IMPLEMENTATION OF CONCURRENCY CONTROL TO PREVENT RACE CONDITION IN A WEB-BASED BILLIARD TABLE RESERVATION SYSTEM

IMPLEMENTASI CONCURRENCY CONTROL UNTUK MENCEGAH RACE CONDITION PADA SISTEM PEMESANAN MEJA BILLIARD BERBASIS WEBSITE

Mohammad Luthfi Hermawan[1], Wildan Suharso[2]
[1,2]Universitas Muhammadiyah Malang, Indonesia
Jalan Raya Tlogomas No 246, Kota Malang, Jawa Timur, Indonesia
Mohammadluthfihemawan24@webmail.umm.ac.id[1], wsuharso@umm.ac.id[2]

Abstract *- The development of information technology has driven the digitalization of various services, including web-based billiard table reservation systems. However, web systems that operate in real time are prone to race conditions when multiple users attempt to book the same table simultaneously, potentially leading to double booking. This study aims to implement a Concurrency Control mechanism using the Firebase Transaction feature to prevent such booking conflicts. The research method adopts a Research and Development (R&D) approach with the ADDIE model, which consists of the stages of Analysis, Design, Development, Implementation, and Evaluation. Furthermore, testing was conducted through pre-test and post-test simulations across 10 trials with concurrent users ranging from 2 to 11 individuals. In the pre-test stage, all users were able to successfully book the same resource simultaneously, resulting in 100% double booking across all trials. In the post-test stage, after implementing the Concurrency Control mechanism using Firebase Transaction, only one request was accepted out of the same 10 trials, while all other requests were automatically rejected, resulting in 0% double booking. These findings demonstrate that the applied concurrency control mechanism is effective in maintaining data consistency and preventing race conditions in the web-based billiard table reservation system.*

Keywords *- Concurrency Control, Race Condition, Firebase Transaction, Real-Time Database.*

Abstrak - Perkembangan teknologi informasi telah mendorong digitalisasi berbagai layanan, termasuk sistem pemesanan meja billiard berbasis *web*. Namun, sistem *website* yang berjalan secara *realtime* rawan mengalami *race condition* ketika beberapa pengguna melakukan pemesanan pada meja yang sama secara bersamaan, sehingga memicu terjadinya *double booking*. Penelitian ini bertujuan mengimplementasikan mekanisme *Concurrency Control* menggunakan fitur *Firebase Transaction* untuk mencegah konflik pemesanan tersebut. Metode penelitian menggunakan pendekatan *Research and Development* (R&D) dengan model ADDIE yang meliputi tahapan *Analysis*, *Design*, *Development*, *Implementation*, dan *Evaluation*. Setelah itu dilakukan pula pengujian melalui simulasi *pre-test* dan *post-test* sebanyak 10 percobaan dengan jumlah pengguna serentak antara 2 hingga 11 orang. Pada tahap *pre-test*, seluruh pengguna berhasil melakukan pemesanan pada *resource* yang sama secara bersamaan sehingga menghasilkan *double booking* pada seluruh percobaan (100%). Pada tahap *post-test*, setelah penerapan mekanisme *Concurrency Control* menggunakan *Firebase Transaction*, dari 10 percobaan yang sama hanya 1 permintaan yang diterima, sementara seluruh permintaan lainnya ditolak secara otomatis, sehingga menghasilkan 0% *double booking*. Hasil tersebut menunjukkan bahwa mekanisme *concurrency control* yang diterapkan terbukti efektif dalam menjaga konsistensi data dan mencegah *race condition* pada sistem pemesanan meja billiard berbasis website.

Kata Kunci - *Concurrency Control, Race Condition, Firebase Transaction, Real-Time Database.*

I. PENDAHULUAN

Perkembangan teknologi informasi dan komunikasi telah memberikan dampak signifikan pada berbagai sektor, termasuk layanan hiburan. Transformasi digital mendorong penyedia layanan untuk mengintegrasikan teknologi dalam sistem operasional guna meningkatkan efisiensi dan kualitas pelayanan [1], [2]. Salah satu bentuk penerapannya adalah *online reservation system*, yang memungkinkan pengguna melakukan pemesanan fasilitas tanpa harus hadir langsung di lokasi [3]. Dalam konteks layanan billiard, sebagian besar tempat hiburan masih menerapkan metode pemesanan manual yang bergantung pada interaksi langsung antara pelanggan dan petugas [4]. Kondisi ini menimbulkan berbagai kendala, seperti antrean panjang, ketidakakuratan informasi ketersediaan meja, serta potensi konflik pemesanan pada periode kunjungan yang tinggi [5], [6]. Implementasi sistem pemesanan berbasis *web* menjadi solusi yang lebih efektif karena memberikan informasi ketersediaan meja secara *real-time* dan memungkinkan pengguna melakukan reservasi kapan saja dan dari mana saja [7]. Meskipun demikian, sistem pemesanan berbasis *web* yang berjalan secara *real-time* tetap menghadapi tantangan ketika beberapa pengguna melakukan pemesanan meja yang sama dalam waktu yang hampir bersamaan[8]. Situasi ini berpotensi menimbulkan *race condition*, yaitu kondisi ketika dua atau lebih proses mengakses dan memodifikasi data yang sama secara paralel tanpa mekanisme pengendalian transaksi yang memadai[9]. Permasalahan ini dapat mengakibatkan *double booking*, yang berdampak pada inkonsistensi data dan menurunkan kualitas pengalaman pengguna[10], [11].

