JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 2, Juni 2025, Pp. 1460-1473

PENGELOLAAN PERLINDUNGAN DATA PRIBADI MENGGUNAKKAN MONGODB CHANGE STREAMS UNTUK SISTEM NOTIFIKASI *REAL-TIME*

Timothy Arif Kurniawan*[1)], Kristoko Dwi Hartomo[2)]
1. Universitas Kristen Satya Wacana, Indonesia
2. Universitas Kristen Satya Wacana, Indonesia

Article Info
Kata Kunci: Pengelolaan, Data, Change Streams, Notifikasi, Realtime, Websockets

Keywords: Management, Data, Change Streams, Notification, Realtime, Websockets

Article history:
Received 11 November 2024
Revised 15 Desember 2024
Accepted 14 Januari 2025
Available online 15 Maret 2025

DOI :
https://doi.org/10.29100/jipi.v10i2.6134

* Corresponding author.
Corresponding Author
E-mail address:
672020104@student.uksw.edu

ABSTRAK
Perkembangan teknologi memberikan dampak positif maupun dampak negatif bagi masyarakat. Salah satu bentuk dampak negatif perkembangan teknologi adalah munculnya aktivitas pencurian data. Hal tersebut merupakan aktivitas yang dapat menghambat kegiatan masyarakat khususnya kalangan organisasi atau perusahaan. Penelitian ini bertujuan untuk membuat sebuah aplikasi pengelolaan data yang dapat menampilkan riwayat aktivitas operasi data yang terjadi pada collection serta menampilkan *response* dari proses pengelolaan data dalam bentuk *message* notifikasi *realtime*. Untuk mendukung proses pengembangan aplikasi yang diinginkan, maka dibutuhkan teknologi yang dapat dimanfaatkan sebagai pendukung pengembangan aplikasi dan teknologi utama yang digunakkan adalah MongoDB Change Streams dan Websockets. Lalu untuk memastikan aplikasi telah bekerja sesuai dengan *requirements user* dan juga *planning*, maka aplikasi akan melalui tahap pengujian *black box* dan *load testing*. Penelitian ini menghasilkan sebuah aplikasi pengelolaan data dengan notifikasi realtime serta riwayat *log* berbasis web. Dengan aplikasi tersebut, user dapat mengelola data serta melakukan pemantauan terhadap seluruh aktivitas pengelolaan data yang terjadi pada *collection* sehingga hal tersebut dapat mencegah terjadinya aktivitas pencurian data.

ABSTRACT
Technological advancements have both positive and negative impacts on society. One negative impact of technological development is the emergence of data theft activities. This is an activity that can hinder the operations of society, particularly within organizations or companies. This research aims to develop a data management application that can display the history of data operation activities occurring on collections and show responses from data management processes in the form of real-time notification messages. To support the desired application development process, technology that can facilitate development is needed, and the primary technologies used are MongoDB Change Streams and WebSockets. To ensure that the application works according to user requirements and planning, the application will undergo black box testing and load testing. This research results in a web-based data management application with real-time notifications and activity logs. With this application, users can manage data and monitor all data management activities occurring on collections, thus preventing data theft activities.

I. PENDAHULUAN

Berkembangnya sektor teknologi membawa dampak positif bagi masyarakat seperti memudahkan komunikasi antar masyarakat, sarana belajar bagi pelajar serta sumber informasi bagi masyarakat. Namun majunya sektor teknologi juga dapat membawa dampak negatif bagi masyarakat seperti munculnya penipuan, penyebaran berita hoax, serta serangan kejahatan siber dan salah satu bentuk serangan kejahatan siber adalah pencurian data pribadi. Pencurian data pribadi tidak hanya terjadi di dalam masyarakat saja, namun juga menyerang dalam ranah perusahaan [1] [2]. Pencurian data pribadi pada ranah perusahaan sangat berbahaya karena dalam perusahaan terdapat data – data pribadi milik karyawan ataupun milik *customer*. Kegiatan pencurian data memiliki bermacam bentuk salah satunya adalah pembobolan data pada tabel database. Dengan semakin berkembangnya teknologi dapat dimanfaatkan oleh perusahaan dalam tahap meningkatkan perlindungan data

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 2, Juni 2025, Pp. 1460-1473

pribadi ketika proses pengelolaan data berjalan. Pemanfaatan teknologi untuk proses perlindungan data pribadi adalah salah satu bentuk tanggung jawab perusahaan dalam menanggapi salah satu dampak negatif dari teknologi yaitu pembobolan data pribadi dari karyawan ataupun *customer* [3].

Pemanfaatan teknologi untuk mencegah pencurian (pembobolan) data pribadi dari sebuah perusahaan dapat dilakukan dengan berbagai macam cara. Dikarenakan kegiatan pembobolan data pada tabel database terdapat banyak bentuk seperti menghapus, mengubah serta menambahkan data tanpa izin. Salah satu caranya yaitu dengan membangun sebuah aplikasi yang memanfaatkan teknologi notifikasi *realtime* serta menampilkan *log* selama proses pengelolaan data berlangsung dan diperlukan proses pencatatan atau *record* dari *log* agar *user* atau perusahaan dapat mengetahui seluruh proses *input* dan *output* dari proses pengelolaan data yang terjadi [4]. Sehingga, apabila terdapat keanehan yang terjadi selama proses pengelolaan data dari *user* seperti menghapus, mengubah serta menambahkan data yang bersifat mencurigakan atau tanpa izin, maka seluruh aktivitas pengelolaan data tersebut dapat di-tracking dengan efektif oleh perusahaan melalui *record* yang didapatkan dari log. Selain itu, untuk mendukung pengembangan aplikasi akan ditambahkan juga fitur-fitur yang dapat membantu serta mempermudah *user* selama menggunakkan aplikasi pengelolaan data.

