Diagram ini menggunakan simbol standar untuk merepresentasikan berbagai proses yang saling terhubung melalui garis penghubung [7], [8].

2.2.2 *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) adalah suatu representasi visual yang menggambarkan hubungan antara entitas dan atributnya dalam suatu sistem, sehingga dapat membantu menggabungkan dan menghubungkan data dengan efektif [9],[10],[11]. ERD juga berfungsi sebagai model untuk menjelaskan struktur data dalam *database* dan membantu merancang sistem yang lebih terstruktur [12].

2.2.3 *Use Case Diagram*

Use Case Diagram adalah model dalam *Unified Modeling Language* (UML) yang menggambarkan hubungan antara aktor dan *use case* dalam sebuah sistem. Aktor dapat berupa manusia, pengelola, pelanggan, atau perangkat lain yang terhubung dengan sistem. Diagram ini memvisualisasikan interaksi aktor dengan sistem melalui berbagai skenario penggunaan, sehingga memperjelas fungsi dan hubungan dalam sistem [13], [14].

2.2.4 *Sequence Diagram*

Sequence Diagram menggambarkan urutan interaksi antara objek dalam suatu use case. Diagram ini digunakan untuk menunjukkan urutan pesan yang dikirim dan diterima oleh objek tertentu, serta menekankan pada waktu pengiriman pesan selama proses berlangsung. Selain itu, Sequence Diagram juga membantu dalam memahami alur logika dari sistem, memvisualisasikan bagaimana objek berkolaborasi untuk mencapai tujuan tertentu, dan memberikan gambaran yang jelas tentang bagaimana sistem berfungsi secara keseluruhan. Dengan demikian, diagram ini menjadi alat yang penting dalam analisis dan perancangan sistem perangkat lunak. [15].

2.3 Tahapan Implementation

Pada tahap implementasi, penerapan sistem dilakukan. Di tahap ini, dilakukan pengkodean dan pembuatan aplikasi berdasarkan analisis sistem dan desain sistem yang telah disusun. Proses ini mencakup pengembangan perangkat lunak yang sesuai dengan spesifikasi yang telah ditentukan, serta memastikan bahwa semua fungsi dan *fitur* yang direncanakan dapat berjalan dengan baik dalam lingkungan yang telah disiapkan. [16].

2.4 Tahapan *Integration & Testing*

Integrasi dan Pengujian Sistem merupakan proses penggabungan seluruh program dan pelaksanaan pengujian sistem secara menyeluruh. Pada tahap ini, semua komponen yang telah dikembangkan akan diintegrasikan untuk memastikan bahwa mereka bekerja sama dengan baik

dan memenuhi spesifikasi yang telah ditetapkan. Pengujian dilakukan untuk mengidentifikasi dan memperbaiki potensi masalah, serta untuk memastikan bahwa sistem berfungsi sesuai dengan harapan.[17].

2.5 Tahapan *Operation & Maintenance*

Operation and Maintenance adalah tahap terakhir yang dilakukan, yaitu pemeliharaan sistem. Pemeliharaan ini memungkinkan pengembang untuk memperbaiki kesalahan yang mungkin tidak terdeteksi pada tahap-tahap sebelumnya [18].

3. HASIL DAN PEMBAHASAN

3.1 Tahapan *Requirement Analisis*

Sistem yang saat ini berjalan menunjukkan bahwa integrasi antara SIMRS pusat dan sistem *bridging* BPJS menghadapi sejumlah tantangan, di antaranya ketergantungan tinggi antara kedua sistem, sehingga gangguan pada salah satu sistem dapat langsung memengaruhi kinerja keseluruhan. Ketika sistem *bridging* BPJS mengalami kendala teknis, proses pengolahan data pasien di SIMRS menjadi terhambat, yang berdampak pada kelancaran layanan kesehatan di RSU XYZ. Selain itu, tidak adanya pencatatan log aktivitas yang terpusat menyulitkan analisis kinerja dan identifikasi masalah, serta memperbesar risiko *downtime* yang memengaruhi pengelolaan data BPJS dan antrian layanan. Untuk mengatasi tantangan tersebut, diusulkan pemisahan antara SIMRS dan sistem *bridging* BPJS agar masing-masing dapat beroperasi secara independen tanpa saling memengaruhi jika terjadi gangguan. Pemisahan ini diharapkan mampu meningkatkan stabilitas operasional, efisiensi pengelolaan data pasien BPJS, dan kualitas layanan kesehatan di RSU XYZ melalui pencatatan log aktivitas yang sistematis, pengelolaan antrian online yang lebih baik, serta pengurangan risiko gangguan layanan.