Permasalahan utama dalam penelitian ini adalah ketiadaan mekanisme *concurrency control* pada proses pemesanan meja billiard berbasis *web* yang memungkinkan terjadinya *race condition* pada skenario transaksi simultan[10], [12]. Untuk mengatasi masalah tersebut, *Firebase Realtime Database* menyediakan fitur *runTransaction(),* yaitu mekanisme transaksi atomik yang memastikan bahwa operasi baca tulis dilakukan secara konsisten dan aman, serta melakukan *retry* otomatis ketika terdeteksi konflik data[13], [14]. Dengan karakteristik tersebut, *Firebase Transaction* berpotensi menjadi solusi yang efektif untuk mencegah *race condition* dalam lingkungan multi-user. Penelitian ini bertujuan untuk mengimplementasikan mekanisme *concurrency control* menggunakan *Firebase Transaction* pada sistem pemesanan meja billiard berbasis *web* serta mengevaluasi efektivitasnya dalam mencegah *double booking* melalui pengujian skenario transaksi simultan.

II. SIGNIFIKANSI STUDI

A. Studi Literatur

1. Penelitian Terdahulu

Penelitian mengenai mekanisme *concurrency control* pada sistem berbasis *web* telah dilakukan dalam berbagai konteks. Beberapa studi mengadopsi pendekatan *locking mechanism* maupun *server-side validation* untuk mencegah konflik transaksi, namun pendekatan tersebut tidak selalu mampu menjamin atomisitas transaksi pada lingkungan real-time multi-user[15]. Penelitian lain yang memanfaatkan *Firebase Realtime Database* umumnya berfokus pada sinkronisasi data secara *real-time* dan implementasi CRUD dasar, tanpa membahas risiko *race condition* pada skenario reservasi atau transaksi paralel[16]. Selain itu, studi yang secara spesifik menerapkan *Firebase Transaction* sebagai solusi untuk mencegah *double booking* masih sangat terbatas dan umumnya tidak disertai pengujian simultan dengan banyak pengguna, karena dokumentasi resmi *Firebase* hanya menyajikan perilaku transaksi secara konseptual tanpa evaluasi performa empiris [17]. Oleh karena itu, penelitian ini memberikan kontribusi empiris melalui implementasi dan pengujian *Firebase Transaction* dalam menangani *race condition* pada sistem pemesanan meja billiard berbasis *web* yang berjalan secara *real-time*.

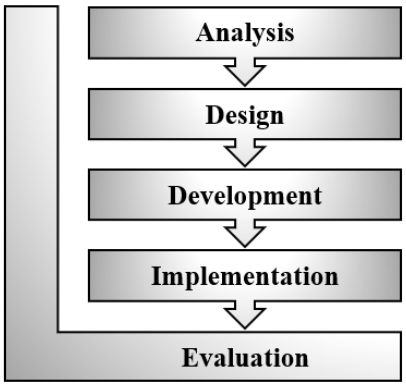
2. Kebaruan Penelitian

Penelitian ini memiliki signifikansi penting dalam pengembangan sistem reservasi berbasis *web* yang melibatkan banyak pengguna secara bersamaan. Permasalahan *race condition* dan *double booking* merupakan *issue* kritis yang dapat menurunkan kualitas layanan, menyebabkan ketidakkonsistenan data, dan mengurangi kepercayaan pengguna. Dengan memanfaatkan *Firebase Transaction* pada *Firebase Realtime Database*, penelitian ini menawarkan solusi yang mampu menjamin atomisitas transaksi dan konsistensi data pada skenario *multi-user* berkecepatan tinggi (kurang dari satu detik).

Dari sisi kebaruan, penelitian ini memberikan kontribusi yang berbeda dari penelitian sebelumnya. Pertama, penelitian ini secara khusus menganalisis penerapan *Firebase Transaction* dalam konteks pemesanan fasilitas, sementara sebagian besar studi sebelumnya hanya membahas fungsi dasar *Firebase* atau sinkronisasi *real-time* tanpa mengevaluasi konflik transaksi. Kedua, penelitian ini menggunakan metode simulasi *pre-test* dan *post-test* melalui 10 percobaan dengan variasi 2 hingga 11 pengguna simultan, sehingga memberikan bukti empiris mengenai efektivitas mekanisme *concurrency control*. Ketiga, penelitian ini diimplementasikan pada sistem nyata dengan pengujian berbasis skenario pengguna riil, sehingga memberikan nilai praktis dan kontribusi langsung terhadap pengembangan sistem reservasi *multi-user*.