Berdasarkan penelitian terdahulu mengenai sistem aplikasi *realtime* yang berjudul "Implementasi Firebase Realtime Database Untuk Aplikasi Pemesanan Menu Berbasis Android**"** [5]. Dalam penelitian tersebut, membahas mengenai perancangan sebuah aplikasi pemesanan menu berbasis android. Pada penelitian tersebut menggunakkan pengimplementasian salah satu teknologi dari firebase yaitu *realtime* database, fungsi dari fitur *realtime* database dimanfaatkan untuk menampilkan data yang memuat hasil pemesanan yang dilakukan oleh *user* secara *realtime* ke bagian *kitchen*. Hasil dari penelitian tersebut mengungkapkan bahwa dengan pemanfaatan fitur *realtime* database tersebut dapat mempermudah *user* serta *kitchen* untuk memproses pemesanan dengan otomatis dan akurat serta lebih cepat. Selain itu dengan pembangunan aplikasi pemesanan menu tersebut juga mempermudah *user* untuk mengetahui harga serta jumlah item yang dipesan pada setiap menu, hal tersebut dikarenakan setiap menu dan seluruh pesanan dari *user* tersimpan pada firebase database API yang kemudian akan terkirim ke *user* lain (*kitchen*) secara *realtime*. Terdapat juga penelitian yang berjudul "Rancang Bangun Aplikasi Mobile Penjadwal Perkuliahan Dengan Firebase Dengan Realtime Notification [6]. Dalam penelitian tersebut, membahas mengenai perancangan dan pembangunan aplikasi yang memiliki tujuan dan manfaat untuk memberikan pemberitahuan atau notifikasi sebagai *reminder* mengenai jadwal dan agenda kegiatan dari mahasiswa (*user*). Pada pembangunan aplikasi akan didukung oleh salah satu teknologi dari firebase yaitu firebase *realtime* database dan karena aplikasi dari penelitian dirancang untuk memberikan notifikasi kepada *user* maka dibutuhkan fitur lain dari firebase yaitu Firebase Notification. Dengan adanya penelitian mengenai rancang bangun aplikasi penjadwal perkuliahan dengan *system realtime notification* dapat mempermudah *user* (mahasiswa) untuk mendapatkan notifikasi secara langsung dari *user* lainnya (dosen) dengan otomatis sehingga mahasiswa dapat menerima informasi yang lebih akurat dari dosen yang bersangkutan tanpa mengalami kesalahpahaman maupun keterlambatan informasi.

Untuk mengatasi permasalahan yang ada dan dalam membangun aplikasi, diperlukan teknologi yang mendukung pengembangan. Teknologi utama yang digunakan adalah MongoDB Change Streams dan Realtime Notification (Websockets). Penggunaan MongoDB Change Streams dikarenakan kemampuannya dapat menampilkan aktivitas *logging* pada tabel (*collection*) secara *realtime* selama proses pengelolaan data. Sebaliknya, Firebase Realtime Database tidak menyediakan fitur otomatis untuk *logging* dan memerlukan *custom log* secara manual. Oleh karena itu, pemanfaatan MongoDB Change Streams memudahkan pengembangan aplikasi karena dapat menampilkan *logging* yang terjadi secara *realtime*, sehingga seluruh aktivitas pengelolaan data dapat di-*tracking* dengan jelas dan tepat serta dapat membantu mengidentifikasi aktivitas mencurigakan salah satunya pembobolan data yang merupakan salah satu bentuk aktivitas pencurian data. Selain itu, pemanfaatan Websockets digunakan untuk menampilkan notifikasi pesan secara *realtime* kepada pengguna setelah melakukan pengelolaan data. Diharapkan dengan pemanfaatan kedua teknologi tersebut dapat memenuhi *requirements* dari *user* yang menjadi dasar dalam membangun aplikasi serta dapat memberikan dampak perubahan bagi perusahaan dalam melakukan perlindungan data pribadi dengan meminimalisir terjadinya pembobolan data pada database selama proses pengelolaan data berlangsung.

II. METODE PENELITIAN

Dalam proses penelitian dan pembangunan aplikasi digunakkan sebuah metode yaitu Metode Agile. Metode Agile adalah metodologi pengembangan *software* yang didasarkan pada proses pengerjaan yang dilakukan

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 2, Juni 2025, Pp. 1460-1473

berulang dimana, aturan dan solusi yang disepakati dan dilakukan secara terorganisir dan terstruktur [7] [8] [9]. Pada Metode Agile terdapat 6 tahapan yaitu *Requirements*, *Design*, *Developing*, *Testing*, *Deploy*, dan *Review*.

Gambar 1. Metode Agile

A. Tahap Requirements

Tahap *requirements* merupakan tahap awal dari metode agile. Pada tahap ini dilakukan perancangan terhadap *requirement user* yang menjadi dasar dalam pengembangan sistem aplikasi kedepannya. Tabel I merupakan tabel *requirement user* yang menjadi dasar dalam pembangunan sistem aplikasi.

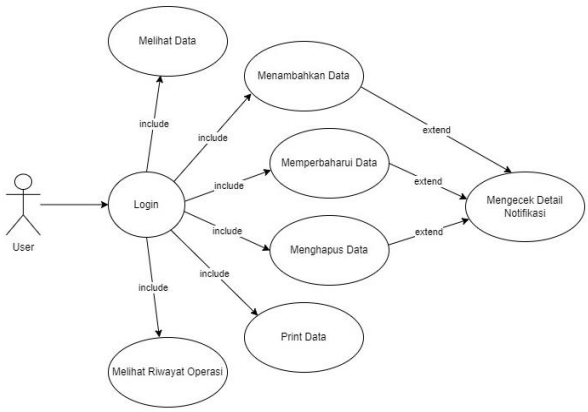
TABEL I
REQUIREMENTS USER

No.	Actor	Activity
1.	USER	Melakukan pengelolaan data (*Create, Read, Update, Delete*).
2.		Melihat aktivitas *log operation* pada database dengan menampilkan pada *page history*.
3.		Menampilkan notifikasi *realtime* setelah melakukan pengelolaan data.
4.		Melakukan *print* data untuk mencetak seluruh data dalam bentuk file.
5.		Melakukan *login* dan *logout*.