3.2 Tahapan *Design*

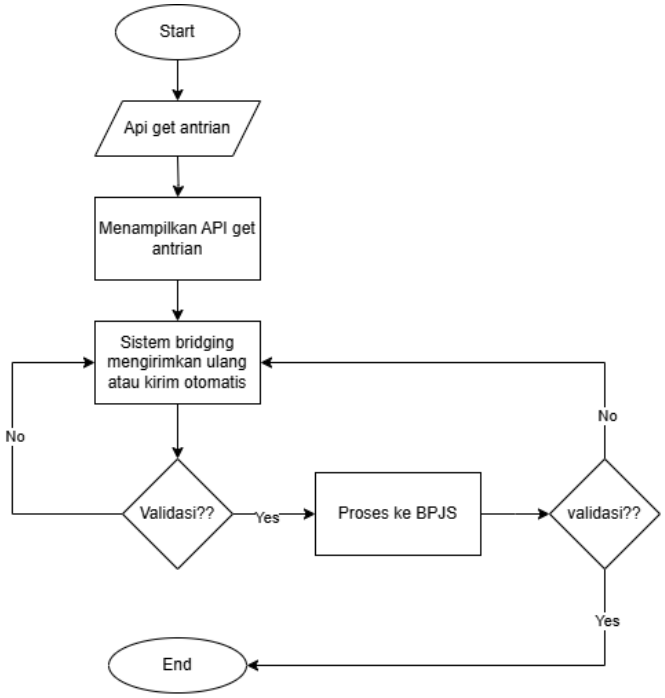
Setelah tahapan *requirement* analisis selesai dilakukan, berikutnya adalah tahapan *design system*. *Design* ini mencakup *Flowchart, Entity Relationship Diagram* (ERD), *Use Case Diagram* dan *sequence Diagram.*

3.2.1 *Flowchart*

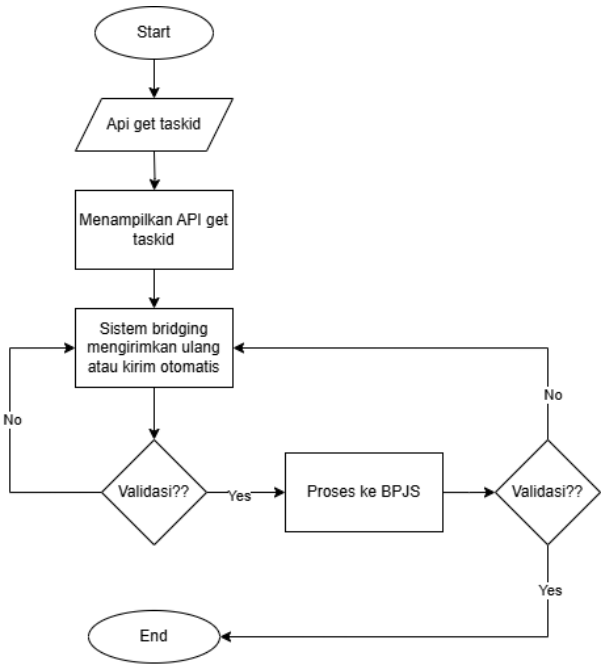
Berikut merupakan *Flowchart* Implementasi Api *Bridging* Bpjs Berbasis Web dilihat pada Gambar 2 dan Gambar 3.

Proses dimulai dengan memanggil API untuk mendapatkan data antrian, kemudian menampilkan hasilnya. Sistem *bridging* bertugas mengirimkan data secara otomatis atau mengulang pengiriman jika diperlukan. Selanjutnya, dilakukan validasi pertama; jika data tidak valid, sistem akan kembali ke proses *bridging* untuk mengulang atau memperbaiki pengiriman. Jika validasi berhasil, data dikirim ke BPJS untuk diproses lebih lanjut. Setelah itu, dilakukan validasi kedua; jika data tidak valid, proses kembali ke sistem *bridging*, namun jika validasi berhasil, proses dianggap selesai. Proses ini bertujuan memastikan data yang dikirimkan ke BPJS benar-benar *valid* untuk diproses.

