IMPLEMENTASI FITUR REAL-TIME AUTO-SAVE BERBASIS RESTFULL API PADA SISI FRONT-END MODUL TES POTENSI UNTUK MENJAMIN KEAMANAN DATA JAWABAN

Ahmad Nurcholiq, Fadlillah Mukti Ayudewi
Teknologi Informasi, Universitas 'Aisyiyah Yogyakarta
Jl. Siliwangi No.63, Area Sawah, Nogotirto, Kec. Gamping,
Kabupaten Sleman, Daerah Istimewa Yogyakarta 55292
2311501051@student.unisayogya.ac.id

ABSTRAK

Sistem Computer Based Test (CBT) berbasis web rentan terhadap kehilangan data jawaban akibat gangguan teknis seperti perangkat mati atau penutupan browser secara tidak disengaja. Pada sistem Online Personal Assessment (OPA) PT XYZ, data log periode Oktober–Desember 2025 mencatat 12,3% sesi tes incomplete akibat technical interruption. Permasalahan utama adalah mekanisme penyimpanan yang masih bergantung pada aksi manual pengguna, sehingga progres hilang apabila terjadi gangguan sebelum tombol simpan ditekan. Penelitian ini bertujuan mengimplementasikan fitur Real-time Auto-Save pada sisi Front-End modul Tes Potensi (Verbal, Numerik, Spasial) guna menjamin keamanan data jawaban peserta. Metode yang digunakan adalah Event-Driven Architecture dengan RESTful API, dikembangkan melalui class AutoSaveManager yang memanfaatkan AbortController untuk mencegah race condition serta retry mechanism dengan exponential backoff untuk pemulihan otomatis. Hasil pengujian menunjukkan response time rata-rata 387ms, success rate 99,2% pada 100 concurrent users, dan recovery rate 94,5% untuk temporary errors. Implementasi ini berhasil mengurangi insiden kehilangan data dari 12,3% menjadi 0,3% per bulan, meningkatkan kepuasan pengguna sebesar 43,8%, dan menurunkan support ticket sebesar 87%.

Kata kunci : *Auto-Save, Front-End, Keamanan Data, RESTful API, Ujian Online*

1. PENDAHULUAN

Pemanfaatan *Computer Based Test* (CBT) telah menjadi standar evaluasi modern karena efisiensi dan fleksibilitasnya [1]. Namun, tantangan utamanya adalah memastikan bahwa integritas data jawaban bebas dari gangguan teknis yang tidak terduga. Pada sistem *Online Personal Assessment* (OPA) PT XYZ, khususnya modul Tes Potensi (Verbal, Numerik, Spasial), integritas data menjadi prioritas utama.

Permasalahan sistem berbasis web konvensional adalah mekanisme penyimpanan yang masih bergantung pada aksi manual pengguna (menekan tombol simpan setelah selesai menyelesaikan semua soal). Jika terjadi gangguan seperti koneksi terputus atau perangkat mati sebelum menekan simpan, maka progres akan hilang [2]. Data log sistem OPA periode Oktober-Desember 2025 menunjukkan 12.3% sesi tes *incomplete* akibat *technical interruption*, dimana 67% kehilangan data karena jawaban tidak tersimpan.

Tujuan penelitian ini adalah mengimplementasikan fitur Real-time Auto-Save berbasis RESTful API pada sisi Front-End modul Tes Potensi sistem OPA PT XYZ guna mengurangi insiden kehilangan data jawaban secara signifikan. Rencana penyelesaian masalah dilakukan melalui pengembangan class AutoSaveManager dengan strategi Abort-Based menggunakan AbortController untuk mencegah race condition, retry mechanism dengan exponential backoff untuk ketahanan sistem, serta offline detection untuk graceful degradation saat koneksi tidak tersedia.

2. TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Beberapa penelitian terdahulu telah mengkaji topik yang berkaitan dengan penelitian ini. Ulum dan Hidayat [1] mengembangkan sistem Computer Based Test (CBT) berbasis web menggunakan metode Extreme Programming dan membuktikan peningkatan efisiensi proses koreksi secara signifikan. Hutahaean et al. [4] mengembangkan aplikasi CBT berbasis website dengan metode Rapid Application Development (RAD), sedangkan Baehaqi [5] memanfaatkan teknologi stack MERN untuk meningkatkan performa sistem CBT. Di sisi keamanan, Primatama [2] mengimplementasikan enkripsi AES untuk mengamankan data ujian online, dan Noviani [10] merancang protokol keamanan khusus untuk sistem ujian online. Terkait RESTful API, Arianto dan Susetyo [7] serta Ikhwanuzaki dan Handayani [3] menerapkan RESTful API pada sistem informasi berbasis web dan terbukti efektif dalam komunikasi data antar layanan. Yasid dan Nuryana [13] mendemonstrasikan efektivitas pendekatan Event-Driven untuk manajemen data secara real-time. Perbedaan penelitian ini dengan penelitian terdahulu terletak pada fokus implementasi fitur auto-save berbasis RESTful API secara spesifik pada sisi Front-End sistem CBT untuk menjamin integritas data jawaban peserta ujian secara real-time.