B. Tahap Design

Pada tahap *design* akan dilakukan perancangan secara fungsionalitas terhadap alur yang akan dijalankan oleh sistem aplikasi. Perancangan dan penggambaran alur diimplementasikan dalam bentuk *output* berupa diagram. Tujuan dari tahap *design* yaitu untuk memberikan panduan atau alur kepada *developer* mengenai proses dan rancangan dari sistem aplikasi yang akan dibangun. Berikut adalah diagram – diagram yang menjadi alur fungsionalitas dari sistem aplikasi.

1) Use – Case Diagram



Gambar 2. Use-Case Diagram

Pada gambar 2 use-case diagram merupakan implementasi dari tabel *requirement user*. Berdasarkan use-case diagram aktivitas yang perlu dilakukan oleh *user* pertama kali adalah melakukan *login*, karena dengan *login user* dapat melakukan aktivitas lainnya pada aplikasi. Maka pada use-case diagram aktivitas melihat data, menambahkan

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 2, Juni 2025, Pp. 1460-1473

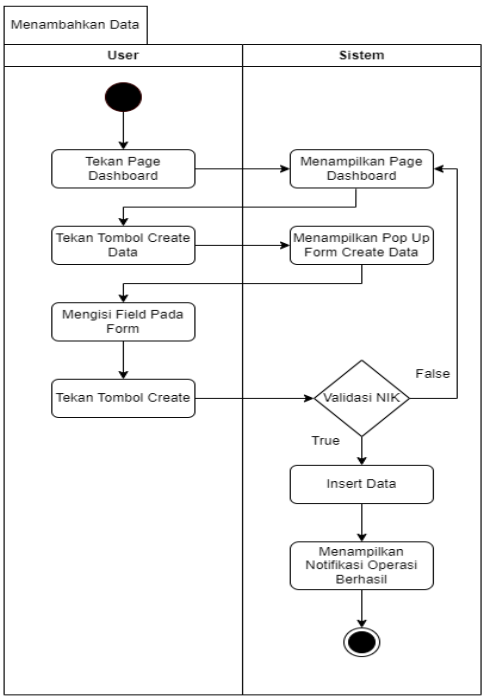
data, memperbaharui data, menghap perasi merupakan *include* dari *login* karena tanpa *login user* tidak dap ut. Pada aktivitas mengecek detail notifikasi memiliki relasi *extend* de i data, dan menghapus data. Hal ini dikarenakan meskipun mengecek d mbutuhkan aktivitas menambahkan data, memperbaharui data, dan meng ail notifikasi memiliki *button* khusus untuk menjalankan aktivitas tersebu

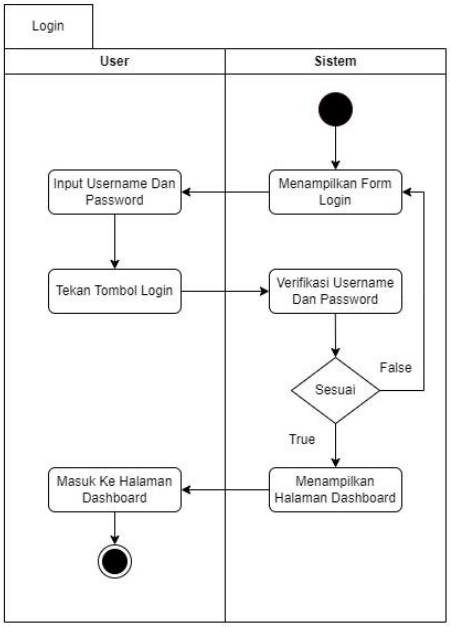
2) Activity Diagram

Dalam perancangan sebuah siste rsebut diperlukan guna memberikan informasi mengenai alur fungsiona m aplikasi yang dibangun. Berikut adalah bentuk activity diagram dari asi.

a) Login

Gambar 3 merupakan activ igambarkan *state* dari sisi *user* mau-pun sistem ketika proses *logi username* dan *password* maka secara otomatis sistem akan melakuk yang diinput oleh *user* sesuai dengan yang tersimpan di database at



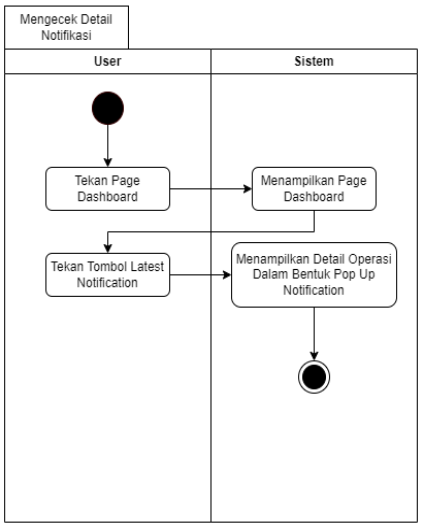


Gambar 3. Activity Diagram Login

b) Menambahkan Data

Gambar 4 merupakan activity diagram menambahkan data. Dalam alur *task* menambahkan data, setelah *user* meng-input *field – field* pada *form* terdapat validasi pada *field* NIK. Hal tersebut bertujuan untuk mengantisipasi agar tidak terdapat NIK yang sama pada data di database. Apabila data sudah tervalidasi maka data akan tersimpan di dalam database.