B. Metode Penelitian

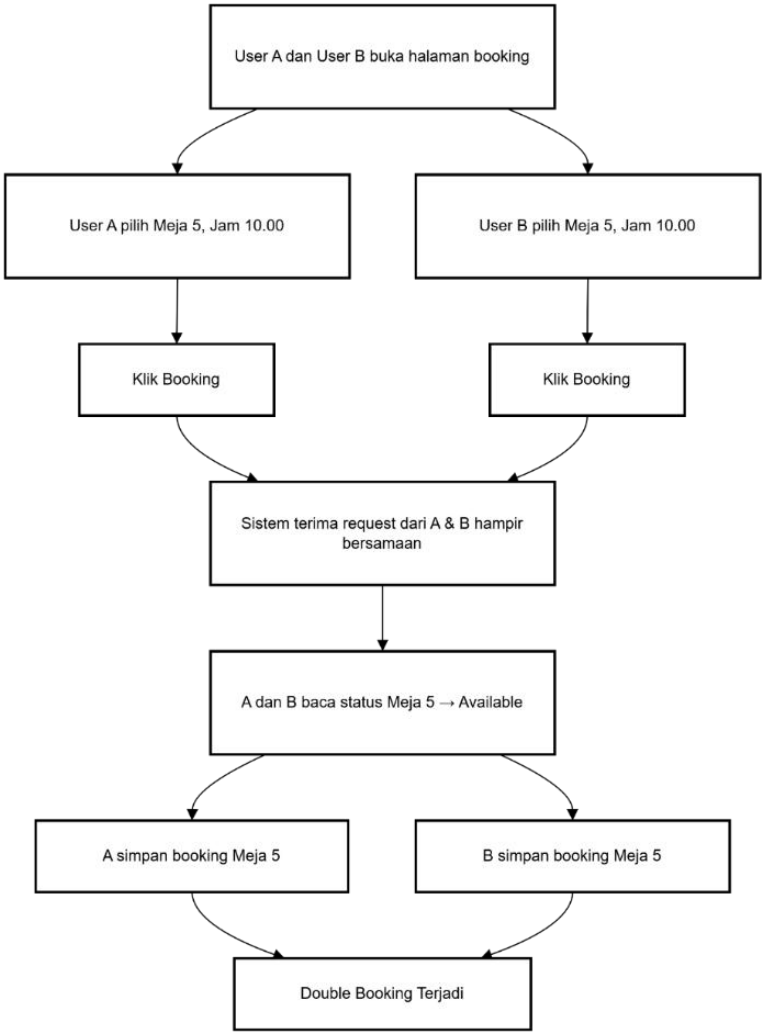
Penelitian ini menggunakan metode *Research and Development* (R&D) dengan mengadaptasi model pengembangan ADDIE yang terdiri dari lima tahap, yaitu *Analysis, Design, Development, Implementation,* dan *Evaluatio n*[18]. Model ini dipilih karena mampu memberikan alur pengembangan sistem yang sistematis, terstruktur, dan sesuai untuk membangun serta mengevaluasi mekanisme *concurrency control* pada sistem pemesanan meja billiard berbasis *web*. Proses penelitian disesuaikan dengan metode *Research and Development* seperti yang tertera di Gambar 1



Gambar 1. Metode *R&D Model ADDIE*

1. Analysis

Tahap *Analysis* diawali dengan proses identifikasi permasalahan pada sistem pemesanan meja billiard berbasis *web* yang telah dikembangkan sebelumnya. Pada tahap ini dilakukan observasi melalui simulasi pemesanan yang dijalankan secara bersamaan menggunakan beberapa perangkat untuk mengetahui bagaimana sistem merespons transaksi yang terjadi secara simultan. Hasil pengamatan awal menunjukkan adanya *race condition* yang menyebabkan beberapa transaksi berhasil masuk pada meja dan waktu yang sama, sehingga memunculkan kasus *double booking*. Alur simulasi yang digunakan untuk mengidentifikasi permasalahan tersebut ditunjukkan pada Gambar 2.



Gambar 2. *Diagram Simulasi Alur Race Condition*

Berdasarkan gambar tersebut, dapat dijelaskan bahwa simulasi dilakukan untuk mengidentifikasi secara empiris bagaimana *race condition* terjadi pada sistem pemesanan sebelum mekanisme *concurrency control* diterapkan. Simulasi dilakukan dengan melibatkan dua hingga beberapa perangkat yang membuka halaman pemesanan secara bersamaan, memilih meja serta waktu yang sama, kemudian menekan tombol *Booking* dalam rentang waktu kurang dari satu detik. Dalam kondisi tersebut, sistem menerima beberapa *request* secara simultan dan setiap klien membaca status meja sebagai *available*. Karena proses baca tulis pada sistem awal belum menggunakan mekanisme atomik, seluruh *request* tersebut menulis data pemesanan tanpa validasi transaksi, sehingga menghasilkan *double booking* pada meja dan waktu yang sama. Hasil simulasi ini menegaskan bahwa sistem sebelumnya sangat rentan terhadap konflik transaksi dan menjadi dasar kebutuhan untuk mengintegrasikan *concurrency control* berbasis transaksi agar proses *booking* tetap konsisten pada lingkungan *multi-user* secara *real-time*.

2. Design

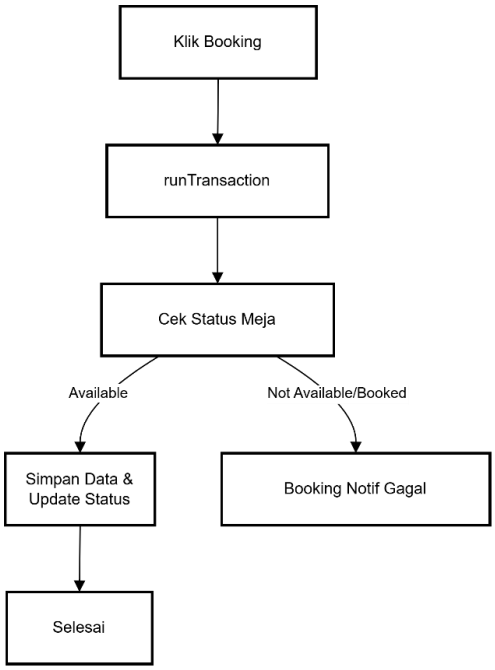
Tahap *Design* berfokus pada perancangan solusi teknis untuk mengatasi permasalahan *race condition* yang ditemukan pada sistem awal. Solusi yang dipilih adalah penerapan transaksi atomik menggunakan fungsi *runTransaction()* pada *Firebase Realtime Database*. Pada tahap ini, struktur data *booking* dirancang ulang agar sistem dapat melakukan pengecekan ketersediaan meja secara *real-time* sebelum transaksi dijalankan. Alur pemesanan juga disusun kembali sehingga proses validasi dan penulisan data berlangsung secara atomik, memastikan bahwa dua pengguna tidak dapat melakukan *booking* pada *resource* yang sama secara bersamaan.