Proses dimulai dengan pemanggilan API untuk mendapatkan *TaskID*, di mana hasilnya akan ditampilkan oleh sistem. Selanjutnya, sistem bridging bertugas untuk mengirimkan data secara otomatis atau mengulang pengiriman apabila diperlukan. Data kemudian melewati tahap validasi pertama, di mana jika data tidak *valid*, proses akan kembali ke sistem *bridging* untuk diperbaiki. Jika data *valid*, maka data tersebut diteruskan ke BPJS untuk diproses lebih lanjut. Setelah itu, dilakukan validasi kedua untuk memastikan data yang telah diproses memenuhi kriteria yang diharapkan. Jika validasi kedua gagal, proses kembali ke sistem *bridging*, tetapi jika validasi berhasil, maka proses dianggap selesai. Alur ini dirancang untuk memastikan data yang dikirim ke BPJS akurat dan sesuai dengan persyaratan.



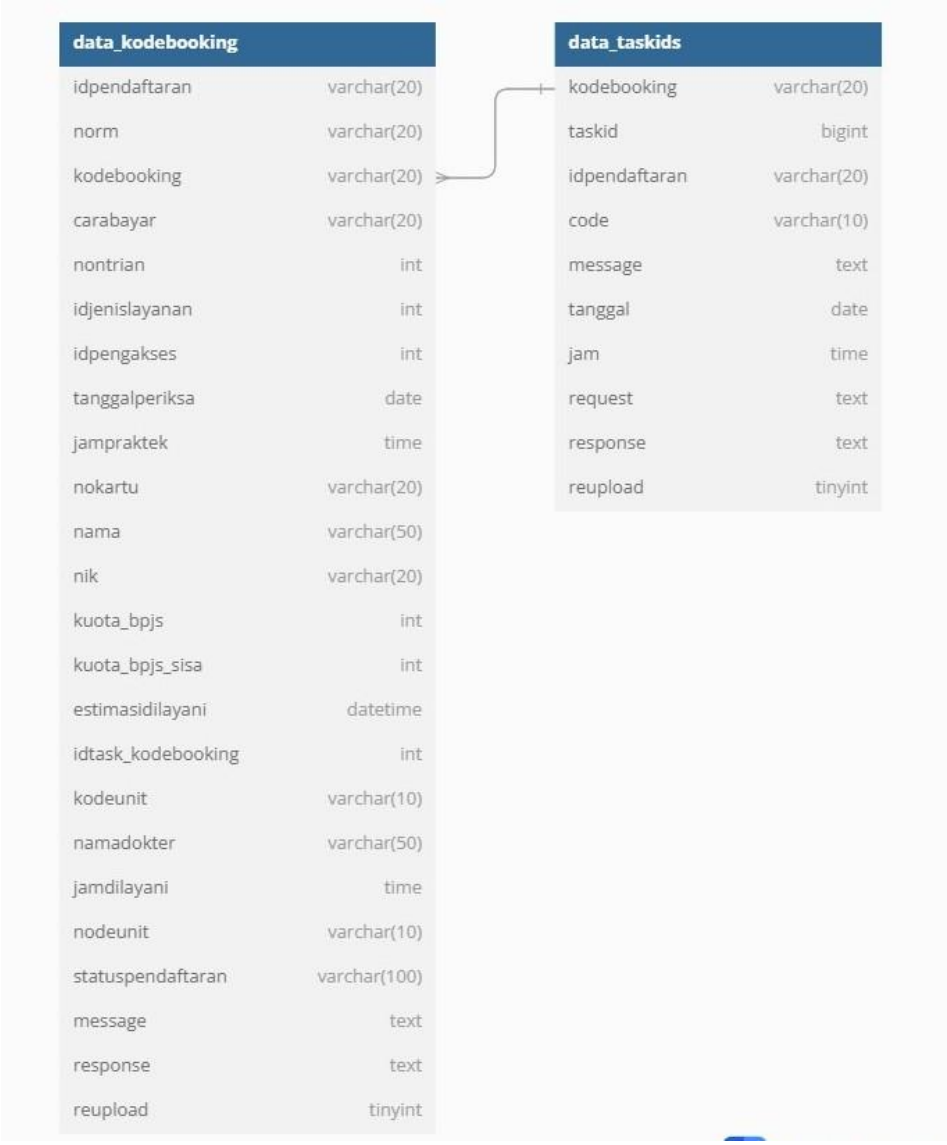
Gambar 2. *Flowchart Data Kodebooking*



Gambar 3. *Flowchart Task ID*

3.2.2 *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) menggunakan jenis *Crow's Foot* untuk mendukung sistem API *Bridging* BPJS berbasis web. Diagram di bawah ini menggambarkan hubungan antara dua entitas utama yaitu data_kodebooking dan data_taskids bisa dilihat pada Gambar 4.

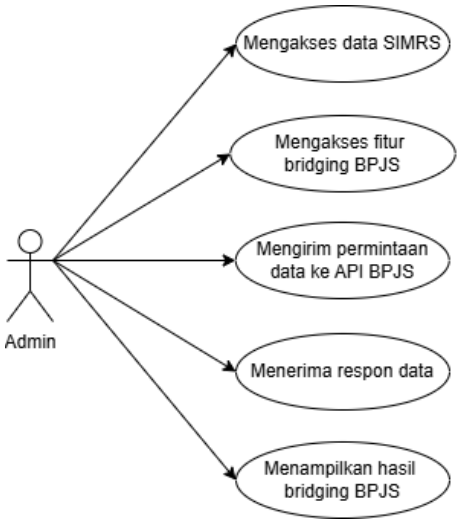


Gambar 4. *Entity Relationship Diagram*