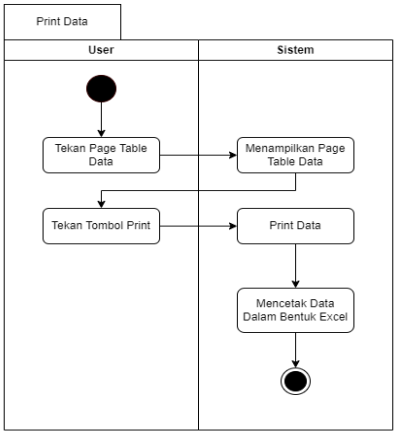
JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 2, Juni 2025, Pp. 1460-1473



Gambar 4. Activity Diagram Menambahkan Data

c) Print Data

Gambar 5 merupakan activity diagram print data. Dalam alur *task print* data ketika *user* tekan tombol *print* maka secara otomatis sistem aplikasi akan mencetak data dalam bentuk file excel. File excel berisikan data – data yang diambil dari *collection* pada database.



d) Mengecek Detail Notifikas

Gambar 6 merupakan activity diagram mengecek detail notifikasi. Dalam alur *task* mengecek detail notifikasi berfungsi untuk menampilkan detail notifikasi dari proses operasi yang sudah dilakukan. Data yang ditampilkan pada detail notifikasi merupakan data yang berasal dari *log* operasi database yang terbaru.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 2, Juni 2025, Pp. 1460-1473

Gambar 6. Activity Diagram Mengecek Detail Notifikasi

C. Tahap Developing

Pada tahap *developing developer* akan melakukan implementasi dari tampilan maupun alur proses dari aplikasi yang sudah dirancang dengan melakukan proses *coding*. Proses implementasi dengan *coding* yang dilakukan oleh *developer* akan berdasarkan *requirement*, tujuan, dan alur yang sudah ditentukan. Dalam proses implementasi aplikasi *developer* membutuhkan beberapa teknologi yang pastinya digunakkan untuk mendukung seluruh alur dan *requirements* dalam pembangunan aplikasi. Berikut adalah teknologi utama yang diperlukan selama proses pembangunan aplikasi.

1) MongoDB Change Streams

MongoDB adalah database yang bersifat *open-source* yang memberikan kinerja tinggi, ketersediaan tinggi serta *automatic scaling*. MongoDB juga merupakan salah satu bentuk basis data NoSQL. Metode penyimpanan yang digunakan oleh MongoDB adalah *document-store* dimana penyimpanan data dimasukkan kedalam dokumen [10] [11]. MongoDB juga memiliki banyak fitur, yang salah satunya adalah MongoDB Change Streams. MongoDB Change Streams memungkinkan aplikasi mengakses perubahan data secara *real-time* tanpa melewati langkah yang rumit. Pengoptimalan dari MongoDB Change Streams memberikan pemanfaatan sumber daya yang lebih efisien dan eksekusi yang lebih cepat. Penggunaan MongoDB Change Streams pada pembangunan aplikasi dapat menghubungkan semua perubahan data pada satu koleksi maupun database. Penerapan MongoDB Change Streams pada aplikasi nantinya akan digunakkan sebagai media untuk menampilkan aktivitas *log* dari sebuah *collection* secara *realtime* yang *log* tersebut akan digunakkan pada aplikasi dalam konteks menampilkan riwayat pengelolaan data. Sehingga user dapat memantau seluruh aktivitas pengelolaan data melalui *log* yang ditampilkan sebagai bentuk riwayat pengelolaan. Contoh apabila *user* melakukan pengelolaan data berupa penambahan data maka akan terjadi aktivitas perubahan data pada *collection*. Dengan pemanfaatan MongoDB Change Streams maka aktivitas perubahan data pada *collection* tersebut akan terpantau dan ditampilkan dalam bentuk *log* yang nantinya *log* tersebut akan digunakkan sebagai *object* untuk ditampilkan sebagai riwayat pengelolaan data.

2) WebSocket

WebSocket merupakan standar baru untuk melakukan komunikasi secara *realtime* pada web ataupun aplikasi mobile. Dengan WebSocket memungkinkan *client* dapat menerima data dari server tanpa melakukan *request* data terlebih dahulu [12] [13]. Penerapan WebSocket pada pembangunan aplikasi digunakkan sebagai media untuk memungkinkan sebuah *message* dikirimkan secara *realtime* menggunakkan socket yang diimplementasikan pada *code back-end* dan *code front-end* sehingga *value message* yang akan ditampilkan antar sesama pengguna akan disimpan dan ditrigger dari *code back-end* dan akan diterima oleh *code front-end* yang kemudian akan ditampilkan dalam bentuk notifikasi antar sehingga *user* dapat menerima notifikasi pengelolaan data antar pengguna secara *realtime*. Dengan penerapan WebSocket pada aplikasi dapat meningkatkan kinerja pengelolaan data pada aplikasi karena nantinya *message* yang ditampilkan berupa notifikasi akan memuat tipe pengelolaan data yang dilakukan beserta nama *user* yang melakukan kelola data tersebut sehingga antar *user* dapat mengetahui *user* yang telah melakukan pengelolaan data.

