STORAGE – Jurnal Ilmiah Teknik dan Ilmu Komputer, Vol. 5 No. 2, Mei 2026, 103 – 117
e-ISSN: 2828-5344 | DOI: 10.55123

OPTIMASI PERFORMA ANTARMUKA REACTIVE DENGAN MENERAPKAN SOCKET PROGRAMMING PADA SISTEM PEMBAYARAN SPP SMA DHARMAWANGSA

[1] Fathur Rahman, [2] Suendri
[1,2]Sistem Informasi, Sains dan Teknologi, Universitas Islam Negeri Sumatera Utara
[1]04fathur.rahman@gmail.com, [2]suendri@gmail.com

INFO ARTIKEL

Riwayat Artikel :
Diterima : 13 April 2026
Disetujui : 5 Mei 2026

Kata Kunci :
Antarmuka Pengguna Reaktif, Socket Programming, Sistem Pembayaran SPP, WebSocket, Sistem Informasi Sekolah.

ABSTRAK

Penelitian ini bertujuan untuk mengoptimalkan performa antarmuka pada sistem pembayaran SPP di SMA Dharmawangsa melalui penerapan Reactive User Interface dan Socket Programming. Permasalahan yang dihadapi adalah informasi pembayaran yang belum dapat diterima secara langsung oleh siswa, karena sistem sebelumnya hanya berfungsi sebagai media pencatatan oleh bendahara dan tidak mendukung pembaruan data secara real-time. Pendekatan yang digunakan berfokus pada integrasi komunikasi berbasis event melalui Socket Programming dengan pembaruan tampilan dinamis menggunakan Reactive User Interface. Pada sistem yang diusulkan, data pembayaran yang telah diproses oleh server secara otomatis dikirimkan ke klien melalui event, sehingga antarmuka dapat diperbarui tanpa proses refresh halaman.
Hasil pengujian menunjukkan bahwa waktu pengiriman data melalui socket berada pada kisaran 8 hingga 18 milidetik, sedangkan waktu render antarmuka berkisar antara 29 hingga 62 milidetik, sehingga keseluruhan proses pembaruan data berlangsung dalam waktu kurang dari satu detik. Hal ini membuktikan bahwa sistem mampu menyajikan informasi pembayaran secara cepat dan responsif. Dengan demikian, penerapan Socket Programming dan Reactive User Interface terbukti meningkatkan kecepatan distribusi data dan responsivitas antarmuka dibandingkan dengan mekanisme sebelumnya.

ARTICLE INFO

Article History :
Received : Apr 13, 2026
Accepted : May 5, 2026

Keywords:
Reactive User Interface, Socket Programming, WebSocket, School Tuition Payment System, Real-Time Information System

ABSTRACT

This study aims to optimize interface performance in the tuition payment system at SMA Dharmawangsa through the implementation of a Reactive User Interface and Socket Programming. The main issue addressed is the delay in delivering payment information to students, as the previous system only functioned as a recording tool and did not support real-time updates. The proposed approach integrates event-based communication using Socket Programming with dynamic interface updates through a Reactive User Interface. In the proposed system, payment data processed by the server is automatically transmitted to the client via events, allowing the interface to update without requiring a page refresh. The results show that data transmission time via socket ranges from 8 to 18 milliseconds, while the interface render time ranges from 29 to 62 milliseconds, resulting in an overall update process that occurs in less than one second. These findings demonstrate that the system is able to deliver payment information quickly and responsively. Therefore, the implementation of Socket Programming and Reactive User Interface effectively improves data distribution speed and interface responsiveness compared to the previous mechanism.

1. PENDAHULUAN

Perkembangan teknologi informasi mendorong digitalisasi berbagai sistem administrasi di lingkungan pendidikan guna meningkatkan efisiensi dan transparansi pengelolaan data. Salah satu aspek penting dalam administrasi sekolah adalah pengelolaan pembayaran SPP. Sistem pembayaran berbasis digital memberikan kemudahan dalam proses transaksi antara siswa, orang tua, dan pihak sekolah. Namun demikian, selain kemudahan transaksi, sistem juga perlu mampu menyajikan informasi pembayaran secara cepat, akurat, dan dapat diakses oleh pihak yang berkepentingan. Oleh karena itu, diperlukan sistem yang tidak hanya memfasilitasi transaksi pembayaran, tetapi juga mampu menghadirkan antarmuka yang responsif serta mendukung komunikasi data secara *real-time*.(Sandeep and Manjunath, 2022)

SMA Swasta Dharmawangsa Medan telah menerapkan pembayaran SPP melalui transfer bank atau mobile banking. Meskipun demikian, proses konfirmasi pembayaran masih dilakukan secara manual, di mana siswa harus menunjukkan bukti pembayaran kepada bendahara untuk kemudian dicatat ke dalam buku besar. Proses tersebut menyebabkan informasi pembayaran tidak dapat diketahui secara langsung oleh pihak terkait. Permasalahan ini menunjukkan bahwa sistem yang ada belum mampu menyajikan informasi pembayaran secara *real-time* dan efisien.(Vaishali V. Raje, 2024)

Untuk mengatasi permasalahan tersebut, penelitian ini mengusulkan pengembangan sistem pembayaran SPP berbasis web dengan menerapkan pendekatan *Reactive User Interface* dan *Socket Programming*. *Reactive User Interface* digunakan untuk menghasilkan tampilan yang mampu memperbarui informasi secara dinamis tanpa memerlukan pemuatan ulang halaman(Hoppe and Souza, 2023), sedangkan *Socket Programming* memungkinkan komunikasi data secara *real-time* antara *server* dan pengguna sistem.(Das and Ghosh, 2025) Dengan pendekatan ini, setiap perubahan data pembayaran dapat langsung ditampilkan kepada pengguna yang berkepentingan secara otomatis.

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk mengoptimalkan performa antarmuka pada sistem pembayaran SPP berbasis web di SMA Dharmawangsa dengan menerapkan *Reactive User Interface* dan Socket Programming. Penerapan kedua pendekatan ini diharapkan mampu meningkatkan kecepatan penyajian informasi pembayaran serta memungkinkan pembaruan data secara *real-time* kepada pengguna sistem. Dengan demikian, sistem yang dikembangkan tidak hanya berfungsi sebagai media pencatatan transaksi, tetapi juga sebagai sarana penyampaian informasi yang lebih responsif, efisien, dan terintegrasi.(Peddireddy, 2023)

2. METODE

Penelitian ini dilakukan di SMA Swasta Dharmawangsa Medan yang berlokasi di Kota Medan, Sumatera Utara. Pemilihan lokasi didasarkan pada kondisi sistem pembayaran SPP yang telah menggunakan metode transfer bank, namun proses pencatatan dan penyampaian informasi pembayaran masih bersifat terpusat pada bendahara dan belum terdistribusi secara langsung kepada pengguna lain, sehingga informasi pembayaran tidak dapat diperoleh secara langsung oleh siswa.

Penelitian ini menggunakan pendekatan kualitatif deskriptif yang berfokus pada analisis proses, mekanisme sistem, serta perilaku pembaruan data dalam penyajian informasi pembayaran. Pendekatan ini dipilih karena penelitian tidak hanya mengkaji kondisi sistem yang berjalan, tetapi juga mencakup perancangan dan evaluasi mekanisme pembaruan data berbasis event-driven melalui penerapan *Reactive User Interface* dan *Socket Programming*.(Yuliana and Purjumatin, 2024)

Subjek penelitian dalam studi ini adalah bendahara sekolah sebagai pengguna utama sistem, serta proses interaksi antara bendahara dan siswa dalam penyampaian informasi pembayaran.

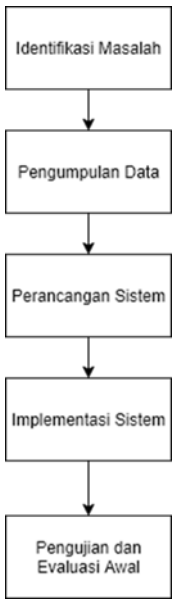
Teknik pengumpulan data dilakukan melalui wawancara, observasi, dan studi pustaka. Data primer diperoleh melalui wawancara semi terstruktur dengan bendahara sekolah serta observasi langsung terhadap proses pembayaran SPP yang berjalan. Data sekunder diperoleh dari dokumen pencatatan pembayaran serta referensi pustaka yang relevan. Data yang dikumpulkan digunakan untuk mengidentifikasi

permasalahan, menganalisis kebutuhan sistem, serta menjadi dasar dalam perancangan mekanisme sistem yang diusulkan.