Selain itu, mekanisme *success–failure* handling dirancang untuk menangani situasi kompetisi akses, sehingga hanya satu transaksi yang berhasil ketika dua atau lebih pengguna mengakses meja yang sama secara simultan. Penyesuaian arsitektur *frontend* berbasis *React* dan *TypeScript* juga dirancang agar antarmuka mampu merespons hasil transaksi (berhasil atau gagal) secara konsisten. Dengan demikian, rancangan ini menjadi fondasi implementasi sistem yang lebih andal, terkontrol, dan bebas dari konflik data pada lingkungan *multi-user real-time*.

3. Development

Tahap *Development* merupakan proses mengimplementasikan rancangan ke dalam sistem. Pada tahap ini, mekanisme *concurrency control* diintegrasikan ke fungsi pemesanan dengan membungkus seluruh proses baca–tulis data ke dalam *method runTransaction*() pada *Firebase Realtime Database*. Implementasi *runTransaction()* memastikan bahwa operasi baca–tulis pada path ketersediaan meja berlangsung secara atomik dalam satu siklus transaksi. Ketika pengguna menekan tombol *booking*, sistem membaca status meja secara real-time. Jika nilai masih kosong (meja tersedia), transaksi akan menuliskan data booking baru; sebaliknya, jika data telah terisi, transaksi dibatalkan otomatis tanpa perubahan apa pun. Dengan mekanisme ini, hanya satu transaksi yang dapat berhasil dalam kondisi akses bersamaan, sehingga potensi *double booking* dapat dicegah sepenuhnya. Sebelum memaparkan implementasi kode, alur proses *concurrency control* disajikan pada gambar berikut sebagai dasar pemahaman mekanisme sistem.

1. *Flowchart* Proses *Concurrency Control*



Gambar 3. *Flowchart Mekanisme Concurrency Control*

Alur proses *concurrency control* pada sistem ditunjukkan pada *flowchart* dan bekerja dimulai ketika pengguna memilih meja serta menentukan waktu pemesanan melalui halaman *booking*. Saat tombol *Booking* ditekan, sistem memulai transaksi menggunakan *method runTransaction()* pada *Firebase Realtime Database* untuk memastikan proses baca–tulis berjalan secara atomik. Melalui transaksi ini, sistem membaca status terbaru dari meja yang dipilih. Apabila meja masih berstatus *available*, sistem akan menuliskan data *booking* baru dan sekaligus mengubah status meja menjadi *booked*. Sebaliknya, apabila meja telah terisi *(not availabel/booked)* oleh pengguna lain, transaksi dibatalkan secara otomatis dan sistem menampilkan notifikasi bahwa meja tidak lagi tersedia. Apabila selama proses terjadi konflik akses data, *Firebase* akan menjalankan mekanisme *retry* secara otomatis hingga satu transaksi dinyatakan berhasil, sementara transaksi lainnya dibatalkan.

2. Implementasi *Transaction Lock*

Potongan kode berikut menunjukkan implementasi mekanisme *concurrency control* dengan *transaction lock* menggunakan fungsi *runTransaction()* pada *Firebase Realtime Database*. Kode ini menjadi inti dari pengendalian *race condition* karena seluruh proses baca–tulis pada node ketersediaan meja dibungkus ke dalam satu siklus transaksi atomik.

```
await runTransaction(newBookingRef, (currentData) => {
  if (currentData === null) {
    return {
      uid: user?.uid,
      name,
      date: today,
      time,
      table: selectedTable,
      duration,
      paymentMethod,
      paid: true,
      price: totalPrice,
      rescheduled: false,
    };
  }
  return;
});
```

Gambar 4. *TransactionLock source code*

Gambar tersebut menunjukkan potongan kode implementasi *runTransaction()* yang digunakan untuk menangani proses *booking* secara atomik. Transaksi dijalankan pada *newBookingRef*, yaitu path yang merepresentasikan status ketersediaan meja di *Firebase Realtime Database*. Pada setiap eksekusi transaksi, sistem membaca nilai terbaru melalui parameter *currentData*. Jika *currentData* bernilai *null*, menandakan bahwa meja masih tersedia *(available)*, sistem akan menuliskan data *booking* baru ke database. Namun, jika data telah terisi *(not available/booked)*, transaksi langsung dibatalkan tanpa melakukan perubahan apa pun. Dengan mekanisme ini, hanya transaksi pertama yang berhasil, sedangkan transaksi lain yang terjadi secara bersamaan dibatalkan secara otomatis oleh Firebase, sehingga potensi *double booking* dapat dieliminasi sepenuhnya.