D. Tahap Testing

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 2, Juni 2025, Pp. 1460-1473

Tahap *testing* akan dilakukan pengujian terhadap aplikasi yang sudah diimplementasikan oleh *developer* dengan proses *coding*. Proses pengujian dilakukan guna untuk memeriksa apakah ada *error* atau *bug* yang terjadi ketika aplikasi dijalankan. Dalam melakukan pengujian sebuah sistem diperlukan sebuah metode untuk mengetahui validasi dari sistem aplikasi. Metode yang akan digunakkan dalam proses pengujian pada aplikasi adalah *Black-Box Testing. Black-Box Testing* merupakan metode pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program untuk mengetahui apakah fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan [14] [15]. Teknik testing yang dilakukan menggunakkan metode Black-Box Testing adalah dengan Functional Test. Functional Test merupakan teknik Black-Box Testing yang menitikberatkan kepada sisi fungsionalitas fitur dari aplikasi secara rinci dan spesifik [16]. Penerapan Black-Box Testing pada sistem aplikasi yaitu dengan melakukan pengujian yang dilakukan oleh penguji eksternal atau penguji pihak ketiga yang dipandu secara spesifik oleh *developer*. Selain diuji menggunakkan metode Black-Box Testing, aplikasi akan dilakukan juga pengujian menggunakkan metode Load Testing. Load Testing adalah metode testing yang menerapkan teknik *performance* yang dimana respon sistem diukur dalam berbagai *load condition* seperti sistem dapat bekerja dalam kondisi beban *task* yang beragam misalnya jumlah *user* yang mengakses aplikasi [17] [18]. Load Testing akan dilakukan menggunakkan aplikasi apache jmeter yang akan diarahkan langsung ke url dari aplikasi.

E. Tahap Deploy

Pada tahap *deploy* akan dilakukan proses konfigurasi dan pemasangan ataupun penyebaran aplikasi agar aplikasi yang sudah dibangun dapat diakses dan digunakkan oleh para *user*.

F. Tahap Review

Tahap *review* dilakukan guna untuk mengetahui fitur, alur, serta tujuan pembangunan aplikasi sesuai atau tidak dengan *requirements* dari *user* tanpa mengalami *error* maupun *bug* pada aplikasi.

III. HASIL DAN PEMBAHASAN

A. Developing Sistem

1) Change Streams

Tabel II adalah *code* python yang merupakan implementasi dari teknologi MongoDB Change Steams.

TABLE II
CHANGE STREAMS

Kode Program

```
def change_stream_monitor(client, time_in_ms=60000, pipeline=[]):
    collection = client.ta.project
    coll = clientLog.cstream.logcs

    change_stream = collection.watch(pipeline)

    for change in change_stream:
        print(change)

        getType = change.get('operationType')

        if getType == "insert" or getType == "update":

            getKey = change.get('documentKey')
            getId = getKey.get('_id')

            if getId:
                findData = collection.find_one({'_id': getId})
                getUser = findData.get('user')
                change['user'] = getUser

        elif getType == "delete":

            change['user'] = "user"

        result = coll.insert_one(change)
```

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 2, Juni 2025, Pp. 1460-1473

```
27          result
28
29      close_change_stream(time_in_ms, change_stream)
```

Penerapan code change streams seperti pada tabel II bermanfaat untuk membantu proses kinerja aplikasi dalam menjalankan proses pengelolaan data karena setiap prosesnya akan tercatat melalui *log* yang nantinya log akan disimpan dalam bentuk riwayat pengelolaan beserta nama *user* yang melakukan pengelolaan tersebut sehingga *user* apabila ingin memastikan apakah proses yang telah dijalankan berhasil atau tidak hanya perlu melakukan *checking* di bagian riwayat. Selain itu dengan *log*, user juga dapat memantau seluruh aktivitas pengelolaan data pada sebuah *collection* secara *realtime* dan spesifik terstruktur.

Pada tabel II baris 5 *code* change_stream = collection.watch(pipeline). Bagian collection.watch merupakan fitur dari mongodb (change streams) untuk menampilkan aktivitas *log* yang terjadi pada *collection* yang digunakkan ketika operasi pengelolaan data berlangsung dan *pipeline* sendiri merupakan variable berbentuk *array list* yang berfungsi untuk menampung data *log* yang terjadi pada *collection*. Baris 26 *code* result = coll.insert_one(change) berfungsi untuk menambahkan data – data *log* dari aktivitas pengelolaan data ke dalam *collection* logcs, yang nantinya *log* – *log* yang tersimpan pada *collection* logcs akan diambil dan ditampilkan pada halaman *history* pengelolaan data. Baris 20 *code* change['user'] = getUser, *code* tersebut berfungsi untuk menambahkan *user* yang melakukan operasi data tersebut kedalam halaman *history*.

2) Operation Data dan Notifikasi Real-Time (WebSocket)

Penerapan teknologi notifikasi *realtime* digunakkan sebagai salah satu alat untuk proses pengelolaan data, apabila *user* telah selesai melakukan pengelolaan data maka akan muncul notifikasi *realtime* yang memberikan *message* kepada *user* mengenai proses yang terjadi setelah *user* melakukan pengelolaan data. Sehingga *user* dengan antar *user* lainnya dapat mengetahui proses pengelolaan data yang telah terjadi tersebut disertai dengan nama usernya secara *realtime*.

Tabel III adalah *code* yang berfungsi untuk melakukan pengelolaan data berupa *insert* data ke dalam *collection.*

TABLE III
OPERATION DATA & NOTIFIKASI REALTIME

Kode Program (Python)

```
1  app.route("/create", methods=["POST"])
2  def create_data():
3      if "user" in session:
4          postNik = int(request.form.get("nik"))
5          getNik = collection.find_one({"nik": postNik})
6
7          if getNik:
8              message = """NIK Sudah Tersedia Harap Ulang Create Data
!!!"""
9              return redirect(url_for('index', message=message))
10
11         new_listing = {
12             "nik": int(request.form.get("nik")),
13             "nama": request.form.get("nama"),
14             "tgllahir": datetime.strptime(request.form.get("tglla-
hir"), '%Y-%m-%d'),
15             "jabatan": request.form.get("jabatan"),
16             "telp": int(request.form.get("telp")),
17             "alamat": request.form.get("alamat"),
18             "user": session["user"]
19         }
20
21         result = collection.insert_one(new_listing)
22
23         if result.inserted_id:
24             message = f"""Operation Was Successful || TYPE : INSERT ||
25 Managed By : {session["user"]} ||
26 Check On Latest Notification For More Detail Operation ||"""
27
28             socketio.emit('notification', {'message': message})
29
```