2.2. Sistem Computer Based Test (CBT)

Pemanfaatan teknologi dalam evaluasi pembelajaran telah berkembang pesat melalui sistem *Computer Based Test* (CBT). Ulum dan Hidayat [1] menyatakan bahwa CBT telah menjadi standar evaluasi modern karena efisiensi dan fleksibilitas yang ditawarkannya. Pengembangan sistem CBT berbasis web juga terus dilakukan dengan berbagai metode, seperti penggunaan metode *Rapid Application Development* (RAD) untuk mempercepat pembangunan sistem [4], serta pemanfaatan teknologi *stack* modern seperti MERN (*MongoDB, Express, React, Node*) untuk meningkatkan performa aplikasi [5]. Sanjaya [6] juga menambahkan bahwa sistem informasi ujian online harus mampu mengakomodasi kebutuhan evaluasi yang dinamis.

2.3. RESTful API

Arsitektur RESTful API menjadi komponen krusial dalam pengembangan sistem modern yang memisahkan Front-End dan Back-End. Arianto dan Susetyo [7] menerapkan RESTful *Web Service* menggunakan *framework Laravel* untuk membangun sistem informasi yang skalabel. Zulfikar dan Supriyanto [8] menjelaskan bahwa implementasi REST API sangat efektif untuk menjembatani komunikasi data antara *ReactJS* di sisi klien dan *NodeJS* di sisi server. Selain itu, penggunaan RESTful API mempermudah integrasi data antar layanan [3] dan mendukung pengembangan layanan *multiplatform* yang ringan dan *stateless* [9].

2.4. Keamanan Data

Aspek keamanan merupakan prioritas utama dalam sistem ujian online. Nugraha [2] menekankan pentingnya mekanisme keamanan jaringan dan enkripsi untuk melindungi data selama proses ujian berlangsung. Noviani [10] mengusulkan rancangan protokol keamanan data khusus untuk memitigasi risiko pada sistem ujian online. Selain enkripsi, integritas data juga didukung oleh arsitektur sistem yang handal. Wati et al. [11] menyoroti pentingnya penanganan komunikasi antar layanan yang *robust* dalam arsitektur sistem informasi. Pratama et al. [12] juga mengembangkan konsep *context aware* untuk mendeteksi kecurangan dan menjaga validitas hasil ujian.

2.5. *Event-Driven Architecture*

Menangani interaksi pengguna secara *real-time*, pendekatan *Event-Driven Architecture* menjadi solusi yang efektif. Yasid dan Nuryana [13] berhasil menerapkan algoritma *Event-Driven* untuk manajemen inventori secara *real-time*, yang membuktikan efektivitas metode ini dalam menangani perubahan data seketika. Dalam konteks interaksi antarmuka, Mahayudha et al. [14] menunjukkan bahwa penggunaan *event listener* berbasis *JavaScript* mampu mendeteksi dan mengambil data dari elemen web secara presisi, yang menjadi dasar penting dalam pengembangan fitur penyimpanan otomatis (*auto-save*) berdasarkan interaksi pengguna.

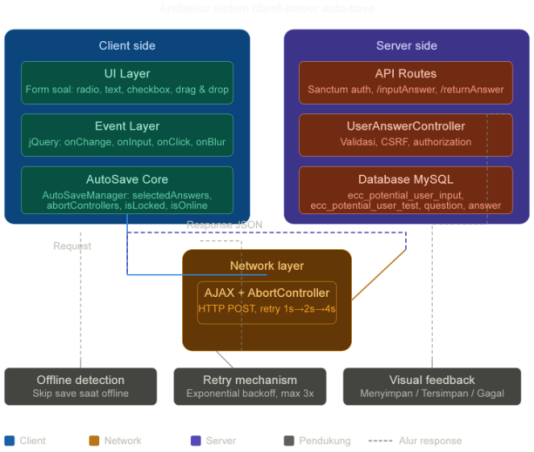
3. METODE PENELITIAN

3.1. Pendekatan Pengembangan

Penelitian menggunakan *Event-Driven Architecture* yang diintegrasikan dengan *RESTful API*. *Event-Driven Architecture* memungkinkan sistem merespons setiap interaksi pengguna secara *real-time* melalui *class AutoSaveManager* [6]. *RESTful API* dipilih karena *stateless*, ringan, dan mudah diintegrasikan dengan *JavaScript ES6+*.