Tahapan penelitian yang dilakukan meliputi:

1. **Identifikasi Masalah**
   Mengidentifikasi permasalahan pada sistem pembayaran SPP yang berjalan melalui observasi dan wawancara, khususnya pada proses pencatatan dan penyampaian informasi pembayaran.
2. **Pengumpulan Data**
   Mengumpulkan data yang berkaitan dengan alur sistem, kendala yang dihadapi, serta kebutuhan pengguna melalui observasi, wawancara, dan dokumentasi.
3. **Perancangan Sistem**
   Merancang sistem pembayaran berbasis web dengan mengintegrasikan *Reactive User Interface* pada sisi antarmuka dan *Socket Programming* sebagai mekanisme komunikasi data untuk mendukung pembaruan informasi secara real-time.(Wagle *et al.*, 2023)
4. **Implementasi Sistem**
   Mengimplementasi sistem menggunakan *React.js* pada sisi *frontend* dan *NestJS* pada sisi backend sebagai *API server* yang menangani pengolahan dan distribusi data.(M *et al.*, 2024)(Dhoke and Lokulwar, 2023)
5. **Pengujian Sistem**
   Pengujian dilakukan menggunakan skenario simulasi dengan melakukan input data pembayaran oleh bendahara, kemudian mengamati mekanisme pembaruan data pada antarmuka sistem setelah data diproses, termasuk bagaimana data didistribusikan dari *server* ke klien serta bagaimana tampilan diperbarui tanpa pemuatan ulang halaman. Evaluasi dilakukan dengan membandingkan mekanisme sistem sebelum dan sesudah pengembangan, serta mengamati perubahan tampilan dan alur distribusi data untuk menilai responsivitas sistem secara kualitatif dalam menyajikan informasi pembayaran. Hasil pengujian tersebut kemudian digunakan sebagai dasar dalam evaluasi mekanisme sistem yang disajikan pada bagian hasil dan pembahasan di.

Kerangka Tahapan penelitian ditunjukkan pada Gambar 1.



Gambar 1. Kerangka Tahapan Penelitian

3. HASIL DAN PEMBAHASAN

3.1 Hasil Observasi Sistem Pembayaran SPP

Berdasarkan hasil observasi yang dilakukan di SMA Dharmawangsa Medan, proses pembayaran SPP lebih sering dilakukan secara langsung di sekolah dibandingkan melalui transfer bank atau mobile banking. Siswa melakukan pembayaran kepada bendahara, kemudian bendahara mencatat data pembayaran tersebut ke dalam sistem yang tersedia serta mencatat ulang pada buku besar sebagai arsip. Dalam satu transaksi pembayaran terjadi dua kali proses pencatatan, yaitu ke dalam sistem dan buku besar, sehingga menunjukkan adanya duplikasi pencatatan pada dua media yang berbeda.

Data siswa dan struktur pembayaran SPP yang digunakan dalam proses pencatatan ditunjukkan pada Tabel 1.

Tabel 1. Data Siswa dan Struktur Pembayaran SPP

NIS	Nama	Kelas	Total SPP	Cicilan
19114	Andy Amirul Iksan Marbun	XI IPS 1	5.400.000	450.000
19115	Habibah Hafshah	XII MIPA 3	5.400.000	450.000
19116	Amira Fadhilah Lubis	XI IPS 2	5.400.000	450.000
19117	Rafa Raditya	XII MIPA 8	5.400.000	450.000
19118	Wafi Fadhlan Hawari	XI MIPA 4	5.400.000	450.000
19219	Raja Alif Fathanah S	XII IPS 1	5.400.000	450.000
19220	Arkan Aprilianda	X 1	5.400.000	450.000
19221	Dinda Syahfira Fitry	X 4	5.400.000	450.000
19122	Bayu Arditya	XII MIPA 6	5.400.000	450.000
19123	Chaisar Putra Adiguna Purba	XII MIPA 8	5.400.000	450.000

Tabel 1 menunjukkan data identitas siswa beserta informasi kelas, total biaya SPP, dan besaran cicilan per bulan yang harus dibayarkan. Data ini menjadi dasar dalam proses pencatatan pembayaran yang dilakukan oleh bendahara.

Berdasarkan hasil wawancara dengan bendahara, pencatatan pembayaran umumnya dilakukan secara langsung pada saat transaksi berlangsung. Namun, data yang telah dicatat hanya tersimpan pada sistem yang diakses oleh bendahara dan tidak secara otomatis terdistribusi kepada pengguna lain. Selain itu, tidak terdapat mekanisme yang memungkinkan data pembayaran diperbarui dan diakses secara bersamaan oleh pihak lain.

Data transaksi pembayaran siswa ditunjukkan pada Tabel 2.

Tabel 2. Data Transaksi Pembayaran SPP (Periode Januari 2026)

NIS	Nama	Periode	Tanggal	Nominal	Total Bayar	Keterangan
19114	Andy Amirul Iksan Marbun	2026-01	18/01/2026	450.000	450.000	Pembayaran Januari
19115	Habibah Hafshah	2026-01	19/02/2026	450.000	450.000	Pembayaran Januari
19116	Amira Fadhilah Lubis	2026-01	19/01/2026	450.000	450.000	Pembayaran Januari
19117	Rafa Raditya	2026-01	20/01/2026	450.000	450.000	Pembayaran Januari
19118	Wafi Fadhlan Hawari	2026-01	20/01/2026	450.000	450.000	Pembayaran Januari
19219	Raja Alif Fathanah S	2026-01	19/02/2026	450.000	450.000	Pembayaran Januari
19220	Arkan Aprilianda	2026-01	18/01/2026	450.000	450.000	Pembayaran Januari
19221	Dinda Syahfira Fitry	2026-01	20/01/2026	450.000	450.000	Pembayaran Januari
19122	Bayu Arditya	2026-01	19/01/2026	450.000	450.000	Pembayaran Januari
19123	Chaisar Putra Adiguna Purba	2026-01	18/01/2026	450.000	450.000	Pembayaran Januari

Tabel 2 menunjukkan data transaksi pembayaran yang dilakukan oleh siswa untuk periode tertentu. Berdasarkan tabel tersebut, pembayaran untuk periode yang sama dapat dilakukan pada waktu yang berbeda oleh siswa, namun tetap dicatat sesuai dengan periode pembayaran yang berlaku.

Pencatatan pembayaran pada buku besar yang dilakukan secara manual oleh bendahara ditunjukkan pada Tabel 3.

Tabel 3. Data Pencatatan Pembayaran SPP pada Buku Besar (Kelas XI MIPA 1)

NIS	Nama	P/L	Jul	Ags	Sep	Okt	Nov	Des	Jan	Feb	Mar	Apr	Mei	Jun
18624	ADITYA WIRADANU	L	28/7	28/8	06/10	29/10	8/12	8/12						
19112	ALMIRA SYAQILLA ANANDA RANGKUTI	P		8/8	10/9	21/10	21/11	8/12	14/1					
18692	ANNISA CHANDA KIRANA	P	17/7	6/8	12/9	1/10	10/11	25/11	12/1					
18756	AQSO SUDJUASMI ELRICO	L	24/7	20/11	20/11	20/11	20/11	8/12						
18660	ATAHIRA RAISYA LUBIS	P	28/7	6/10	6/10	6/10	2/12	2/12						
18592	ATHAARIQ FAWWAZ SETIAWAN	L	31/7	29/8	6/10	29/10	8/12	8/12						
18757	AURA DINDA	P	1/8	2/12	2/12	2/12	2/12	4/12						
19101	DAMAR MAHADIKA SYAHPUTRA	L		12/8	9/9	15/10	0/11	3/12						
18631	DHECA DWI KAEYLA	P	28/7	7/8	8/12	8/12	8/12	8/12						
18572	DINI ALYA KARTIKA	P	17/7	22/8	22/8	29/10	29/10	29/10						

Tabel 3 menunjukkan data pencatatan pembayaran dalam bentuk rekapitulasi per kelas, di mana setiap kolom merepresentasikan periode bulan pembayaran dan diisi dengan tanggal transaksi yang dilakukan oleh siswa. Data ini menggambarkan bagaimana pencatatan pembayaran dilakukan secara manual oleh bendahara.