4. Implementation

Tahap *Implementation* dilakukan dengan menguji sistem yang telah dikembangkan untuk menilai efektivitas mekanisme *concurrency control*. Pengujian dilakukan melalui simulasi pemesanan secara simultan dalam dua skenario, yaitu sebelum dan sesudah penerapan *runTransaction()*. Simulasi melibatkan dua hingga sebelas pengguna yang secara bersamaan mencoba memesan meja yang sama dalam rentang waktu yang hampir identik. Parameter yang diamati meliputi jumlah transaksi yang berhasil, jumlah transaksi yang gagal karena konflik, serta jumlah kasus *double booking* yang muncul selama pengujian. Hasil kedua skenario kemudian dibandingkan secara langsung untuk mengevaluasi efektivitas mekanisme *concurrency control* menggunakan *Firebase Transaction* dalam menghilangkan *race condition* serta menjaga konsistensi data pada lingkungan *multi-user real-time*.

5. Evaluation

Tahap *Evaluation* dilakukan dengan menganalisis dan membandingkan hasil pengujian dari dua skenario sebelum dan sesudah penerapan *concurrency control*. Hasil evaluasi menunjukkan bahwa penggunaan *runTransaction()* secara efektif menghilangkan kasus *double booking* karena setiap proses baca–tulis data dipaksa berjalan secara atomik dan konsisten. Selain itu, evaluasi mencakup penilaian terhadap stabilitas sistem, keandalan proses *booking real-time* saat beberapa pengguna melakukan pemesanan secara bersamaan, serta identifikasi area pengembangan pada aspek skalabilitas dan performa. Temuan evaluasi ini menjadi dasar rekomendasi bahwa mekanisme transaksi atomik layak diterapkan pada sistem reservasi serupa yang membutuhkan kontrol konkurensi pada lingkungan *multi-user*.

III. HASIL DAN PEMBAHASAN

A. HASIL

Penelitian ini menunjukkan bahwa sistem pemesanan meja billiard berbasis *web* yang belum menggunakan mekanisme *concurrency control* sangat rentan mengalami *race condition* ketika dua atau lebih pengguna melakukan pemesanan meja yang sama secara bersamaan. Bagian ini menyajikan rangkuman hasil pengujian sebelum dan sesudah penerapan *Firebase Transaction*, serta analisis mengenai efektivitas mekanisme tersebut dalam mencegah terjadinya *double booking*. Pengujian dilakukan dengan tujuan untuk mengevaluasi kemampuan sistem dalam menangani konflik transaksi pada lingkungan *multi-user real-time*, khususnya pada skenario pemesanan meja yang dilakukan secara bersamaan. Tanpa adanya pengendalian transaksi yang memadai, sistem *real-time* yang memproses permintaan dari banyak pengguna secara simultan sangat rawan mengalami inkonsistensi data, sehingga tahap pengujian ini menjadi penting untuk memastikan bahwa solusi yang diterapkan mampu menjaga integritas data secara konsisten.

Metode pengujian dilakukan melalui simulasi transaksi *booking* secara bersamaan dengan menggunakan beberapa perangkat berbeda dalam waktu yang hampir identik. Simulasi ini dirancang untuk merepresentasikan kondisi nyata ketika banyak pengguna mengakses sistem pada waktu yang sama, sehingga dapat diamati bagaimana sistem merespons permintaan simultan pada *resource* meja yang sama. Setiap percobaan dilakukan dengan memvariasikan jumlah pengguna yang terlibat, dari dua hingga sebelas pengguna, baik pada skenario sebelum maupun sesudah penerapan mekanisme *concurrency control*. Secara keseluruhan, pengujian dilakukan sebanyak sepuluh kali percobaan. Variasi jumlah pengguna ini digunakan untuk mengamati bagaimana peningkatan beban transaksi mempengaruhi kemampuan sistem dalam mengatasi konflik data dan mempertahankan konsistensi. Pendekatan bertahap ini memungkinkan peneliti untuk mengevaluasi stabilitas sistem pada berbagai tingkat beban, sekaligus mengidentifikasi batas kemampuan *Firebase Transaction* dalam mencegah *race condition* dan menghilangkan potensi *double booking* pada proses pemesanan meja billiard.

a. Hasil pengujian sebelum penerapan mekanisme *Concurrency Control* ditunjukan pada Tabel 1.

TABEL I
PRE-TEST

No.	Jumlah Pengguna	Waktu Booking	Nomor Meja	Hasil Booking	Jumlah Double Booking
1	2	10:00 WIB	Meja 1	*Double Booking*	2
2	3	11:00 WIB	Meja 2	*Double Booking*	3
3	4	12:00 WIB	Meja 3	*Double Booking*	4
4	5	13:00 WIB	Meja 4	*Double Booking*	5
5	6	14:00 WIB	Meja 5	*Double Booking*	6
6	7	15:00 WIB	Meja 6	*Double Booking*	7
7	8	16:00 WIB	Meja 7	*Double Booking*	8
8	9	17:00 WIB	Meja 8	*Double Booking*	9
9	10	18:00 WIB	Meja 9	*Double Booking*	10
10	11	19:00 WIB	Meja 10	*Double Booking*	11

Hasil pengujian pada kondisi *pre-test* menunjukkan bahwa sistem tanpa mekanisme *concurrency control* sepenuhnya gagal menangani pemesanan simultan. Pada seluruh sepuluh percobaan, setiap skenario menghasilkan *double booking* dengan jumlah yang selalu sama dengan jumlah pengguna yang mencoba memesan meja secara bersamaan. Sebagai contoh, pada percobaan pertama yang melibatkan dua pengguna, sistem mencatat dua entri *booking* untuk meja yang sama. Begitu pula pada percobaan kesepuluh, sebelas pengguna berhasil melakukan *booking* terhadap satu meja pada waktu yang identik.