Entity Relationship Diagram (ERD) mendukung sistem API *Bridging* BPJS untuk pengelolaan data layanan kesehatan. Tabel data_kodebooking mencatat informasi seperti NoRM, kode booking, tanggal pemeriksaan, dan status pendaftaran, sedangkan tabel data_taskids mencatat *log* aktivitas seperti waktu *request*, kode status, dan *respons*. Relasi *one-to-many* menghubungkan satu kode booking dengan banyak log aktivitas. Pengelolaan data dilakukan secara terstruktur melalui API, memastikan pencatatan efisien, *respons* cepat, dan akurasi layanan tanpa membebani SIMRS.

3.2.3 *Use Case Diagram*

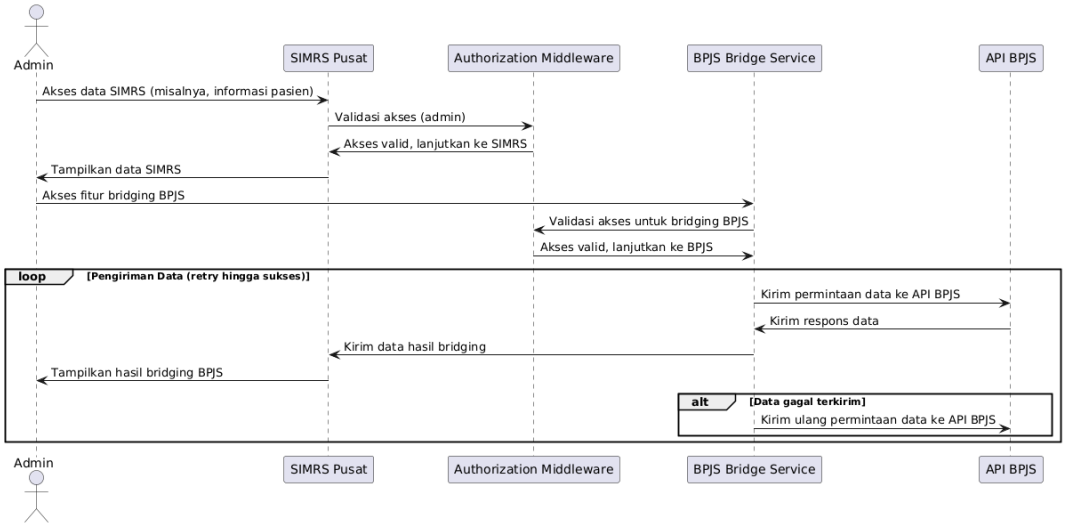
Berikut ini merupakan *use case diagram* Implementasi Api *Bridging* Bpjs Berbasis Web, yang dapat dilihat pada Gambar 5. Admin memulai interaksi dengan sistem untuk mengakses data SIMRS yang diperlukan, seperti data pasien atau informasi terkait lainnya. Setelah itu, *admin* mengaktifkan *fitur bridging* BPJS yang berfungsi menjembatani data antara SIMRS dan layanan BPJS. Sistem kemudian mengirimkan permintaan ke API BPJS untuk memproses data yang diperlukan, seperti pendaftaran atau pengecekan informasi pasien. Setelah *respon* diterima dari API BPJS, sistem menampilkan hasil *bridging* tersebut kepada *admin* dalam bentuk data atau laporan yang siap dianalisis atau diproses lebih lanjut.