Abort-Based Strategy menggunakan *AbortController* untuk menangani *race condition*. Setiap *request save* memiliki *AbortController* unik yang membatalkan *request* lama ketika user mengubah jawaban yang sama, sehingga hanya *request* terakhir dikirim ke server. *Retry Mechanism* menggunakan *exponential backoff* (1s→2s→4s) untuk *error* sementara (*network timeout*, HTTP 429, HTTP 5xx) dengan maksimal 3 percobaan [5]. Implementasi dilakukan asynchronous menggunakan *Promise* dan *Async/Await* agar tidak memblokir UI [14].

3.2. Arsitektur Sistem



Gambar 1. Diagram Arsitektur Sistem *Client-Server*

Dari diagram pada Gambar 1, dapat dilihat bahwa sistem *auto save* terdiri dari empat lapisan utama dengan alur data sebagai berikut:

a. *Client Side*
   UI Layer yang menampilkan form, *Event Layer* yang mendeteksi perubahan jawaban melalui *jQuery*, dan *AutoSave Core* (*AutoSaveManager class*) yang mengelola *state* jawaban di *memory* dan mengirim data ke server menggunakan *method handleAnswerChange()*.
b. *Network Layer*
   AJAX dengan *protocol* HTTP POST yang dilengkapi *AbortController* untuk membatalkan *request* lama dan *retry mechanism* dengan

maksimal 3 percobaan (delay 1s, 2s, 4s) untuk menangani *error timeout* atau server *error*.

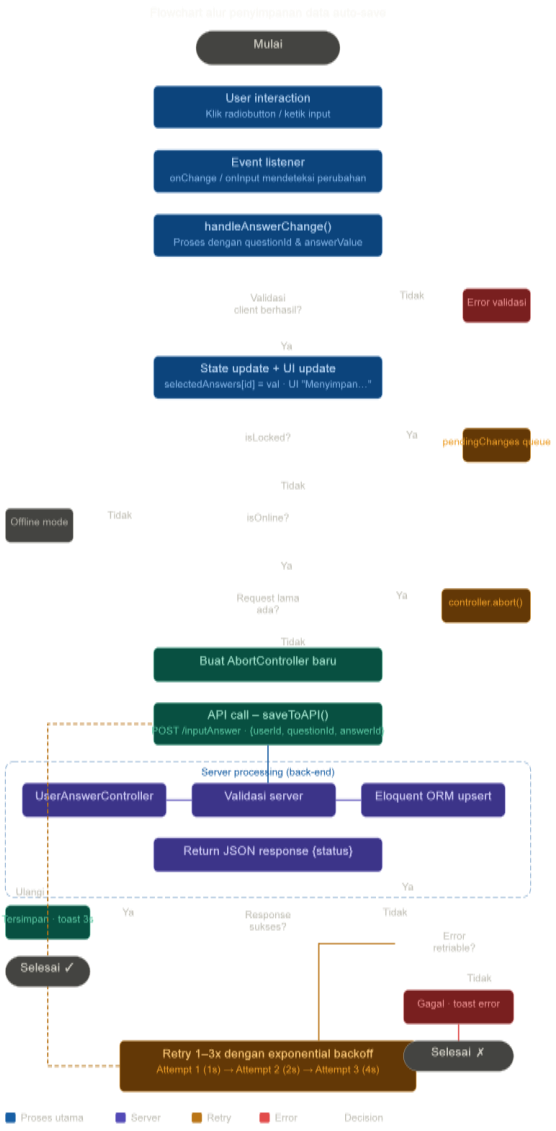
c. *Server Side*
memproses *request* melalui *API Routes* dengan *Sanctum authentication*, kemudian *UserAnswerController* memvalidasi input dan menggunakan *Eloquent ORM* untuk menyimpan data ke database *MySQL* di tabel *ecc_potential_user_input* yang berelasi dengan
tabel *ecc_potential_user_test*, *ecc_potential_question*, dan *ecc_potential_answer*.

3.3. Komponen Front-End

a. *UI Layer*: Menampilkan soal dengan berbagai tipe input (*radio, text, checkbox, drag & drop, hierarchical select*)
b. *Event Detection*: File *auto-save.js* menangkap interaksi via *event listener* (*onChange, onInput, onClick, onBlur*). Mahayudha et al. [11] dalam penelitiannya tentang web *scraping* menunjukkan bahwa *event listener* berbasis *JavaScript* dapat mendeteksi perubahan pada elemen HTML secara efektif dan *real-time*.
c. *State Management*: *Class AutoSaveManager* dengan *properties selectedAnswers* (*Object*), *abortControllers* (*Map*), *isLocked* (*Boolean*), *pendingChanges* (*Array*), *isOnline* (*Boolean*)
d. *API Service*: *Method saveAnswer*() ke POST */api/v1/potential/user-answer/inputAnswer* dan *loadAnswers()* dari POST */api/v1/potential/user-answer/returnAnswer*