Dari sisi siswa, meskipun pembayaran telah dilakukan dan dicatat, informasi status pembayaran tidak dapat diakses secara langsung melalui sistem. Siswa masih bergantung pada konfirmasi dari bendahara untuk mengetahui status pembayaran mereka. Kondisi ini menunjukkan bahwa sistem yang berjalan belum mampu menyediakan informasi secara real-time dan masih bergantung pada proses manual dalam penyampaian informasi.

Berdasarkan hasil observasi tersebut, permasalahan utama dalam sistem yang berjalan bukan terletak pada proses pencatatan transaksi, melainkan pada tidak adanya mekanisme pembaruan dan distribusi data secara otomatis. Hal ini menyebabkan informasi pembayaran

tidak dapat diakses secara cepat dan efisien oleh pihak yang membutuhkan.

3.2 Analisis dan Solusi

Berdasarkan hasil observasi yang ditunjukkan pada Tabel 1, Tabel 2, dan Tabel 3, permasalahan utama dalam sistem pembayaran SPP tidak terletak pada proses pencatatan transaksi, melainkan pada mekanisme pengelolaan dan distribusi data pembayaran. Meskipun pencatatan dilakukan secara langsung oleh bendahara, data pembayaran masih bersifat terpusat dan tidak dapat diakses secara *real-time* oleh pengguna lain.

Pada Tabel 2, terlihat bahwa pembayaran untuk periode yang sama dapat dilakukan pada waktu yang berbeda oleh siswa. Namun, informasi tersebut hanya tersimpan pada sistem yang diakses oleh bendahara dan tidak langsung tersedia bagi siswa. Sementara itu, pada Tabel 3, pencatatan pembayaran masih dilakukan secara manual dalam bentuk rekapitulasi pada buku besar, sehingga menunjukkan adanya duplikasi pencatatan serta keterbatasan dalam penyampaian informasi secara langsung.

Berdasarkan kondisi tersebut, permasalahan utama yang diidentifikasi meliputi:

1. Tidak adanya mekanisme distribusi data secara otomatis kepada pengguna lain
2. Informasi pembayaran tidak dapat diakses secara *real-time* oleh siswa
3. Terjadinya duplikasi pencatatan pada sistem dan buku besar

Untuk mengatasi permasalahan tersebut, solusi yang diusulkan dalam penelitian ini tidak hanya berfokus pada digitalisasi pencatatan, tetapi pada perancangan mekanisme pembaruan dan distribusi data secara real-time. Solusi ini diimplementasikan dengan membagi peran sistem ke dalam tiga komponen utama, yaitu antarmuka (*frontend*), *API server* (*backend*), dan mekanisme komunikasi data berbasis *event*.

1. *Reactive User Interface* (*Frontend - React*)

Pada sisi antarmuka, pendekatan *Reactive User Interface* diterapkan untuk mengatasi permasalahan tampilan yang tidak terbarui secara otomatis. Setiap data pembayaran dikontrol menggunakan state pada *React*, sehingga ketika terjadi perubahan data, komponen antarmuka akan melakukan *render* ulang secara otomatis tanpa memerlukan proses pemuatan ulang halaman.(Keshari *et al.*, 2023)

Dengan pendekatan ini, informasi pembayaran yang sebelumnya tidak dapat diakses secara langsung oleh siswa dapat ditampilkan secara dinamis pada antarmuka. Perubahan data yang terjadi setelah proses pencatatan oleh bendahara dapat langsung terlihat pada tabel maupun dashboard, sebagaimana ditunjukkan pada Gambar 4.

2. *API Server* (*Backend - NestJS*)

Pada sisi *backend*, sistem menggunakan *API* berbasis *NestJS* yang berfungsi sebagai pengelola utama data pembayaran. *API* ini menerima data dari antarmuka, melakukan proses validasi, serta menyimpan data ke dalam basis data.(Vadia *et al.*, 2024)

Selain itu, *API* juga berperan sebagai penghubung dalam proses distribusi data, di mana setiap data yang berhasil diproses akan diteruskan ke mekanisme komunikasi untuk didistribusikan kepada klien. Dengan demikian, *API* memastikan bahwa data yang dikirimkan telah terstruktur dan siap digunakan oleh sistem.

3. *Socket Programming* (*Real-time Communication*)

Untuk mengatasi permasalahan tidak adanya distribusi data secara langsung, digunakan mekanisme *Socket Programming* berbasis *event-driven*. Setelah data pembayaran berhasil disimpan melalui *API*, *server* akan mengirimkan *event* pembaruan data kepada klien yang terhubung melalui koneksi *socket*.(Nilnoree *et al.*, 2024)

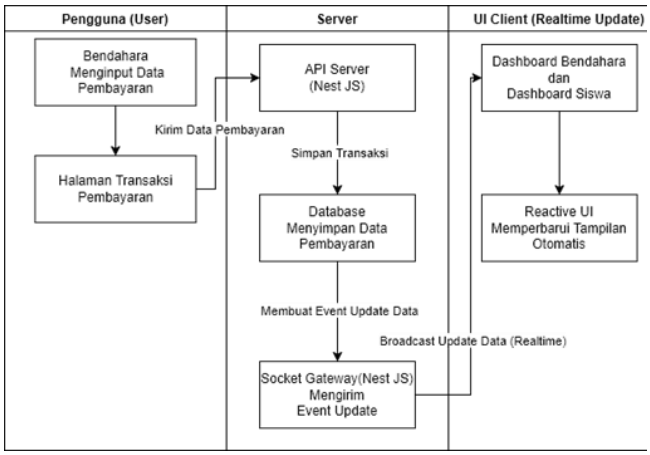
Mekanisme ini memungkinkan data dikirim secara langsung dari *server* ke klien tanpa menunggu permintaan ulang. Dengan demikian, informasi pembayaran yang sebelumnya hanya tersimpan pada sisi bendahara dapat langsung didistribusikan kepada pengguna lain secara *real-time*.(Hasnain and Ullah, 2023)

Pendekatan ini tidak hanya mengubah proses pencatatan menjadi digital, tetapi juga mengubah mekanisme sistem dari yang sebelumnya bersifat statis menjadi dinamis

berbasis *event-driven*, sehingga mampu mendukung penyajian informasi pembayaran secara lebih cepat dan responsif.

3.3 Mekanisme Pembaruan Data Sistem

Untuk mendukung penyajian informasi pembayaran berdasarkan permasalahan yang telah diidentifikasi pada subbab sebelumnya, sistem yang dirancang menggunakan mekanisme pembaruan data yang melibatkan tiga komponen utama, yaitu antarmuka (*React*), *API server* (*NestJS*), dan komunikasi data menggunakan *Socket Programming*. Mekanisme ini menggambarkan bagaimana data diproses dan didistribusikan secara *event-driven* setelah transaksi pembayaran dilakukan. Alur mekanisme ditunjukkan pada Gambar 2.



Gambar 2. Mekanisme *Reactive User Interface* dan *Socket Programming* pada Sistem Pembayaran SPP

Berdasarkan Gambar 2, proses dimulai ketika bendahara melakukan input data pembayaran melalui antarmuka berbasis React. Data yang diinput dikirim ke *API server* menggunakan metode HTTP untuk diproses dan disimpan ke dalam basis data. Pada tahap ini, *API server* berperan sebagai pengelola utama data yang menerima dan memvalidasi informasi pembayaran sebelum disimpan.