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 2, Juni 2025, Pp. 1460-1473

```
            return redirect(url_for('index', message=message))
        else:
            message = """Tidak Berhasil Create Data"""
            return redirect(url_for('index', message=message))
    else:
        return redirect('/')
```

Pada tabel III baris 21 *code* result = collection.insert_one(new_listing) merupakan *code* utama yang menjalankan proses pengelolaan *insert* data ke dalam *collection.* Baris 24 dan 28 terdapat juga baris *code message* dan socketio.emit, *code* tersebut berfungsi untuk mengirimkan *message* ke tampilan yang dimana *message* tersebut ditampilkan ketika proses *insert* data telah berhasil dan pada *message* juga terdapat format session[“user”] yang dimana *code* tersebut berfungsi untuk menambahkan nama *user* yang telah melakukan operasi data ke dalam notifikasi *message*. *Code* socketio.emit merupakan penerapan teknologi websocket yang berfungsi untuk membantu mengirimkan *value* secara *realtime* ke sistem aplikasi. Pada baris 28, *code* ‘notification’ merupakan *key* yang digunakkan oleh socket sebagai media untuk mengirim *value* yang berupa *message* dari python yang nantinya *value message* tersebut akan dikirim ke *code* javascript dengan ‘notification’ sebagai *key* yang akan digunakkan oleh socket untuk melakukan validasi terhadap *value* yang dikirim dengan media *key* ‘notification’.

Tabel IV adalah *code* yang berbentuk script dalam *language* javascript yang berfungsi untuk menerima *value* berupa *message* yang dikirimkan oleh socketio.emit pada python dengan *key* ‘notification’. *Message* yang ditampung di script pada *code* menggunakkan teknologi websocket akan ditampilkan pada sistem aplikasi dalam bentuk *alert* notifikasi yang muncul setelah proses pengelolaan data dilakukan yang notifikasi tersebut berbentuk *realtime* dengan pemanfaatan teknologi websocket.

TABLE IV
OPERATION DATA & NOTIFIKASI REALTIME

Kode Program (JavaScript)

```
<script>
            $(document).ready(function() {
                $('#createButton').click(function() {
                    $.get('/create', function(data) {
                        console.log("Data dari server:", data);
                        alert("Data dari server: " + data);
                    });
                });

                var socket = io.connect('http://' + document.domain +
':' + location.port);
                console.log("Socket.IO:", socket);
                socket.on('notification', function(data) {
                    alert(data.message);
                });
            });
</script>
```

3) *Print Data*

Tabel V adalah *code* yang berfungsi untuk mencetak data dari *collection* ke dalam bentuk file excel.

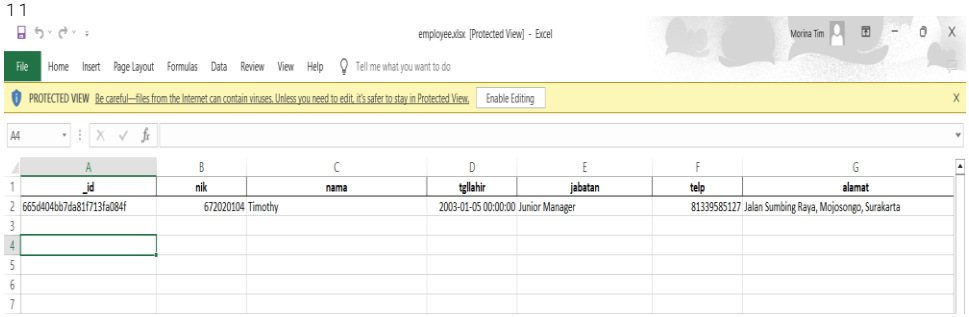
TABLE V
PRINT DATA

Kode Program

```
def generate_excel():
    data = list(collection.find())

    df = pd.DataFrame(data)

    excel_file = 'employee.xlsx'

```

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 2, Juni 2025, Pp. 1460-1473

```
8       writer = pd.ExcelWriter(excel_file, engine='openpyxl')
9
10      df.to_excel(writer, index=False)
11
```



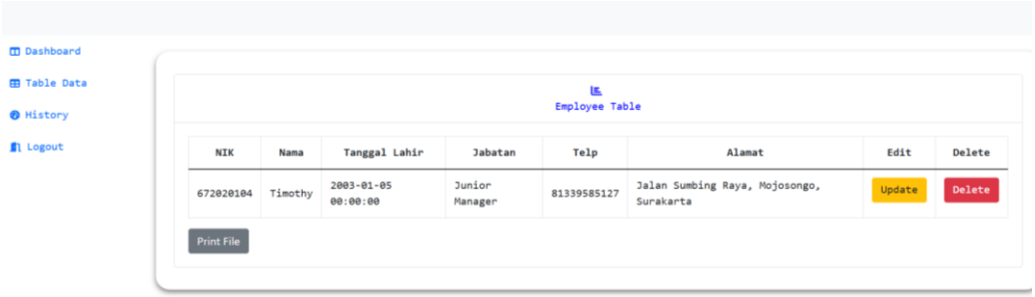

```
24
25      for col, width in column_widths.items():
26          worksheet.column_dimensions[col].width = width
27
28      writer.close()
29
30      return excel_file
31
32  @app.route('/download')
33  def download_excel():
34      excel_file = generate_excel()
35      return send_file(excel_file, as_attachment=True)
```

Untuk *develope* fitur 'Print Data' diperlukan penambahan *library* baru yaitu openpyxl. Untuk menjalankan teknologi openpyxl pada python perlu melakukan *import* openpyxl untuk memanggil *library* atau fungsi – fungsi yang tersedia pada openpyxl.