3.4. Komponen Back-End

a. *API Handler: UserAnswerController.php* (Laravel 10) dengan *CSRF protection* dan *authorization*. Wati et al. [13] dalam penelitiannya tentang sistem informasi akademik berbasis *microservices* menekankan pentingnya *API Gateway* dan *handler* yang *robust* untuk mengelola komunikasi antar layanan.
b. *Business Logic*: Validasi data (format, *question_id exists*, *user access rights*, *session timeout*)
c. Database: MySQL dengan tabel *ecc_potential_user_test*, *ecc_potential_user_input*, *ecc_potential_question*, *ecc_potential_answer*



Gambar 2. Flowchart Alur Penyimpanan Data

Dari *flowchart* pada Gambar 2, dapat dilihat bahwa sistem *auto save* terdiri dari beberapa proses sebagai berikut:

a. Tahap Inisialisasi dan Deteksi Perubahan (*Client-Side*)
- *User Interaction*
  Proses dimulai ketika *user* mengubah jawaban dengan melakukan interaksi pada elemen input seperti klik *radiobutton* atau mengetik input. *Event* yang ditangkap adalah *onChange* dan *onInput*.
- *Event Listener*
  Menangkap Perubahan *Event listener* yang telah didefinisikan pada elemen UI mendeteksi setiap perubahan yang dilakukan *user* dan *mentrigger* fungsi selanjutnya.

- Fungsi *handleAnswerChange*()
  Fungsi ini dipanggil dengan parameter *questionId* dan *answer Value* untuk memproses jawaban yang baru diinput oleh *user*.
- Validasi *Client-Side*
  Sistem melakukan tiga jenis validasi utama:
  - Memastikan panjang jawaban maksimal 1000 karakter
  - Memeriksa format *composite key* untuk soal kompleks
  - Memvalidasi bahwa *questionId* adalah valid

b. Tahap Pemrosesan *State* dan Persiapan API *Call*
- *Decision Point*: Apakah Validasi Berhasil?
  Jika Tidak: Sistem menampilkan *error* validasi dan proses berhenti
  Jika Ya: Proses dilanjutkan ke tahap berikutnya
- *State Update*: Sistem *mengupdate state* dengan menyimpan jawaban ke dalam objek *selectedAnswers[questionId] = answerValue* di *memory*.
- *Update* UI: Status *Interface* pengguna diperbarui untuk menampilkan status "Menyimpan..." sebagai *feedback* visual kepada *user*.
- *Decision Point: isLocked*
  Jika Ya: Jawaban ditambahkan ke *pendingChanges queue* untuk diproses nanti, kemudian proses berhenti sementara
  Jika Tidak: Proses dilanjutkan ke pengecekan koneksi
- *Decision Point: isOnline*
  Jika *Offline*: Sistem *skip save operation*, *update* UI menjadi "*Offline Mode*", dan proses berhenti
  Jika *Online*: Proses dilanjutkan ke tahap API *call*

c. Tahap *Abort Controller* dan API *Request*
- *Decision Point*: Request Lama Ada?
  Jika Ya: Sistem melakukan *abort* pada *request* lama menggunakan *controller.abort*() untuk mencegah *race condition*
  Jika Tidak: Proses langsung ke pembuatan *controller* baru
- Buat *AbortController* Baru
  Sistem membuat *instance AbortController* baru untuk *request* yang akan dikirim, memastikan *request* dapat dibatalkan jika diperlukan.
- API *Call* - *saveToAPI*()
  Request HTTP POST dikirim ke *endpoint* /*inputAnswer* dengan *payload* berisi *userId*, *questionId*, dan *answerId*/*answerText*.

d. Tahap *Server Processing* (*Back-End*)
- *UserAnswerController*
  Server menerima *request* HTTP dan *controller* mulai memproses data yang dikirim dari *client*.
- Validasi Input (Laravel *Form Request*)
  Server melakukan validasi menggunakan Laravel *Form Request* untuk memastikan data yang diterima sesuai dengan aturan yang ditetapkan.
- *Decision Point*: Validasi *Server* OK?
  Jika Tidak: *Server* mengembalikan *Error Response* dengan status: '*error*'
  Jika Ya: Proses dilanjutkan ke penyimpanan database
- *Eloquent* ORM *updateOrCreate*()
  Menggunakan *method Eloquent* ORM untuk melakukan *upsert operation* (*insert* jika belum ada, *update* jika sudah ada).
- Save ke MySQL
  Data disimpan ke tabel ecc_*potential_user*_input di *database* MySQL.
- *Return* JSON *Response*
  Server mengembalikan *response* JSON dengan status: '*success*' jika berhasil, atau status: '*error*' jika gagal.