Setelah data berhasil disimpan, *server* secara otomatis memicu pengiriman *event* pembaruan data melalui mekanisme *Socket Programming* kepada klien yang terhubung. *Event* ini berisi data pembayaran terbaru yang dikirimkan sebagai indikator bahwa telah terjadi perubahan data pada sistem. Mekanisme ini memungkinkan distribusi data dilakukan secara langsung dari *server* ke klien tanpa menunggu permintaan ulang, sehingga mengatasi permasalahan keterbatasan distribusi informasi yang ditemukan pada hasil observasi.

Pada sisi klien, *event* yang diterima akan direspons oleh komponen antarmuka yang menggunakan pendekatan *Reactive User Interface*. Komponen yang terhubung dengan data pembayaran akan memperbarui state secara otomatis, sehingga tampilan seperti tabel data maupun informasi pembayaran dapat berubah secara langsung tanpa proses pemuatan ulang halaman.(Sidorov, 2024)

Dengan mekanisme tersebut, dapat ditunjukkan bahwa *Socket Programming* berperan dalam proses pengiriman data dari *server* ke klien secara *real-time*, sedangkan *Reactive User Interface* berperan dalam memperbarui tampilan berdasarkan perubahan data yang diterima. Integrasi kedua mekanisme ini tidak hanya mengatasi permasalahan distribusi data, tetapi juga meningkatkan responsivitas sistem dalam menyajikan informasi pembayaran sesuai dengan kebutuhan pengguna.

Hasil perancangan mekanisme ini menjadi dasar dalam proses evaluasi pada subbab berikutnya, khususnya dalam mengamati bagaimana pendekatan *event-driven* mempengaruhi kecepatan pembaruan data serta penyajian informasi secara real-time pada sistem.(Ahire, Joshi and Hajare, 2025)

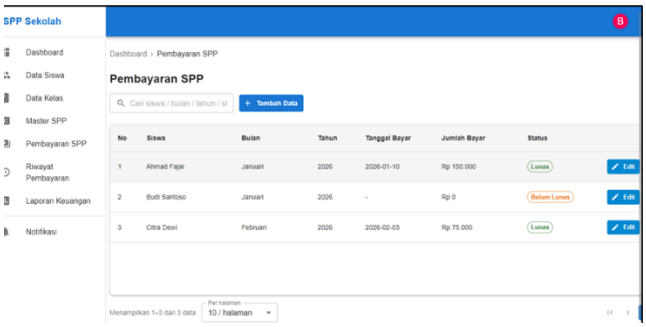
3.4 Implementasi Sistem

Sistem pembayaran SPP yang dikembangkan diimplementasikan dalam bentuk aplikasi berbasis web yang dapat difokuskan pada penggunaan oleh bendahara, sementara akses siswa direpresentasikan sebagai simulasi distribusi data dalam mekanisme sistem. Berbeda dengan sistem sebelumnya yang hanya digunakan oleh bendahara, sistem yang dikembangkan juga menyediakan akses bagi siswa untuk melihat informasi pembayaran secara langsung. Antarmuka sistem dirancang menggunakan pendekatan *Reactive User Interface* untuk memastikan bahwa setiap perubahan data dapat langsung ditampilkan secara dinamis pada halaman sistem. Implementasi ini bertujuan untuk meningkatkan kemudahan penggunaan serta mempercepat

akses terhadap informasi pembayaran.(Mufti Prasetiyo and Ariestia, 2023)

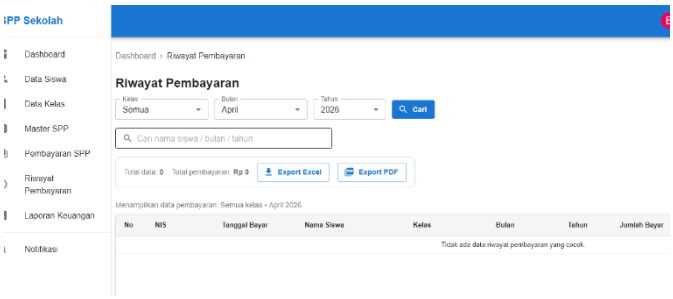
Pada sisi bendahara, sistem menyediakan fitur utama berupa input data pembayaran siswa. Bendahara dapat mencatat transaksi pembayaran melalui halaman input yang tersedia, yang kemudian akan diproses oleh sistem dan disimpan ke dalam basis data. Selain itu, sistem juga menampilkan data pembayaran dalam bentuk tabel yang diperbarui secara otomatis setelah transaksi dilakukan. Hal ini menunjukkan penerapan *Reactive User Interface* dalam memperbarui tampilan tanpa memerlukan proses pemuatan ulang halaman.

Pada sisi siswa, sistem menyediakan halaman informasi pembayaran yang memungkinkan pengguna untuk melihat status pembayaran secara langsung. Data pembayaran yang telah dicatat oleh bendahara akan dikirimkan dari *server* menggunakan *Socket Programming* dan ditampilkan pada halaman siswa secara *real-time*. Dengan demikian, siswa tidak perlu menunggu konfirmasi manual untuk mengetahui status pembayaran mereka.



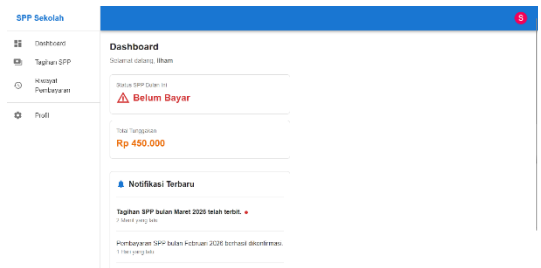
Gambar 3. Halaman Transaksi Pembayaran SPP

Gambar 3 menunjukkan tampilan halaman input pembayaran yang digunakan oleh bendahara untuk mencatat transaksi pembayaran siswa. Pada halaman ini, bendahara dapat memasukkan data pembayaran yang kemudian akan diproses oleh sistem dan disimpan ke dalam basis data. Setiap data yang diinput akan langsung dikirim ke *server* untuk diproses lebih lanjut, sehingga menjadi titik awal dalam mekanisme pembaruan data pada sistem.



Gambar 4. Halaman Data Pembayaran (*Reactive Update*)

Gambar 4 menunjukkan tampilan data pembayaran yang ditampilkan dalam bentuk tabel pada sisi bendahara. Halaman ini memanfaatkan pendekatan *Reactive User Interface*, di mana setiap perubahan data yang terjadi akan langsung diperbarui pada tampilan tanpa memerlukan proses pemuatan ulang halaman. Dengan mekanisme ini, bendahara dapat melihat hasil pencatatan pembayaran secara langsung setelah transaksi dilakukan.



Gambar 5. Menunjukkan simulasi tampilan informasi pembayaran pada sisi siswa yang menerima pembaruan data secara real-time

Gambar 5 menunjukkan tampilan dashboard pada sisi siswa yang menampilkan informasi status pembayaran. Data pembayaran yang telah dicatat oleh bendahara akan secara otomatis ditampilkan pada halaman ini melalui mekanisme komunikasi *real-time* menggunakan *Socket Programming*. Dengan demikian, siswa dapat mengetahui status pembayaran secara langsung tanpa perlu melakukan pemuatan ulang halaman atau menunggu konfirmasi manual dari bendahara.

3.5 Evaluasi Mekanisme Pembaruan Data Sistem

Evaluasi dilakukan untuk tidak hanya menjelaskan, tetapi juga membuktikan mekanisme pembaruan data yang diusulkan melalui beberapa pendekatan, yaitu perbandingan mekanisme sistem, pengamatan

komunikasi data, visualisasi perubahan tampilan antarmuka, serta pengujian respons pembaruan data berdasarkan skenario penggunaan. Pendekatan ini digunakan untuk memastikan bahwa penerapan *Reactive User Interface* dan *Socket Programming* tidak hanya bersifat konseptual, tetapi dapat diamati secara langsung melalui mekanisme yang berjalan pada sistem.