B. Implementasi Sistem

1) Halaman Table Data

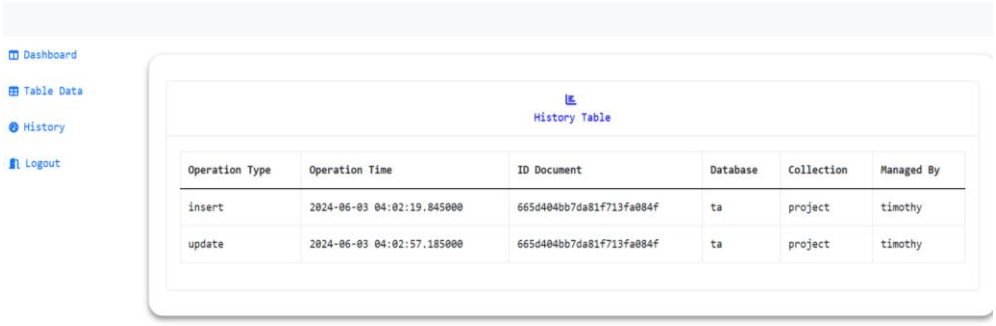
Gambar 7 merupakan tampilan halaman table data. Pada halaman table data terdapat *button* "Update" yang berfungsi untuk memperbaharui data yang diinginkan yang kemudian apabila proses berhasil maka data yang diperbaharui akan tersimpan di dalam *collection.* Terdapat *button* "Delete" yang berfungsi untuk menghapus data yang diinginkan dari *collection*. Lalu terdapat *button* "Print" yang berfungsi untuk mencetak data dari *collection* ke dalam bentuk *output* berupa file excel.



2) Print Data

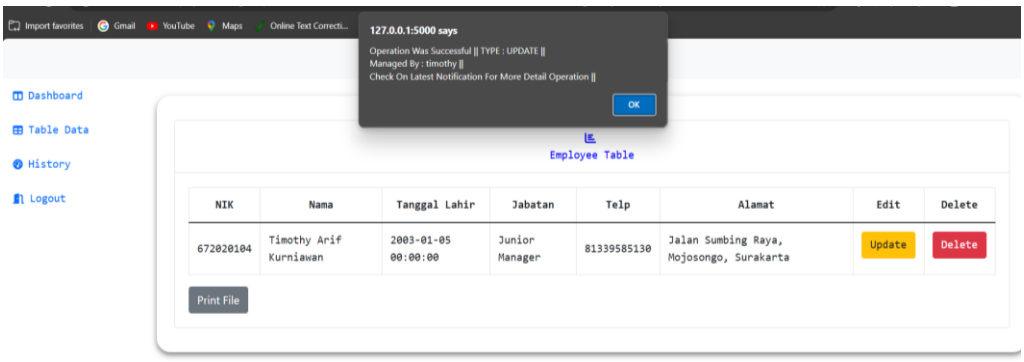
Gambar 8 merupakan tampilan dari hasil ketika *user* melakukan *print* data yang menghasilkan *output* berupa file excel yang memuat data-data. Data yang didapat berasal dari *collection* pada database. *Output* berupa file excel tersebut merupakan implementasi dari penggunaan *library* openpyxl dengan salah satu fungsinya yaitu ExcelWriter.

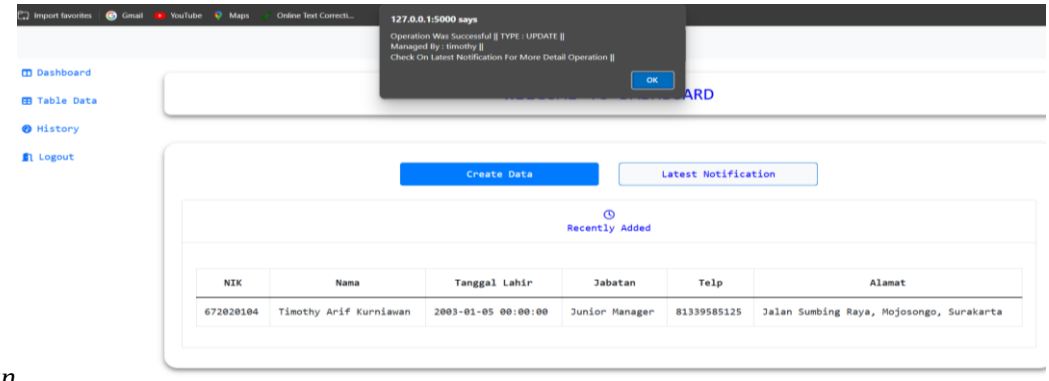
JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 2, Juni 2025, Pp. 1460-1473



3) Real-time Notification

Gambar 9 dan 10 merupakan tampilan 2 tab browser yang memuat tampilan aplikasi. Pada kedua gambar tersebut merupakan bentuk notifikasi yang *realtime* apabila pada browser 1 melakukan pengelolaan data maka akan memunculkan notifikasi pada browser 1 dan browser 2 secara langsung atau *realtime* dan apabila browser 2 melakukan pengelolaan data maka akan memunculkan juga notifikasi singkat pada browser 1 dan browser 2 secara *realtime.* Notifikasi yang terjadi secara *realtime* tersebut merupakan implementasi dari penggunaan websokcet.