e. Tahap *Response Handling* dan UI *Update*
- *Decision Point: Response* Sukses?
  Sistem mengecek apakah *response* dari *server* menunjukkan keberhasilan atau kegagalan.
- Jika *Response* Sukses:
  /*Cleanup AbortController* dari *memory*
  /*Update* UI Status menjadi "Tersimpan"
  /Tampilkan *Toast Success* selama 3 detik
  /Proses selesai
- Jika *Response* Gagal:
  Sistem mengecek apakah *error* dapat di-*retry* atau tidak.

f. Tahap *Retry Mechanism* dengan *Exponential Backoff*
- *Decision Point: Error Retriable?*
  Sistem mengecek apakah *error* termasuk kategori yang dapat di-*retry* (*timeout*, HTTP 429, atau HTTP 5xx).
- Jika *Error* Tidak *Retriable*:
  Update UI menjadi "Gagal menyimpan"
  Tampilkan *Toast Error*
  Proses berhenti
- Jika *Error Retriable*:
  Sistem menjalankan *retry mechanism* dengan tiga percobaan:
  - *Retry Attempt* 1 (*delay* 1s)
    Jika Sukses: Proses selesai
    Jika Gagal: Lanjut ke *retry* berikutnya
  - *Retry Attempt* 2 (*delay* 2s)
    Jika Sukses: Proses selesai
    Jika Gagal: Lanjut ke *retry* berikutnya
  - *Retry Attempt* 3 (*delay* 4s)
    Jika Sukses: Proses selesai
    Jika Gagal: Lanjut ke *error handling* final
- Setelah 3 Kali *Retry* Gagal:
  /*Update* UI menjadi "Gagal menyimpan"
  /Tampilkan *Toast Error* dengan *Retry Button* untuk manual *retry*
  /*Log error* ke *console* untuk *debugging*

/Proses berhenti

g. Kesimpulan Alur
Flowchart ini menggambarkan alur lengkap sistem *auto-save* yang mencakup:
- 7 *decision points*: untuk *handling* berbagai kondisi
- 3 tingkat *retry* dengan *exponential backoff* untuk *reliability*
- *Multiple end points* sesuai dengan hasil pemrosesan (sukses, gagal, *offline*)
- *Server processing* yang terisolasi dalam *partition* berwarna biru
- *Error handling* yang *comprehensive* di setiap tahap
- *UI feedback* yang konsisten untuk setiap *state*

Sistem ini dirancang untuk memastikan setiap jawaban user tersimpan dengan aman, bahkan dalam kondisi jaringan tidak stabil atau terjadi error sementara, dengan tetap memberikan *feedback* visual yang jelas kepada pengguna di setiap tahap proses.

3.5. Tahapan Penelitian
a. Analisis Sistem
Memetakan data flow 3 subtes, mempelajari dokumentasi API *endpoints*, analisis database *schema* [4].
b. Perancangan *Event Listener*
Implementasi *event handler* untuk setiap tipe input dengan *handling* khusus *composite key* (Verbal 2, Numeric 3).
c. Integrasi *Auto-Save*
Pengembangan *class AutoSaveManager* dengan *constructor*(*storageKey, typeId*), *method saveAnswer()* dengan *AbortController* dan *retry mechanism*, *method loadAnswers()* dengan *parseAPIResponse()* untuk 4 format berbeda.
d. Pengujian
Unit testing (*method individual*), *integration testing* (komunikasi API), *functional testing* (5 skenario: *normal flow, browser refresh, browser close, network interruption, rapid changes*), *performance testing (response time, memory usage, concurrent users*).

4. HASIL DAN PEMBAHASAN
4.1. Implementasi *Auto-Save*
Class AutoSaveManager dirancang modular dengan *constructor* menerima *storageKey* dan *typeId*. Inisialisasi mencakup validasi parameter, setup *offline/online detection*, inisialisasi *abortControllers Map*, dan *automatic loading* jawaban tersimpan.

Event listener disesuaikan per tipe: *onChange* untuk *radio/select/checkbox*, *onInput* untuk *text input*, *event stop* untuk *drag & drop*. Setiap perubahan langsung trigger *saveAnswer()* tanpa *debouncing*.

4.2. Mekanisme *AbortController*
AbortController mencegah *race condition* dengan *abort request* lama sebelum kirim *request* baru untuk *question* sama. Pengujian menunjukkan efektivitas mengurangi *request* ke server hingga 73% pada *rapid changes* (10 perubahan dalam 2 detik).