Pada sistem yang berjalan di sekolah, proses pembayaran SPP dilakukan melalui bendahara yang mencatat transaksi ke dalam sistem. Namun berdasarkan hasil observasi, pencatatan tersebut masih bersifat semi manual, karena selain diinput ke dalam sistem, data juga dicatat kembali pada media lain sebagai arsip. Selain itu, sistem yang digunakan belum menyediakan mekanisme pembaruan data secara langsung kepada pihak lain seperti siswa. Informasi pembayaran hanya dapat diketahui setelah dilakukan pengecekan atau konfirmasi secara manual, sehingga tidak mendukung penyajian data secara *real-time*.

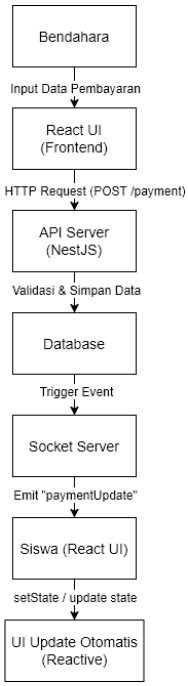
Pada sistem yang diusulkan, mekanisme pembaruan data dirancang melalui integrasi antara *API*, *Socket Programming*, dan *Reactive User Interface*. Ketika bendahara melakukan input data pembayaran, data dikirim ke *server* melalui *API* untuk diproses dan disimpan ke dalam basis data. Setelah data berhasil disimpan, *server* secara otomatis mengirimkan *event* pembaruan data kepada klien melalui koneksi *socket. Event* tersebut diterima oleh antarmuka dan direspons melalui pembaruan *state*, sehingga tampilan dapat berubah secara langsung tanpa proses pemuatan ulang halaman. Mekanisme ini menunjukkan perubahan dari pendekatan berbasis *request* menjadi *event-driven*, di mana server secara aktif mengirimkan data kepada klien.

Tabel 4. Perbandingan Mekanisme Pembaruan Data Sistem

Aspek	Sistem Sebelumnya	Sistem Usulan
Proses pencatatan	Semi-manual (input + arsip tambahan)	Terintegrasi dalam sistem
Akses informasi siswa	Tidak langsung	Tersedia melalui sistem
Pembaruan data	Tidak otomatis	Otomatis (real-time)
Mekanisme komunikasi	HTTP (request)	Event-driven (socket)
Kebutuhan refresh	Ya	Tidak
Respons perubahan data	Bergantung aksi pengguna	Langsung

Berdasarkan Tabel 4, sistem usulan menunjukkan perubahan mendasar pada mekanisme pembaruan data dibandingkan sistem sebelumnya. Pada sistem sebelumnya, proses pembaruan data masih bersifat terbatas pada sisi pencatatan, sehingga meskipun data telah dimasukkan ke dalam sistem, perubahan tersebut tidak langsung tercermin pada antarmuka pengguna. Hal ini menyebabkan informasi tidak dapat diperoleh secara langsung dan masih memerlukan tindakan tambahan dari pengguna.

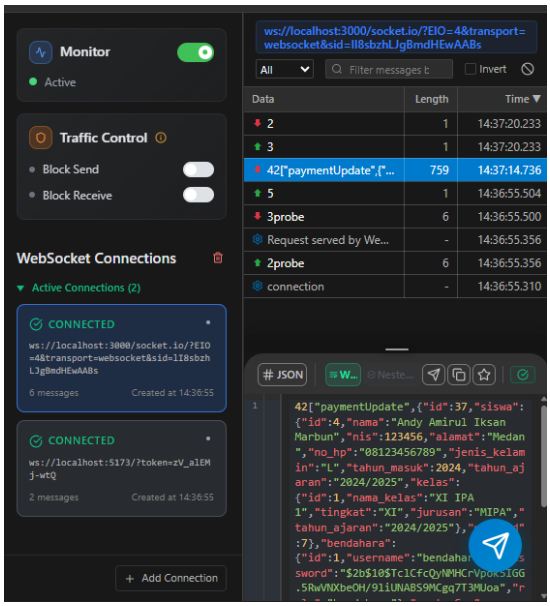
Sebaliknya, pada sistem usulan, mekanisme pembaruan data telah mengadopsi pendekatan *event-driven* melalui integrasi *Socket Programming* dan *Reactive User Interface*. Data yang telah diproses oleh server tidak hanya disimpan, tetapi juga langsung dikirimkan kepada klien melalui *socket*. Selanjutnya, *Reactive User Interface* secara otomatis merespons perubahan tersebut dengan memperbarui state pada antarmuka, sehingga tampilan dapat berubah secara langsung tanpa proses *refresh*. Perubahan ini menjadi dasar utama dalam peningkatan responsivitas sistem. Untuk mendukung evaluasi tersebut, dilakukan simulasi mekanisme pembaruan data yang divisualisasikan pada Gambar 6.



Gambar 6. Simulasi *Event* Pembaruan Data

Gambar 6 menunjukkan alur pembaruan data berbasis *event-driven* yang terjadi dalam sistem. Proses dimulai ketika bendahara melakukan input pembayaran melalui antarmuka. Data tersebut dikirim ke *server* melalui *API* untuk diproses dan disimpan ke dalam basis data. Setelah data berhasil disimpan, *server* memicu pengiriman *event paymentUpdate* melalui koneksi *socket* kepada klien yang terhubung. Klien yang menerima *event* tersebut kemudian memperbarui *state* pada antarmuka sehingga tampilan berubah secara otomatis tanpa proses pemuatan ulang halaman. Mekanisme ini menunjukkan bahwa pembaruan data tidak lagi bergantung pada permintaan dari klien, melainkan dikendalikan oleh *server* secara aktif.

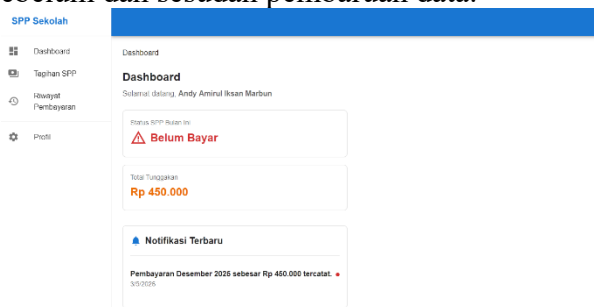
Untuk membuktikan bahwa sistem menggunakan komunikasi berbasis socket, dilakukan pengamatan terhadap koneksi *WebSocket* menggunakan bantuan *WebSocket DevTools* pada browser. Pengamatan ini bertujuan untuk melihat secara langsung proses pertukaran data antara server dan klien melalui mekanisme event.



Gambar 7. Monitoring Koneksi dan *Event WebSocket* menggunakan *WebSocket DevTools*

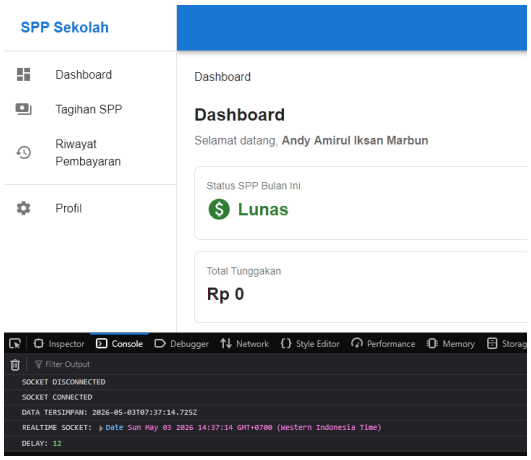
Gambar 7 menunjukkan adanya koneksi WebSocket aktif antara klien dan server dengan status *CONNECTED*. Selain itu, terlihat bahwa server mengirimkan *event paymentUpdate* yang berisi data pembayaran dalam format JSON kepada klien. Hal ini membuktikan bahwa sistem telah menggunakan mekanisme komunikasi berbasis socket, di mana server secara aktif mendistribusikan data kepada klien tanpa menunggu permintaan, sehingga mendukung pembaruan data secara *real-time*.

Selanjutnya, dilakukan pengamatan terhadap perubahan tampilan antarmuka sebelum dan sesudah pembaruan data.