Gambar 5. *Use Case Diagram*

3.2.4 *Sequence Diagram*

Berikut ini merupakan *Sequence Diagram* Implementasi Api *Bridging* Bpjs Berbasis Web, yang dapat dilihat pada Gambar 6.



Gambar 6. *Sequence Diagram*

Proses pengiriman data dimulai ketika *Admin* mengakses data dari SIMRS untuk kebutuhan operasional. Permintaan divalidasi melalui *Authorization Middleware* (Authz) untuk memastikan akses yang aman. Setelah data ditampilkan, *Admin* dapat meminta *fitur bridging* BPJS melalui BPJS *Bridge Service*, yang juga divalidasi oleh *Authz*. BPJS *Bridge Service* kemudian mengirimkan permintaan ke API BPJS, menerima *respons*, dan meneruskan data hasil *bridging* ke SIMRS untuk ditampilkan kepada *Admin*. Sistem dilengkapi dengan mekanisme *retry* yang secara otomatis mencoba mengirim ulang permintaan jika terjadi kegagalan, memastikan data terkirim sesuai kebutuhan *real-time.*

3.3 Tahapan *Implementation*

Pada tahap ini, sistem *bridging* BPJS yang dibangun menggunakan Laravel 9 dan bahasa pemrograman PHP 8 diimplementasikan. Proses implementasi mencakup dua aspek utama: *Endpoint* API dan Pembuatan *Controller*.

1. *Endpoint* API

Tabel 1. *Endpoint* API

***Endpoint* API**	**Deskripsi**
http://localhost:8000/api/taskid-all-ql	Mengambil data task ID yang pending dari BPJS dan menyimpannya ke dalam *database*.
http://localhost:8000/api/kodebooking-all-ql	Mengambil data kode booking yang pending dari BPJS dan menyimpannya ke dalam *database*.

Kedua *endpoint* ini dirancang untuk memberikan *respons* yang cepat dan akurat terhadap permintaan data.

2. Pembuatan *Controller*

Tabel 2. *Controller* yang digunakan

Controller	**Parameter**	**Fungsi**
PengirimanTaskIDController	data_pending_taskID_get($urlQL = 'QLJ')	Mengambil data task ID yang tertunda dari endpoint BPJS dan menyimpannya ke dalam *database*. Melakukan pengecekan untuk memastikan bahwa data yang disimpan relevan dan tidak ada duplikasi.
TambahAntrianController	data_pending_kodebooking_get($urlQL = 'QLJ')	Mengambil data kode booking yang tertunda dari endpoint BPJS dan menyimpannya ke dalam *database*. Melakukan pengecekan untuk memastikan bahwa data yang disimpan relevan dan tidak ada duplikasi.

Controller yang dibuat memiliki fungsi untuk menangani logika pengambilan data dari kedua *endpoint* tersebut. Metode dalam *controller* diimplementasikan untuk mengakses dan mengolah data yang diperlukan, memastikan informasi yang disajikan relevan dan tepat waktu.

3.4 Tahapan *Integration & Testing*

Pada tahap ini, seluruh komponen sistem, termasuk API, *controller*, dan *database*, diintegrasikan untuk memastikan fungsionalitas sinergis. Pengujian dilakukan dengan Metode B*lack box*, yang menilai fungsionalitas sistem berdasarkan input dan *output* yang diharapkan, tanpa mempertimbangkan cara kerja internal. yang dapat dilihat pada Tabel 3.