Temuan ini mengindikasikan bahwa sistem melakukan proses *write* tanpa sinkronisasi, sehingga setiap permintaan yang masuk tetap diproses dan disimpan, meskipun terjadi konflik akses pada *resource* yang sama. Tidak adanya mekanisme pengecekan *atomic* maupun *locking* menyebabkan seluruh proses *booking* dianggap valid oleh sistem, sehingga *race condition* tidak dapat dicegah. Secara empiris, hasil *pre-test* menegaskan bahwa sistem dalam kondisi awal bersifat *vulnerable*, tidak konsisten, dan tidak mampu menjamin integritas data ketika menerima permintaan transaksi secara bersamaan.

b. Hasil pengujian sesudah penerapan mekanisme *Concurrency Control* ditunjukan pada Tabel 2.

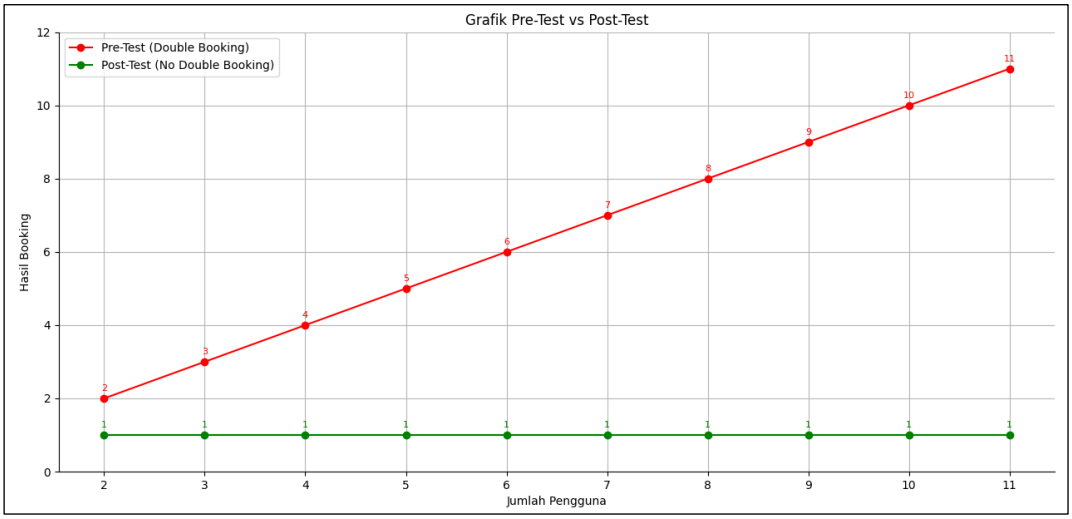
TABEL II
POST-TEST

No.	Jumlah Pengguna	Waktu Booking	Nomor Meja	Hasil Booking	Jumlah Double Booking
1	2	10:00 WIB	Meja 1	1 diterima, 1 ditolak	0
2	3	11:00 WIB	Meja 2	1 diterima, 2 ditolak	0
3	4	12:00 WIB	Meja 3	1 diterima, 3 ditolak	0
4	5	13:00 WIB	Meja 4	1 diterima, 4 ditolak	0
5	6	14:00 WIB	Meja 5	1 diterima, 5 ditolak	0
6	7	15:00 WIB	Meja 6	1 diterima, 6 ditolak	0
7	8	16:00 WIB	Meja 7	1 diterima, 7 ditolak	0
8	9	17:00 WIB	Meja 8	1 diterima, 8 ditolak	0
9	10	18:00 WIB	Meja 9	1 diterima, 9 ditolak	0
10	11	19:00 WIB	Meja 10	1 diterima, 10 ditolak	0

Hasil *post-test* menunjukkan bahwa penerapan mekanisme *concurrency control* menggunakan *runTransaction()* berhasil menghilangkan seluruh kasus *double booking*. Pada setiap percobaan, meskipun beberapa pengguna melakukan pemesanan meja yang sama secara bersamaan, hanya satu transaksi yang diterima, sementara seluruh transaksi lainnya ditolak secara otomatis oleh *Firebase*. Hal ini tampak konsisten dari seluruh sepuluh percobaan, di mana jumlah *double booking* tercatat nol pada semua skenario dengan jumlah pengguna antara dua hingga sebelas.

Pola hasil tersebut mengonfirmasi bahwa transaksi atomik yang dijalankan melalui *runTransaction()* bekerja sesuai desainnya: hanya transaksi pertama yang berhasil mengunci dan menuliskan data, sedangkan transaksi berikutnya gagal karena mendeteksi perubahan *state* pada *resource* yang sama. Dengan demikian, konflik yang muncul akibat akses simultan dapat diselesaikan secara otomatis tanpa menimbulkan inkonsistensi pada data. Temuan *post-test* ini sekaligus membuktikan bahwa mekanisme *concurrency control* mampu menjaga integritas data secara konsisten meskipun beban akses meningkat pada lingkungan *multi-user real-time.*

c. Grafik perbandingan *Pre-Test vs Post-Test*



Gambar 5. *Grafik Pre-Test vs Post-Test*

Grafik perbandingan antara hasil *pre-test* dan *post-test* menunjukkan perubahan yang sangat signifikan setelah penerapan mekanisme *concurrency control* menggunakan *runTransaction()*. Pada grafik hasil *pre-test* terlihat bahwa jumlah *double booking* meningkat secara linier seiring bertambahnya jumlah pengguna, mulai dari dua hingga sebelas pengguna, dengan nilai *double booking* yang selalu sama dengan jumlah pengguna pada setiap percobaan. Pola ini mengindikasikan bahwa sistem sebelumnya tidak memiliki mekanisme pencegahan konflik data sehingga seluruh transaksi yang masuk pada waktu bersamaan dianggap valid dan diproses secara paralel tanpa sinkronisasi.