Gambar 8. Tampilan Dashboard Siswa Sebelum Pembaruan Data

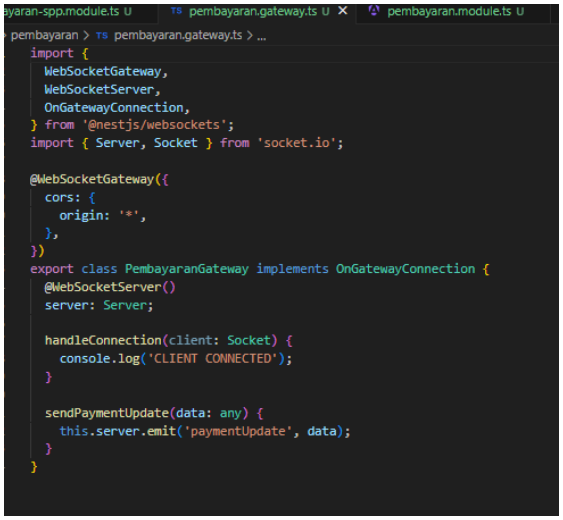
Gambar 8 menunjukkan kondisi sebelum data pembayaran ditambahkan, di mana status pembayaran masih belum berubah dan informasi yang ditampilkan belum diperbarui.



Gambar 9. Tampilan Dashboard Siswa Setelah Pembaruan Data

Gambar 9 menunjukkan kondisi setelah bendahara melakukan input data pembayaran. Informasi pembayaran langsung diperbarui pada antarmuka tanpa proses refresh halaman. Perubahan ini menunjukkan bahwa sistem mampu merespons data secara langsung melalui mekanisme *Reactive User Interface*.

Selain itu, implementasi teknis dari mekanisme pembaruan data ditunjukkan melalui potongan kode program pada sisi *server* dan klien.



```
import {
  WebSocketGateway,
  WebSocketServer,
  OnGatewayConnection,
} from '@nestjs/websockets';
import { Server, Socket } from 'socket.io';

@WebSocketGateway({
  cors: {
    origin: '*',
  },
})
export class PembayaranGateway implements OnGatewayConnection {
  @WebSocketServer()
  server: Server;

  handleConnection(client: Socket) {
    console.log('CLIENT CONNECTED');
  }

  sendPaymentUpdate(data: any) {
    this.server.emit('paymentUpdate', data);
  }
}
```

Gambar 10. Implementasi *Socket Programming* pada *Server*

Gambar 9 menunjukkan implementasi *Socket Programming* pada sisi *server* menggunakan *WebSocket Gateway* pada *NestJS*. Setelah data pembayaran berhasil diproses dan disimpan ke dalam basis data, *server* memanggil fungsi *sendPaymentUpdate* untuk mengirimkan *event paymentUpdate* kepada seluruh klien yang terhubung. Mekanisme ini menunjukkan bahwa komunikasi tidak lagi bergantung pada permintaan berulang dari klien, melainkan *server* secara proaktif mendistribusikan data melalui koneksi yang bersifat persisten. Pendekatan ini mendukung arsitektur *event-driven* yang lebih efisien dibandingkan pola *request-response* pada HTTP.

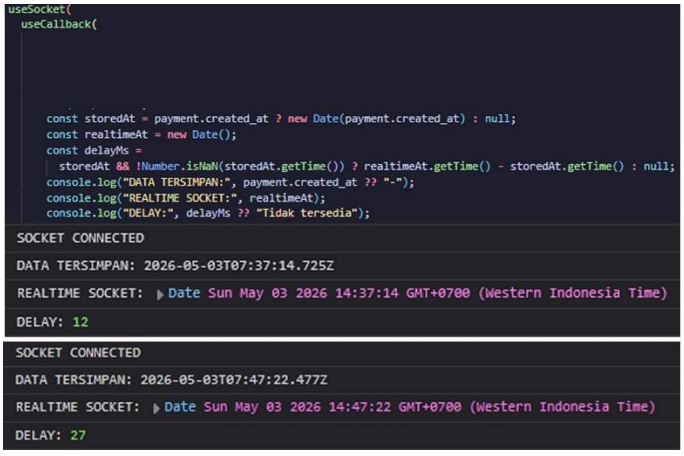
```
const paymentNis = String(payment.siswa?.nis ?? "");
if (String(me.nis) !== paymentNis && Number(payment.siswa_id) !== Number(me.id)) return;

if (payment.periode === periodeSekarang) {
  setStatusSppBulanIni(payment.status === "Lunas" ? "Lunas" : "Belum Bayar");
}
if (payment.status === "Lunas") {
  setTotalTunggakan((prev) => Math.max(prev - Number(payment.jumlah_bayar ?? 0), 0));
}
setNotifikasiTerbaru((prev) =>
  [
    {
      id: payment.id,
      pesan: `Pembayaran ${getMonthName(payment.bulan)} ${payment.siswa?.nama ?? ""}
      ${payment.tahun} sebesar Rp ${Number(payment.jumlah_bayar ?? 0).toLocaleString("id-ID")} tercatat.`,
      waktu: formatDate(payment.tanggal_bayar ?? payment.created_at),
      isRead: false,
    },
    ...prev,
  ].slice(0, 3),
);
```

Gambar 11. Implementasi Reactive User Interface pada Client

Gambar 10 menunjukkan penggunaan *state management* pada React yang secara otomatis memperbarui tampilan ketika event diterima, seperti penggunaan fungsi *setState* seperti *setStatusSppBulanIni*, *setTotalTunggakan*, dan *setNotifikasiTerbaru*. Hal ini membuktikan bahwa antarmuka bersifat reaktif terhadap perubahan data.

Untuk mengukur waktu pembaruan data pada sistem, dilakukan pengujian melalui *console browser* dengan menampilkan informasi waktu saat data tersimpan di server, waktu diterima oleh klien melalui socket, serta waktu yang dibutuhkan antarmuka untuk melakukan pembaruan tampilan.



```
useSocket(
  useCallback(

    const storedAt = payment.created_at ? new Date(payment.created_at) : null;
    const realtimeAt = new Date();
    const delayMs =
      storedAt && !Number.isNaN(storedAt.getTime()) ? realtimeAt.getTime() - storedAt.getTime() : null;
    console.log("DATA TERSIMPAN:", payment.created_at ?? "-");
    console.log("REALTIME SOCKET:", realtimeAt);
    console.log("DELAY:", delayMs ?? "Tidak tersedia");
```

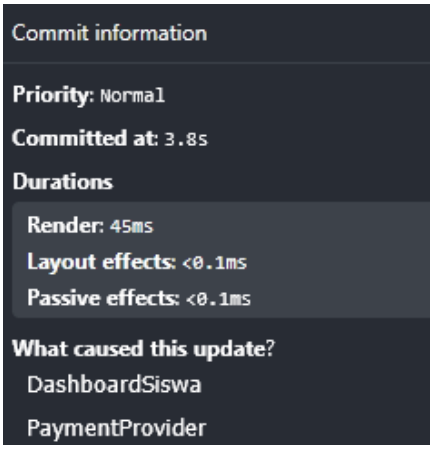
```
SOCKET CONNECTED
DATA TERSIMPAN: 2026-05-03T07:37:14.725Z
REALTIME SOCKET: Date Sun May 03 2026 14:37:14 GMT+0700 (Western Indonesia Time)
DELAY: 12
SOCKET CONNECTED
DATA TERSIMPAN: 2026-05-03T07:47:22.477Z
REALTIME SOCKET: Date Sun May 03 2026 14:47:22 GMT+0700 (Western Indonesia Time)
DELAY: 27
```

Gambar 12. Pengujian Waktu Pembaruan Data melalui *Console Browser*

Gambar 12 menampilkan potongan kode pengujian serta hasil keluaran pada *console*

browser yang menunjukkan waktu penyimpanan data (data tersimpan), waktu penerimaan data melalui *socket* (*realtime socket*), serta selisih waktu (*delay*) yang dihasilkan.

Untuk melengkapi pengujian waktu pembaruan data yang diamati melalui *console browser*, dilakukan pengujian tambahan pada sisi antarmuka guna mengukur waktu yang dibutuhkan sistem dalam melakukan proses render setelah data diterima melalui socket.