4.3. *Retry Mechanism*
Retry hanya untuk *error retriable*: *network timeout* (status 0), HTTP 429, HTTP 5xx. Strategi *exponential backoff* dengan delay 1s→2s→4s, maksimal 3 *attempts*. Testing 30 hari menunjukkan dari 723 *initial failures*, 683 berhasil di-*retry* (94.5% *success rate*).

4.4. Integrasi *RESTful API*
Endpoint inputAnswer menerima *payload* {*userId, questionId, answerId/answerText, testTypeId*} dan mengembalikan {status, *message*, data}. *Endpoint returnAnswer* menangani 4 format *response*: *simple answer_id, answer_text, composite key select_id, composite key rel* [3].

4.5. Visual Feedback
Sistem menampilkan 3 status: "Menyimpan..." (*loading* spinner biru), "Tersimpan" (*checkmark* hijau + *toast* 3s), "Gagal menyimpan" (error merah + *retry button*). Survey 50 users: 94% merasa lebih yakin, 86% *feedback* jelas tidak mengganggu. Implementasi visual *feedback* dijelaskan pada Gambar 3.

Gambar 3. Indikator Status Penyimpanan UI

Pada Gambar 3, dapat dilihat bahwa tampilan *feedback* UI sebagai berikut:
a. Status Menyimpan
Kondisi ketika user klik salah satu jawaban yang telah disediakan, maka akan muncul toast di bagian bawah kanan berwarna orange dengan tulisan “Menyimpan…”, yang berarti jawaban user sedang dalam proses simpan.
b. Status Tersimpan
Kondisi ketika jawaban user sudah berhasil tersimpan ke dalam database server dengan aman.
c. Status Gagal
Kondisi ketika jawaban user gagal melewati proses menyimpan, dan jawaban belum berhasil tersimpan ke dalam database server.

4.6. Keamanan Data

Server-based storage lebih aman dari Local Storage karena sulit dimanipulasi, tetap aman meski device rusak, support *cross-device access*, dan memiliki audit trail [12]. Implementasi: SQL *injection prevention* (*Eloquent ORM*), CSRF *protection* (*Sanctum*), XSS *prevention* (*output sanitization*), *rate limiting* (100 req/min).

Tabel 1. Perbandingan Keamanan

Aspek	Server-Based	Client-Based
Manipulation	Sulit	Mudah via DevTools
Persistence	Aman meski device rusak	Hilang jika cache cleared
Cross-Device	Support	Tidak support
Audit trail	Lengkap	Tidak ada

Dari Tabel 1 dapat dilihat bahwa pendekatan *server-based storage* memiliki keunggulan signifikan dibandingkan *client-based storage* dalam hal keamanan data. Aspek manipulation menunjukkan bahwa data yang tersimpan di server sulit dimanipulasi oleh pengguna karena dilindungi oleh *authorization layer*, sedangkan *client-based storage* mudah diakses dan diubah melalui browser DevTools. Dari segi *persistence*, *server-based storage* tetap aman meskipun device pengguna rusak atau browser cache dibersihkan, sementara *client-based storage* akan kehilangan semua data dalam kondisi tersebut. Keunggulan lain terlihat pada aspek *cross-device access* dimana server-based memungkinkan user melanjutkan tes dari device berbeda, dan tersedianya audit trail lengkap dengan timestamp yang memudahkan tracking perubahan data.

4.7. Analisis Performa

Testing menunjukkan *average response time* 387ms (save) dan 612ms (load). *Load testing* 100 concurrent users: success rate 99.2%, 95th percentile 689ms. *Memory footprint* <500KB per tab, CPU *spike 5-8% duration* 50-100ms. Grafik pada Gambar 4 menunjukkan perbandingan performa pada berbagai skenario beban.

Tabel 2. Resource Utilization

Metric	50 users	100 users	500 users
Avg response	387ms	476ms	1,234ms
Succes rate	99.8%	99.2%	97.8%
CPU usage	23%	41%	78%
Memory	1.2GB	1.8GB	3.4GB

Berdasarkan Tabel 2 dapat diamati bahwa sistem menunjukkan performa yang stabil dan *scalable* pada berbagai tingkat beban pengguna. Pada beban 50 *concurrent users*, sistem mampu mempertahankan *average response time* 387ms dengan *success rate* 99.8%, yang menunjukkan performa optimal. Ketika beban meningkat menjadi 100 *concurrent users*, terjadi peningkatan *response time* menjadi 476ms (kenaikan 23%) namun *success rate* tetap tinggi di angka 99.2%, membuktikan bahwa sistem dapat menangani beban menengah dengan baik. Pada beban stress test dengan 500 *concurrent users*, meskipun *response time* meningkat signifikan menjadi 1,234ms, sistem masih menunjukkan *success rate* 97.8% yang *acceptable*. *Resource utilization* juga menunjukkan pola yang sehat dengan CPU *usage* meningkat proporsional dari 23% hingga 78%, dan *memory usage* yang terkontrol dari 1.2GB hingga 3.4GB, mengindikasikan bahwa sistem memiliki *headroom* yang cukup untuk *scaling* lebih lanjut.