Tabel 3. Pengujian Menggunakan *Black Box*

Endpoint	**Deskripsi Pengujian**	**Input**	**Hasil yang Ditampilkan**
http://localhost:8000/api/kodebooking-all-ql	Mengambil data kodebooking	Kode booking yang *valid*	Status 200 OK, JSON dengan data kode booking
http://localhost:8000/api/taskid-all-ql	Mengambil data task ID	Task ID yang *valid*	Status 200 OK, JSON dengan data task ID
http://localhost:8000/api/kodebooking-all-ql	Mengambil data kodebooking	Kode booking yang tidak *valid*	Status 422 *Unprocessable Entity*, pesan *error* yang sesuai

Endpoint	**Deskripsi Pengujian**	**Input**	**Hasil yang Ditampilkan**
http://localhost:8000/api/taskid-all-ql	Mengambil data task ID	Task ID yang tidak *valid*	Status 422 *Unprocessable Entity*, pesan *error* yang sesuai

Dari 4 pengujian yang dilakukan, semua pengujian berhasil sesuai dengan ekspektasi. Keempat skenario pengujian dipilih untuk memastikan bahwa sistem dapat menangani baik skenario input yang *valid* maupun tidak *valid*, sehingga mencakup berbagai kondisi yang mungkin terjadi saat digunakan secara operasional. Pengujian dengan input yang *valid* memastikan bahwa data dapat diambil dengan benar, sementara pengujian dengan input yang tidak *valid* memeriksa apakah sistem dapat menampilkan pesan error yang sesuai dan menjaga keandalannya.Oleh karena itu, kita dapat menghitung tingkat keberhasilan sebagai berikut:

$$\textbf{Tingkat Keberhasilan} = 4/4 \times 100\% = 100\%$$

Dengan tingkat keberhasilan 100%, pengujian menunjukkan bahwa sistem sudah berjalan dengan sangat baik, memenuhi standar implementasi, dan siap digunakan secara operasional.

3.5 Tahapan *Operation & Maintenance*

Setelah sistem berhasil diimplementasikan dan diuji, tahap operasi dan pemeliharaan dimulai untuk memastikan sistem tetap berfungsi dengan baik dan dapat beradaptasi dengan kebutuhan yang berubah. Pemeliharaan dilakukan secara rutin oleh tim IT untuk memperbaiki *bug*, melakukan pembaruan perangkat lunak, dan menjaga kinerja semua komponen. Selain itu, sistem diperbarui dan ditingkatkan sesuai dengan perkembangan teknologi, penambahan fitur baru, dan penyesuaian terhadap regulasi yang berlaku. Monitoring dan evaluasi juga dilakukan secara berkelanjutan dengan menganalisis data penggunaan dan umpan balik untuk mengidentifikasi area yang memerlukan perbaikan.

4. KESIMPULAN

Pengembangan sistem *bridging* BPJS di RSU XYZ menggunakan Metode *Waterfall* telah berhasil meningkatkan stabilitas operasional dan pengelolaan data pasien. Dengan pengujian *Black box* yang menunjukkan tingkat keberhasilan 100%, sistem ini siap digunakan secara operasional. Kontribusi penelitian ini meliputi peningkatan efisiensi layanan kesehatan melalui pemisahan sistem *bridging* BPJS dari SIMRS, yang mengurangi risiko gangguan layanan. Selain itu, sistem ini memungkinkan pencatatan *log* aktivitas yang lebih baik, membantu analisis kinerja, dan meningkatkan akurasi pengelolaan data. Penelitian ini juga memberikan solusi untuk tantangan integrasi data antara SIMRS dan BPJS, serta berpotensi meningkatkan kualitas layanan kesehatan. Saran untuk penelitian selanjutnya adalah mengeksplorasi teknologi baru dan integrasi dengan sistem kesehatan lainnya.

5. DAFTAR RUJUKAN

[1] E. P. Putri, I. N. Syakira, M. F. Salim and F. M. Janah, "Analisis Kesiapan Bridging Simrs Dan V-Claim Di Rumah Sakit Pratama Kota Yogyakarta," *Jurnal Informasi Kesehatan Indonesia*, vol. IX, no. 1, pp. 47-58, 2023.

[2] M. Kurniawan and A. Harjoko, "Implementasi Bridging System Aplikasi SIKDA Generik dengan P-Care BPJS Kesehatan di kabupaten Lamongan," Universitas Gadjah Mada, Journal of Information Systems for Public Health (JISPH), Yogyakarta, 2018.