Gambar 13. Pengujian Waktu Render Antarmuka Menggunakan *React DevTools*

Gambar 13 menampilkan hasil pengukuran kinerja antarmuka menggunakan *React DevTools Profiler*. Pada pengujian ini, ditampilkan informasi durasi proses render (*render duration*), *layout effects*, serta *passive effects* yang terjadi pada komponen antarmuka. Selain itu, terlihat pula informasi *"what caused this update"* yang menunjukkan bahwa pembaruan tampilan dipicu oleh komponen DashboardSiswa sebagai respons terhadap event *paymentUpdate* Pengujian dilakukan sebanyak beberapa kali percobaan dengan melakukan input data pembayaran melalui sistem bendahara, kemudian diamati waktu pengiriman data melalui socket serta waktu pembaruan tampilan pada sisi klien. Hasil pengujian tersebut disajikan pada Tabel 5.

Tabel 5. Hasil Pengujian Waktu Pembaruan Data Real-Time

No	Waktu Simpan (Server)	Waktu Diterima (Client)	Delay Socket (ms)	Render UI (ms)
1	15:14:49	15:14:49	18	52
2	15:35:46	15:35:46	8	62
3	15:39:02	15:39:02	13	44
4	15:40:55	15:40:55	13	52
5	15:42:29	15:42:29	13	47
6	15:44:20	15:44:20	11	51
7	15:45:47	15:45:47	10	32
8	15:47:02	15:47:02	11	56
9	15:48:06	15:48:06	10	29
10	15:49:08	15:49:09	10	40

Berdasarkan hasil pengujian pada Tabel 5, terlihat bahwa waktu pengiriman data dari server ke klien melalui mekanisme *Socket Programming* berada pada rentang 8 ms hingga 18 ms, yang menunjukkan bahwa proses distribusi data berlangsung sangat cepat dalam skala milidetik. Hal ini membuktikan bahwa data pembayaran yang telah disimpan di *server* dapat diterima oleh klien hampir secara langsung tanpa keterlambatan yang signifikan.

Selain itu, waktu render antarmuka yang diukur menggunakan *React DevTools* menunjukkan nilai berkisar antara 29 ms hingga 62 ms, yang menandakan bahwa antarmuka mampu merespons perubahan data dengan cepat setelah data diterima. Dengan demikian, total waktu respons sistem, mulai dari data tersimpan hingga ditampilkan pada antarmuka, tetap berada jauh di bawah 1 detik.

Konsistensi hasil pengujian pada beberapa percobaan menunjukkan bahwa sistem memiliki performa yang stabil dalam menangani pembaruan data secara *real-time*. Hal ini mengindikasikan bahwa integrasi antara *Socket Programming* dan *Reactive User Interface* mampu menghasilkan mekanisme pembaruan data yang cepat, responsif, dan sesuai dengan kebutuhan sistem pembayaran SPP.

Tabel 6. Perbandingan Mekanisme Komunikasi Data (HTTP vs *Socket*)

Aspek	HTTP (Request-Response)	Socket (Event-driven)
Pola komunikasi	Client request	Server push
Frekuensi komunikasi	Berdasarkan aksi user	Berdasarkan event
Efisiensi	Kurang efisien	Lebih efisien
Pembaruan data	Tidak langsung	Real-time
Ketergantungan refresh	Ya	Tidak

Berdasarkan Tabel 6, perbedaan utama antara komunikasi berbasis HTTP dan *Socket* terletak pada pola distribusi data serta efisiensi pembaruan informasi. Pada HTTP, komunikasi bersifat *request-response*, di mana klien harus secara aktif meminta data untuk memperoleh pembaruan. Pendekatan ini berpotensi menimbulkan keterlambatan serta meningkatkan beban server akibat permintaan berulang (*polling*).

Sebaliknya, pada *Socket Programming*, komunikasi bersifat *event-driven* dengan mekanisme *server push*. Hal ini terbukti dari hasil pengujian pada Tabel 5, di mana data dapat dikirim dari server ke klien dalam waktu milidetik tanpa memerlukan permintaan ulang. Integrasi dengan *Reactive User Interface* memungkinkan data yang diterima langsung direspons oleh antarmuka melalui pembaruan state secara otomatis, sehingga perubahan dapat ditampilkan secara *real-time*.

Tabel 7. Evaluasi Respons Pembaruan Data Sistem

Skenario	Sistem Sebelumnya	Sistem Usulan
Input pembayaran	Disimpan oleh bendahara	Disimpan ke database
Penyampaian informasi	Melalui pengecekan manual	Otomatis melalui sistem
Pembaruan tampilan	Tidak langsung	Otomatis
Waktu respons	Tidak terukur	berada pada kisaran milidetik (<1 detik)
Ketersediaan data	Terbatas	Real-time

Berdasarkan Tabel 7, sistem usulan menunjukkan peningkatan signifikan dalam respons pembaruan data dibandingkan sistem sebelumnya. Pada sistem sebelumnya, meskipun data telah dicatat ke dalam sistem, perubahan tersebut tidak langsung terlihat pada antarmuka pengguna dan masih memerlukan tindakan tambahan seperti pengecekan ulang atau penyegaran halaman.

Sebaliknya, pada sistem usulan, setiap perubahan data langsung dikirimkan oleh *server* melalui *Socket Programming* dan direspons oleh *Reactive User Interface* secara otomatis. Hal ini diperkuat oleh hasil pengujian pada Tabel 5 yang menunjukkan bahwa waktu respons sistem berada pada kisaran milidetik, sehingga pengguna dapat langsung melihat perubahan data tanpa keterlambatan yang berarti.

Untuk memperkuat hasil evaluasi tersebut, dilakukan analisis lebih lanjut terhadap kontribusi teknologi yang digunakan dalam sistem, yang dirangkum pada Tabel 8.

Tabel 8. Analisis Kontribusi Teknologi terhadap Performa Sistem

Aspek	Socket Programming	Reactive User Interface
Peran utama	Mengirim data secara real-time dari server ke klien	Menampilkan perubahan data secara langsung pada antarmuka
Mekanisme kerja	Server push (event-driven)	State-based rendering (otomatis re-render)
Ketergantungan request	Tidak membutuhkan request berulang	Tidak membutuhkan refresh halaman
Dampak ke performa	Mengurangi beban request (menghindari polling berat)	Mengurangi beban rendering manual
Respons sistem	Data dikirim dalam milidetik (<1 detik)	Data langsung ditampilkan tanpa delay visual
Konsistensi data	Data tersinkron antar klien	Tampilan selalu mengikuti state terbaru
Pengalaman pengguna	Informasi diterima lebih cepat	Tampilan terasa instan dan responsif

Berdasarkan Tabel 8, dapat dilihat bahwa *Socket Programming* dan *Reactive User Interface* memiliki kontribusi yang saling melengkapi dalam meningkatkan performa sistem. *Socket Programming* berperan dalam mendistribusikan data secara cepat melalui mekanisme *server push*, yang terbukti mampu mengirim data dalam waktu milidetik berdasarkan hasil pengujian.

Di sisi lain, *Reactive User Interface* berperan dalam merespons data yang diterima dengan melakukan pembaruan *state* secara otomatis, sehingga perubahan dapat langsung ditampilkan pada antarmuka dengan waktu render yang relatif rendah. Hasil pengujian menunjukkan bahwa waktu render antarmuka berada pada kisaran puluhan milidetik, yang menandakan bahwa sistem mampu menampilkan perubahan data secara cepat setelah diterima.

Kombinasi kedua teknologi tersebut menghasilkan sistem yang tidak hanya cepat dalam distribusi data, tetapi juga responsif dalam penyajian informasi. Hal ini menunjukkan bahwa optimasi performa yang dilakukan mencakup dua aspek utama, yaitu kecepatan komunikasi data dan kecepatan respons antarmuka.

Selain itu, sistem sebelumnya berpotensi menggunakan pendekatan polling, yaitu klien secara berkala mengirim permintaan untuk mengecek pembaruan data. Pendekatan ini dapat meningkatkan beban server dan menyebabkan ketidakefisienan dalam komunikasi data.