Sebaliknya, grafik untuk hasil *post-test* menunjukkan garis mendatar pada nilai satu, yang berarti bahwa berapa pun jumlah pengguna yang mencoba memesan *resource* yang sama secara simultan, sistem hanya menerima satu transaksi saja. Seluruh transaksi lainnya otomatis ditolak sehingga tidak menghasilkan *double booking*. Pola ini menegaskan bahwa setelah penerapan transaksi atomik, yaitu mekanisme *concurrency control* mampu memastikan hanya satu pengguna yang berhasil melakukan booking pada satu waktu, meskipun terjadi kompetisi akses dari banyak pengguna secara bersamaan. Perbedaan drastis antara kedua grafik tersebut memberikan bukti empiris bahwa mekanisme *concurrency control* bekerja secara efektif dalam mengelola kondisi kompetisi akses dan menjaga konsistensi data pada lingkungan *multi-user real-time*.

B. PEMBAHASAN

Hasil pengujian pada penelitian ini menunjukkan perbedaan yang jelas antara sistem sebelum dan sesudah penerapan mekanisme *concurrency control berbasis Firebase Transaction*. Pada skenario *pre-test*, seluruh pengguna yang melakukan pemesanan secara simultan berhasil menulis data ke *node* yang sama karena tidak adanya mekanisme kontrol atomik. Kondisi ini menyebabkan jumlah transaksi yang tercatat selalu sama dengan jumlah pengguna, sehingga *double booking* muncul secara konsisten. Temuan tersebut selaras dengan karakteristik sistem *NoSQL* yang mengandalkan *eventual consistency* tanpa dukungan *transaction management* bawaan, sehingga operasi baca–tulis yang tidak atomik rentan menimbulkan *race condition* ketika terjadi akses bersamaan[19].

Setelah mekanisme transaksi atomik melalui fungsi *runTransaction()* diterapkan, hasil *post-test* menunjukkan perubahan yang sangat signifikan. Dalam setiap percobaan, hanya satu transaksi yang berhasil, sementara transaksi lainnya dibatalkan secara otomatis karena *Firebase* mendeteksi perubahan data pada proses validasi ulang. Pola ini konsisten dengan prinsip *optimistic concurrency control (OCC)*, di mana setiap operasi membaca status terbaru sebelum menulis data, dan konflik diselesaikan melalui mekanisme *retry* atau pembatalan transaksi. Dengan demikian, penelitian ini membuktikan bahwa penggunaan *transaction lock* berbasis *Firebase Transaction* mampu menghilangkan *double booking* serta menjaga konsistensi data pada lingkungan *multi-user real-time*, sekaligus memberikan dukungan empiris terhadap teori *OCC* pada sistem basis data modern[20].

IV. KESIMPULAN

Hasil penelitian menunjukkan bahwa mekanisme *concurrency control* menggunakan *Firebase Transaction* efektif menghilangkan *race condition* dan mencegah *double booking* pada sistem pemesanan meja billiard berbasis *web*site, di mana sebelum penerapan mekanisme ini seluruh transaksi simultan selalu menghasilkan konflik, sedangkan setelah diterapkan hanya satu transaksi yang valid dan lainnya dibatalkan secara otomatis. Meskipun demikian, penelitian ini memiliki beberapa keterbatasan. Pertama, simulasi pengujian hanya melibatkan hingga 11 pengguna, sehingga belum merepresentasikan beban akses tinggi sebagaimana yang terjadi pada lingkungan produksi. Kedua, sistem belum diuji secara langsung pada skenario operasional nyata dengan variasi trafik, kondisi jaringan, dan pola penggunaan aktual. Ketiga, penelitian ini belum melakukan pengukuran performa tambahan seperti *latency*, tingkat *retry*, *throughput*, serta respons sistem terhadap skala transaksi yang lebih besar. Oleh karena itu, penelitian selanjutnya disarankan untuk melakukan uji beban berskala besar, mengukur parameter performa secara komprehensif, dan menerapkan pengujian pada lingkungan produksi guna memperoleh hasil yang lebih representatif dan generalis.

REFERENSI

[1] D. Dharmawan And S. Sofiana, “Rancang Bangun Sistem Informasi Penyewaan Studio Musik Berbasis Web Dengan Metode Prototype (Studi Kasus : Legend Musik Studio Ciputat),” Vol. 1, No. 1, 2023.

[2] N. Nurmalasari, S. Mayanti, S. Dewi Ayu Safitri, And M. Kamal Reza, “Sistem Informasi Manajemen Pemesanan Jasa Percetakan Berbasis Web,” *J. Sist. Inf. Akunt.*, Vol. 2, No. 2, Pp. 60–67, Sept. 2021, Doi: 10.31294/Justian.V2i02.999.