[3] K. and M. Badrul, "Penerapan Metode waterfall untuk Perancangan Sistem Informasi Inventory Pada Toko Keramik Bintang Terang," *Jurnal Pengembangan Riset & Observasi Sistem Komputer*, vol. VIII, no. 2, 2021.

[4] E. S. Dewi, E. A. Mesia Putri, D. . W. and J. T. Beng, "Perbandingan Antara Metode Waterfall Dan Metode RAD Dalam Pembuatan Aplikasi E-Rekrutmen Berbasis Website:

Studi Kasus PT XYZ," *Journal of Information Technology and Computer Science (INTECOMS)*, vol. VII, no. 4, 2024.

[5] A. Y. Rifanda, C. P. Nugroho, E. Nurfauziah, R. A. Lestari and A. Saifudin, "Pengembangan Aplikasi Inventori Barang Dengan Metode Waterfall," *JURIHUM : Jurnal Inovasi dan Humaniora*, vol. I, no. 1, pp. 165-172, 2023.

[6] F. Eawantini and I. Nurmawati, Implementation of mobile IMCI (Integrated Management of Childhood Illness) at Primary Health Care, Jember: Politeknik Negeri Jember.

[7] R. Rosaly and A. Prasetyo, "Pengertian Flowchart Beserta Fungsi dan Simbol-simbol Flowchart yang Paling Umum Digunakan," academia.edu, 2019.

[8] M. Fuad, "Perancangan Sistem Informasi Simpan Pinjam Pada Koperasi "KOPITAMA" Depok," *UG Journal*, vol. IX , no. 5, 2015.

[9] O. Veza and M. Ropianto , "Perancangan Sistem Informasi Inventory Data Barang Pada PT. Andalas Berlian Motors," *Jurnal Teknik Ibnu Sina (JT-IBSI)*, vol. II, no. 2, pp. 2541-2647, 2017.

[10] J. F. Widjojo, E. Rusdianto and F. K. Sari Dewi, "Pembangunan Sistem Manajemen Proyek pada Perusahaan Konstruksi Bangunan Berbasis Website," Universitas Atma Jaya, Yogyakarta, 2020.

[11] M. L. Ayusmara Latukolan, A. Arwan and M. T. Ananta, "Pengembangan Sistem Pemetaan Otomatis Entity Relationship Diagram Ke Dalam Database," Universitas Brawijaya, Malang, 2019.

[12] M. Permatasari and D. Wijayanto, "Rancang Bangun Sistem Informasi berbasis Website Studi kasus Koperasi XYZ," Universitas "Aisyiyah Yogyakarta, Yogyakarta, 2024.

[13] A. and A. Nurdin, "Rancang Bangun Sistem Informasi Administrasi Pendaftaran Kursus (Studi Kasus: Ghibrant English Course-Pandeglang)," *Jurnal Pengembangan Riset dan Observasi Sistem Komputer*, vol. X, no. 2, 2018.

[14] N. M. R. Anugrah, N. Nazira, N. A. Al-Qadr and N. , "Rancang Bangun Sistem Informasi Koperasi SImpan Pinjam," UIN Sultan Syarif Kasim, Riau, 2022.

[15] S. Sabarudin and S. Arif, "PEMBANGUNAN SISTEM INFORMASI PENYEWAAN ALAT CAMPING BERBASIS WEB DI POTONG KOMPAS," Universitas Nasional Pasim, Bandung, 2019.

[16] M. Ridwan, I. Fitri and B. , "Rancang Bangun Marketplace Berbasis Website menggunakan Metodologi Systems Development Life Cycle (SDLC) dengan Model Waterfall," *Jurnal Teknologi Informasi dan Komunikasi)*, vol. X, no. 2, pp. 173-184, 2021.

[17] S. D. Supriadi and B. Susanto, "Perancangan Sistem Informasi Penggajian Karyawan Dengan metode Waterfall," *Journal Computer Science*, vol. I, no. 1, 2022.

[18] W. Harjono and K. J. Tute, "Perancangan Sistem Informasi Perpustakaan Berbasis Web Menggunakkan Metode Waterfall," *SATESI: Jurnal Sains Teknologi dan Sistem Informasi," SATESI: Jurnal Sains Teknologi dan Sistem Informasi*, vol. II, no. 1, pp. 47-51, 2022.