Dengan penerapan *Socket Programming*, mekanisme tersebut digantikan dengan pendekatan *event-driven*, di mana *server* hanya mengirimkan data ketika terjadi perubahan. Hal ini terbukti lebih efisien karena mengurangi lalu lintas komunikasi yang tidak diperlukan serta mempercepat distribusi data ke klien.

Berdasarkan seluruh hasil evaluasi yang telah dilakukan, dapat disimpulkan bahwa penerapan *Socket Programming* dan *Reactive User Interface* mampu meningkatkan performa sistem secara signifikan. Hasil pengujian menunjukkan bahwa waktu respons pembaruan data berada pada kisaran milidetik, sehingga informasi dapat diterima dan ditampilkan pada antarmuka secara hampir instan. Dengan demikian, sistem yang diusulkan tidak hanya mengatasi keterbatasan sistem sebelumnya, tetapi juga memberikan peningkatan dalam efisiensi komunikasi data serta kualitas interaksi pengguna.

4. PENUTUP

4.1. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan evaluasi sistem pembayaran SPP berbasis web pada SMA Dharmawangsa, dapat disimpulkan bahwa penerapan *Socket Programming* dan *Reactive User Interface* mampu meningkatkan performa sistem secara signifikan, khususnya dalam mekanisme pembaruan data. Sistem yang sebelumnya hanya berfungsi sebagai media pencatatan kini berkembang menjadi sistem yang mampu mendistribusikan informasi secara langsung melalui pendekatan *event-driven. Socket Programming* memungkinkan *server* untuk mengirimkan data kepada klien secara otomatis tanpa memerlukan permintaan berulang, sehingga proses komunikasi menjadi lebih efisien dan tidak bergantung pada aksi pengguna.

Di sisi lain, *Reactive User Interface* berperan dalam memastikan bahwa setiap data yang diterima dapat langsung direspons melalui pembaruan *state*, sehingga perubahan dapat ditampilkan secara otomatis pada antarmuka tanpa proses refresh halaman. Hasil pengujian menunjukkan bahwa waktu pengiriman data melalui *socket* berada pada kisaran 8 hingga 18 milidetik, sedangkan waktu render antarmuka berkisar antara 29 hingga 62 milidetik. Dengan demikian, keseluruhan proses pembaruan data, mulai dari penyimpanan hingga ditampilkan pada antarmuka, berlangsung dalam waktu yang sangat singkat dan tetap berada di bawah satu detik. Integrasi kedua teknologi tersebut membuktikan bahwa sistem mampu meningkatkan kecepatan distribusi data, responsivitas tampilan, serta konsistensi informasi yang diterima oleh pengguna secara *real-time*.

4.2. Saran

Untuk pengembangan lebih lanjut, sistem dapat ditingkatkan dengan menambahkan fitur notifikasi yang lebih interaktif, seperti push notification, sehingga pengguna dapat menerima informasi pembaruan secara langsung tanpa harus membuka sistem. Selain itu, pengelolaan data pada sisi *backend* perlu terus dioptimalkan, khususnya dalam pengaturan data historis seperti kelas dan master SPP, agar tetap terstruktur dan efisien seiring dengan pertambahan data setiap periode.

Di samping itu, sistem dapat dikembangkan dengan menambahkan fitur pencarian dan penyaringan data yang lebih optimal pada antarmuka untuk mendukung aktivitas pengelolaan data oleh bendahara. Pengujian pada skala yang lebih besar juga perlu dilakukan untuk mengevaluasi performa sistem dalam kondisi beban tinggi. Pengembangan lebih lanjut ke arah perluasan akses pengguna di lingkungan sekolah diharapkan dapat meningkatkan manfaat sistem secara keseluruhan.

5. DAFTAR PUSTAKA

Ahire, P., Joshi, A. and Hajare, S. (2025) "Client-Server Communication Using Socket Programming," *IJCRT*, 13(4), pp. 52–59. Available at: https://doi.org/10.13140/RG.2.2.26515.95526.

Das, A.K. and Ghosh, S. (2025) "Real-Time Virtual Classroom Platform: Integrating React, Firebase, WebRTC, and Socket.io," *2025 8th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech)*. IEEE, pp. 1–4. Available at: https://doi.org/10.1109/IEMENTech65115.2025.10959621.

Dhoke, P. and Lokulwar, P. (2023) "Evaluating the Impact of No-Code/Low-Code Backend Services on API Development and Implementation: A Case Study Approach," *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, pp. 1–5. Available at: https://doi.org/10.1109/ICCCNT56998.2023.10306945.

Hasnain, M. and Ullah, S. (2023) "Learning and programming challenges of React.js open-source framework," *Library Hi Tech News* [Preprint]. Available at: https://doi.org/10.1108/LHTN-05-2023-0088.

Hoppe, P.H.B. and Souza, V.E.S. (2023) "Support for Single Page Application Frameworks on FrameWeb," *Proceedings of the 29th Brazilian Symposium on Multimedia and the Web*. New York, NY, USA: ACM, pp. 260–268. Available at: https://doi.org/10.1145/3617023.3617059.

Keshari, P. *et al.* (2023) "Web Development Using ReactJS," *2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*. IEEE, pp. 1571–1575. Available at: https://doi.org/10.1109/ICAC3N60023.2023.10541743.

M, K.P. *et al.* (2024) "CRUD Application Using ReactJS Hooks," *EAI Endorsed Transactions on Internet of Things*, 10. Available at: https://doi.org/10.4108/eetiot.5298.

Mufti Prasetiyo, S. and Ariestia, F.A. (2023) "OKTAL : Jurnal Ilmu Komputer dan Science Mengenal User Interface dan User Experience dalam Dunia Desain dan Teknologi," *Jurnal Ilmu Komputer dan Science*, 2(10), pp. 2671–2679. Available at: https://journal.mediapublikasi.id/index.php/oktal.

Nilnoree, S. *et al.* (2024) "Enhancing Wireless Sensor Network in Structural Health Monitoring through TCP/IP Socket Programming-Based Mimic Broadcasting: Experimental Validation," *Applied Sciences*, 14(8), p. 3494. Available at: https://doi.org/10.3390/app14083494.

Peddireddy, K. (2023) "Streamlining Enterprise Data Processing, Reporting and Realtime Alerting using Apache Kafka," *2023 11th International Symposium on Digital Forensics and Security (ISDFS)*. IEEE, pp. 1–4. Available at: https://doi.org/10.1109/ISDFS58141.2023.10131800.

Sandeep, K. V and Manjunath, T.C. (2022) "A Novel Mechanism for Design and Implementation of Confidentiality in Data for the Internet of Things with DES Technique," *2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp. 106–110. Available at: https://doi.org/10.1109/I-SMAC55078.2022.9987268.

Sidorov, D. (2024) "ANALYZING VIRTUAL DOM PERFORMANCE IN FRONT-END FRAMEWORKS," *Наука і техніка сьогодні* [Preprint], (8(36)). Available at: https://doi.org/10.52058/2786-6025-2024-8(36)-840-858.

Vadia, K. *et al.* (2024) "Bug Testing Automation with Playwright and a Backend API," *2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. IEEE, pp. 1867–1870. Available at: https://doi.org/10.1109/I-SMAC61858.2024.10714759.

Vaishali V. Raje, E. al. (2024) "Realtime Anomaly Detection in Healthcare IoT: A Machine Learning-Driven Security

Framework," *Journal of Electrical Systems*, 19(3), pp. 192–202. Available at: https://doi.org/10.52783/jes.700.

Wagle, R. *et al.* (2023) "Optimal power flow-based reactive power control in smart distribution network using real-time cyber-physical co-simulation framework," *IET Generation, Transmission & Distribution*, 17(20), pp. 4489–4502. Available at: https://doi.org/10.1049/gtd2.12786.

Yuliana, D. and Purjumatin, P. (2024) "Optimalisasi Pembayaran SPP Berbasis Web Dengan Framework Codeigniter Pada SMK Terpadu Bina Insan Mandiri," *INTECOMS: Journal of Information Technology and Computer Science*, 7(4), pp. 1315–1322. Available at: https://doi.org/10.31539/intecoms.v7i4.10875.