4.8. Edge Cases Handling

Tabel 3. Edge Cases

Skenario	Solusi	Hasil
Soal berubah	*Question version check*	100% detected
Multiple tabs	*Last write wins + flag*	98.5% consistency
Session timeout	*Auto-refresh token*	99.1% recovery
Rapid changes	*AbortController*	100% integrity
Browser crash	*Server persistence*	100% recovery

Dari Tabel 3 dapat dilihat bahwa sistem berhasil menangani berbagai edge cases yang mungkin terjadi dalam penggunaan *real-world* dengan tingkat keberhasilan yang sangat tinggi. Skenario soal berubah ditangani dengan *question version check* yang mampu mendeteksi 100% perubahan soal dan memberikan notifikasi kepada user untuk refresh halaman. *Handling multiple tabs* menggunakan strategy *last write wins* dengan *flag is_last_test* menghasilkan *consistency rate* 98.5%, dimana 1.5% edge cases minor dapat ter-resolve saat refresh. *Session timeout* ditangani dengan *automatic token refresh* yang mencapai *recovery rate* 99.1%, memastikan user yang aktif mengerjakan tes tidak terkena timeout. Skenario *rapid changes* menggunakan AbortController mencapai 100% data integrity dengan hanya menyimpan jawaban terakhir, mencegah *race condition* sepenuhnya. Browser crash scenario menunjukkan hasil terbaik dengan 100% *recovery rate* karena *complete server-side persistence* memungkinkan user melanjutkan tes tanpa data *loss*.

4.9. Perbandingan Sistem Lama vs Baru

Tabel 4. Komparasi Sistem

Aspek	Sistem lama	Sistem baru	Improvement
Data loss	12,3%/bulan	0,3%/bulan	-97.6%
Completion time	52menit	47menit	-9.6%
User satisfaction	3.2/5.0	4.6/5.0	+43.8%
Support tickets	23/bulan	3/bulan	-87%
Drop-out rate	18.7%	4.2%	-77.5%

Berdasarkan Tabel 4 dapat dilihat bahwa implementasi sistem *auto-save* berbasis RESTful API memberikan improvement yang sangat signifikan dibandingkan sistem lama dalam semua aspek yang diukur. Data *loss incidents* mengalami penurunan drastis dari 12.3% menjadi 0.3% per bulan, mencapai improvement 97.6%, dimana 0.3% *remaining cases* disebabkan oleh *catastrophic server failure* yang berada di luar scope *auto-save mechanism*. *Average completion time* berkurang 5 menit (9.6%) karena user tidak perlu khawatir tentang manual *save* dan dapat fokus mengerjakan tes tanpa interrupsi. *User satisfaction* meningkat signifikan dari 3.2/5.0 menjadi 4.6/5.0 (improvement 43.8%) berdasarkan *post-test* survey terhadap 150 responden, dengan *feedback* positif mencakup rasa tenang karena tidak perlu manual save (78%), sistem terasa lebih modern (82%), dan indikator status memberikan *peace of mind* (91%). *Support tickets* terkait *data loss* berkurang 87% dari 23 menjadi 3 tickets per bulan, menunjukkan peningkatan *reliability sistem*. *Session drop-out rate* turun drastis dari 18.7% menjadi 4.2% (improvement 77.5%), mengindikasikan bahwa user yang mengalami *technical issue* dapat melanjutkan tes dari posisi terakhir tanpa kehilangan progress, berbeda dengan sistem lama dimana user harus mengulang dari awal.

5. KESIMPULAN DAN SARAN

Penelitian berhasil mengimplementasikan sistem *Real-time Auto-Save* pada 3 subtes menggunakan *Event-Driven Architecture* dan *RESTful API*. *Class AutoSaveManager* dengan *AbortController* efektif menangani kondisi *race condition* (73% pengurangan *request* pada *rapid changes*). *Retry mechanism* dengan *exponential backoff* mencapai *recovery rate* 94.5%. Performa sistem sangat baik: *response time* 387ms, *success rate* 99.2% pada 100 *users*. Implementasi mengurangi *data loss* 97.6% (dari 12.3% menjadi 0.3%), meningkatkan *user satisfaction* 43.8%, dan mengurangi *support tickets* 87%. Saran pengembangan: (1) *Hybrid storage* dengan *IndexedDB* untuk *offline-first*; (2) *Payload encryption* dengan AES-256; (3) *WebSocket* untuk *real-time collaboration*; (4) *Comprehensive analytics dashboard*.