[3] I. K. Yudiantoro, “Sistem Informasi Pemesanan Meja Biliar Berbasis Web Studi Kasus: Predator Billiard,” Skripsi, Universitas Kristen Duta Wacana, Yogyakarta, 2024.

[4] Rina Wijayanti, Yety Ariesta Indrastuti, Unsa Hudiana, Dwi Oktafiyah Sumadya, Rena Febrita Sarie, And Mei Indrawati, “Sistem Informasi Penyewaan Lapangan Futsal Berbasis Web Dengan Metode Waterfall Menggunakan Php,” *J. Ilm. Sist. Inf. Dan Ilmu Komput.*, Vol. 5, No. 1, Pp. 09–22, Mar. 2025, Doi: 10.55606/Juisik.V5i1.935.

[5] I. R. M. Mukhlis And Alya Rizky Natasya, “Sistem Informasi Pemesanan Tiket Wisata Kota Surabaya Berbasis Web Menggunakan Metode Model View Controller,” *Informatech J. Ilm. Inform. Dan Komput.*, Vol. 1, No. 1, Pp. 1–9, Mar. 2024, Doi: 10.69533/Bfb9x126.

[6] M. Noor And B. Rahmani, "Aplikasi Web Cerdas Penghindar Konflik Jadwal Order Dokumentasi Kegiatan Pelanggan," *Jutisi: Jurnal Ilmiah Teknik Informatika Dan Sistem Informasi*, Vol. 14, No. 1, Pp. 527–536, Apr. 2025.

[7] I. H. Batubara, E. A. Raihan, M. I. Tanjung, D. Fadlurohman, And A. Can, "Pemanfaatan Sistem Informasi Dalam Pemesanan Serta Digitalisasi Tiket Bus Berbasis Website," *Blend Sains J. Tek.*, Vol. 1, No. 1, Pp. 55–61, July 2022, Doi: 10.56211/Blendsains.V1i1.73.

[8] T. R. Visser, N. A. H. Agatz, And F. R. Spliet, "Managing Concurrent Interactions In Online Time Slot Booking Systems For Attended Home Delivery," *Transp. Sci.*, Vol. 58, No. 5, 2024.

[9] T. H. Lolita B. L. T. And H. Kabetta, "Studi Web Race Condition Sebagai Acuan Pembuatan Modul Praktikum," *Jurnal Info Kripto*, Vol. 15, No. 1, Pp. 46–53, Apr. 2021.

[10] T. Farah, R. Shelim, M. Zaman, And D. Alam, "Study Of Race Condition: A Privilege Escalation Vulnerability," *Systemics, Cybernetics And Informatics*, Vol. 16, No. 1, Pp. 22–26, 2018.

[11] D. Ardiansyah, W. Suharso, And G. I. Marthasari, "Analisis Penerima Bantuan Sosial Menggunakan Bayesian Belief Network," *J. Resti Rekayasa Sist. Dan Teknol. Inf.*, Vol. 2, No. 2, Pp. 506–513, June 2018, Doi: 10.29207/Resti.V2i2.447.

[12] G. Kliukovkin, "How To Solve Race Conditions In A Booking System," Hackernoon, Jan. 2023. [Online]. Available: Https://Hackernoon.Com/How-To-Solve-Race-Conditions-In-A-Booking-System.

[13] B. Ketsman, C. Koch, F. Neven, And B. Vandevoort, "Concurrency Control For Database Theorists," *Acm Sigmod Rec.*, Vol. 51, No. 4, Pp. 6–17, Jan. 2023, Doi: 10.1145/3582302.3582304.

[14] Moldstud, "Solving The Mystery Of Firebase Realtime Database Concurrency Issues," 2024. [Online]. Available: Https://Moldstud.Com/Articles/P-Solving-The-Mystery-Of-Firebase-Realtime-Database-Concurrency-Issues.

[15] M. Y. Shams, A. S. Abolaban, And A. A. Abohany, "A Review On Concurrency Control Techniques In Database Management Systems," *Kafrelsheikh Journal Of Information Sciences*, Vol. 3, No. 1, Pp. 1–10, 2022.

[16] K. N. M. Ngafidin, A. Arista, And R. N. S. Amriza, "Implementasi Firebase Realtime Database Pada Aplikasi Feedbackme Sebagai Penghubung Guru Dan Orang Tua," *Jurnal Resti (Rekayasa Sistem Dan Teknologi Informasi)*, Vol. 5, No. 2, Pp. 327–334, Apr. 2021.

[17] S. Kanungo And R. D. Morena, "Concurrency Versus Consistency In Nosql Databases," *J. Auton. Intell.*, Vol. 7, No. 3, Dec. 2023, Doi: 10.32629/Jai.V7i3.936.

[18] S. Safaruddin, H. Syam, And Darmawang, "Development Of Microlearning Content For Software Engineering Courses By Using Addie Model," *Asian Journal Of Education And Social Studies*, Vol. 49, No. 3, Pp. 515–521, 2023.

[19] M. Diogo, B. Cabral, And J. Bernardino, "Consistency Models Of Nosql Databases," *Future Internet*, Vol. 11, No. 2, P. 43, Feb. 2019, Doi: 10.3390/Fi11020043.

[20] A. A. E. Alflahi, M. A. Y. Mohammed, And A. Alsammani, "Enhancement Of Database Access Performance By Improving Data Consistency In A Non-Relational Database System (Nosql)," Pp. 1–12, 2023.