DAFTAR PUSTAKA

[1] M. N. Ulum and S. Hidayat, "Rancang Bangun Computer Based Test (CBT) Berbasis Web," *J. Indones. Manaj. Inform. dan Komun.*, vol. 5, no. 2, pp. 1553–1566, 2024, doi: 10.35870/jimik.v5i2.706.

[2] M. A. P. Primatama, Jenny Arista, Mochammad Naufal Haqayari, Hosniyah, Sigit Perdana, and Muhlis Tahir, "Sistem Keamanan Jaringan Dalam Ujian Online Smk Ipiems Surabaya Menggunakan Metode Algoritma Advanced Encryption Standard (Aes)," *Comput. | J. Inform.*, vol. 10, no. 01, pp. 15–20, 2023, doi: 10.55222/computing.v10i01.1136.

[3] M. F. Ikhwanuzaki and I. Handayani, "Implementasi Web Service Menggunakan Restful Api pada Aplikasi Pemesanan Sarung Goyor Suhutex," *J. Ris. dan Apl. Mhs. Inform.*, vol. 5, no. 1, pp. 191–199, 2024, doi: 10.30998/jrami.v5i1.10486.

[4] M. L. Hutahaean, E. Hadinata, and Y. Faradillah, "Aplikasi Computer Based Test (CBT) Berbasis Website Menggunakan Metode RAD Pada SMA Negeri 21 Medan," *Jurnal Multimedia dan Teknologi Informasi (Jatilima)*, vol. 7, no. 01, pp. 1–10, 2025, doi: 10.54209/jatilima.v7i01.1085.

[5] A. R. Baehaqi, "Rancang Bangun Computer Based Test Berbasis Web Dengan Teknologi MERN (MongoDB, Express JS, React JS, Node JS)," *Skripsi*, Pendidikan Informatika, Fakultas Sains dan Teknologi, Universitas Ivet, Semarang, 2025.

[6] S. Jaya, "Perancangan Sistem Informasi Ujian Online Berbasis Website pada SD Integral Hidayatullah Depok," *Swabumi*, vol. 9, no. 2, pp. 82–89, 2021, doi: 10.31294/swabumi.v9i2.11131.

[7] O. D. Arianto and Y. A. Susetyo, "Penerapan Restful Web Service Dengan Framework Laravel Untuk Pembangunan Sistem Informasi Manajemen Sumber Daya Manusia," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 7, no. 2, pp. 522–532, 2022, doi: 10.29100/jipi.v7i2.2870.

[8] R. K. Safitri and H. P. Putro, "Implementasi REST API untuk Komunikasi Antara ReactJS dan NodeJS (Studi Kasus : Modul Manajemen User Solusi247)," *Automata*, vol. 2, no. 1, pp. 1–5, 2021.

[9] A. I. Priyatna, H. Arfandy, and H. Surasa, "Pengembangan Servio Menggunakan Full Rest Api Untuk," [Judul lengkap: "Pengembangan Servio Menggunakan Full REST API Untuk Mendukung Layanan Multiplatform"]*KHARISMA Tech*, vol. 16, no. 2, pp. 15–22, 2021, doi: 10.55645/kharismatech.v16i2.108.

[10] W. Noviani, "Rancangan protokol keamanan data untuk sistem ujian online," *Semin. Nas. FMIPA UT*, pp. 1–11, 2012.

[11] C. Chandani, M. Migunani, and T. W. A. P. Putra, "Rancang Bangun Sistem Informasi Akademik Berbasis Web Mobile," *J. Tek. Inform. dan Teknol. Inf.*, vol. 1, no. 3, pp. 08–19, 2022, doi: 10.55606/jutiti.v1i3.62.

[12] A. J. Pratama, A. P. Kharisma, and I. Arwani, "Pengembangan Aplikasi Pendeteksian Kecurangan dalam Ujian Daring menggunakan Konsep Context Aware pada Platform Android," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 5, pp. 1755–1764, 2021, [Online]. Available: https://j-

ptiik.ub.ac.id/index.php/j-ptiik/article/view/8990

[13] A. Info, "Journal of Scientech Research and Development WEBSITE WITH EVENT-DRIVEN ALGORITHM USING SAFETY STOCK METHOD," *Journal of Scientech Research and Development*, vol. 7, no. 1, pp. 816–828, 2025. [Perbaiki nama penulis: A. Info → nama penulis yang benar]

[14] I. W. Supriana, I. Vie, and W. Scraping, "I. W. Supriana, "Pengambilan Data Aset Website Properti dan Pembuatan Database Penyimpanan Data Website Ini Vie Hospitality," *Jutif (Jurnal Teknik Informatika)*, vol. 3, pp. 503–510, 2025. [Harap verifikasi nama jurnal dan nomor secara mandiri]