4) Halaman History

Gambar 11 merupakan tampilan halaman *history* yang memuat riwayat dari proses pengelolaan data. Data yang ditampilkan pada halaman *history* didapatkan dari aktivitas *log* pada *collection* yang terjadi selama proses pengelolaan data berlangsung. Aktivitas *log* pada *collection* yang didapat berasal dari penggunaan *code* collection.watch() yang merupakan penerapan dari MongoDB Change Streams.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 2, Juni 2025, Pp. 1460-1473

Gambar 11. Tampilan History

C. *Testing Sistem*

Dalam pengembangan sebuah sistem aplikasi perlu adanya aktivitas *testing* terhadap aplikasi yang sedang dikembangkan tersebut. Hal ini bertujuan untuk mengetahui kelebihan maupun kekurangan dari aplikasi seperti *error* ataupun *bug*. Dalam melakukan proses *testing* digunakkan *Black-Box Testing* untuk proses pengujian sistem aplikasi. *Black-Box Testing* merupakan pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program untuk mengetahui apakah fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Tabel VI adalah tabel proses *testing* yang dilakukan menggunakkan metode *Black-Box Testing*.

TABEL VI
BLACKBOX TESTING

No.	Action	Hasil Yang Diharapkan	Output
1.	Input username dan password, kemudian menekan tombol login	System akan melakukan check kevalidan username dan password, lalu akan berpindah ke halaman dashboard	Valid
2.	Mengosongi field form data pada pengelolaan insert data, kemudian menekan tombol create	System akan memberikan peringatan untuk mewajibkan user tidak boleh mengosongi data	Valid
3.	Input data yang sama dengan collection khususnya bagian field NIK, kemudian menekan tombol create	System akan menolak insert data dan memberikan alert kepada user bahwa data dengan NIK yang diinput sudah terdapat di collection	Valid
4.	Melakukan proses create, update, dan delete, kemudian menekan tombol dari masing – masing proses tersebut	System akan melakukan aktivitas sesuai dengan proses yang dilakukan dan akan menampilkan notifikasi secara realtime sesuai dengan proses yang dilakukan	Valid
5	Apabila belum login dan sudah logout, melakukan “tembak” route aplikasi pada url	System akan menolak untuk berpindah pada halaman yang memiliki route yang dimaksud dan tetap akan berada pada halaman login	Valid
6	Melakukan task Print Data	System akan melakukan cetak dan unduh file data dalam bentuk excel	Valid
7	Menekan tombol update pada data yang dipilih	System akan menampilkan form update data dengan masing-masing field sudah terisi dengan data sebelumnya	Valid
8	Menekan tombol page History	System akan menampilkan data aktivitas log dari collection beserta dengan user yang telah melakukan aktivitas log pengelolaan data tersebut	Valid
9	Melakukan task Latest Notification	System akan menampilkan notifikasi secara detail mengenai aktivitas pengelolaan data terbaru	Valid
10	Melakukan proses create, update, dan delete pada data	System akan melakukan auto refresh data secara otomatis antar user	Tidak Valid

Pada Tabel VI merupakan bentuk penerapan functional testing yang menggunakan metode Black-Box Testing. Dari hasil testing pada tabel VI dapat dilakukan perhitungan persentase keberhasilan fungsi aplikasi, yaitu dengan rumus [19] berikut:

$$Persentase\ Keberhasilan = \frac{Jumlah\ Action\ Berhasil}{\text{Jumlah Total Action}}\ x\ 100\%$$

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 2, Juni 2025, Pp. 1460-1473

Diketahui untuk jumlah *action* yang berhasil sebanyak 9 *action* sedangkan untuk jumlah total *action* adalah 10, maka dengan perhitungan rumus hasil persentase keberhasilan fungsionalitas aplikasi, yaitu:

$$Persentase\ Keberhasilan = \frac{9}{10}\ x\ 100\% = 90\%$$

Hasil dari pengujian pada aplikasi menggunakkan metode Black-Box Testing dengan teknik functional testing mendapatkan hasil dengan persentase keberhasilan dari fungsionalitas dari sistem aplikasi sebesar 90%. Dapat disimpulkan bahwa masih terdapat 1 *action* yang masih belum dapat berjalan sesuai dengan *output* yang ditentukan yaitu *action* pengelolaan data dengan sistem *auto refresh* data secara otomatis tanpa *user* perlu menekan tombol *refresh*. Dikarenakan action tersebut masih belum berjalan sesuai dengan *output* yang ditentukan maka agar antar *user* ingin melihat data terbaru pada tampilan antar *user* perlu menekan tombol *refresh* terlebih dahulu meskipun notifikasi *message* pengelolaan data yang ditampilkan serta *log* dari *collection* bersifat *realtime*.

Pada tahap pengujian, selain menggunakkan metode Black-Box Testing diterapkan juga metode Load Testing. Load Testing adalah metode testing yang menerapkan teknik *performance* yang dimana respon sistem diukur dalam berbagai *load condition* seperti sistem dapat bekerja dalam kondisi beban *task* yang beragam misalnya jumlah *user* yang mengakses aplikasi. Tabel VII adalah tabel proses testing yang dilakukan menggunakkan metode Load Testing.

TABEL VII
LOAD TESTING

No.	User	Loop	Label	Connect Time
1.	5 User	2	HTTP Request (GET)	1 – 3 ms
2.	10 User	2	HTTP Request (GET)	0 – 1 ms
3.	15 User	2	HTTP Request (GET)	0 – 2 ms
4.	20 User	2	HTTP Request (GET)	0 – 2 ms

Berdasarkan tabel VII yang merupakan hasil dari proses testing pada aplikasi menggunakkan metode Load Testing maka dapat disimpulkan bahwa jumlah *user* yang mengakses aplikasi secara bertahap (sesuai batasan) tidak mempengaruhi *performance connect time* dari aplikasi karena rata-rata waktu yang dibutuhkan *user* untuk *connect* dengan aplikasi berada pada *range* antara 0-3 ms meskipun jumlah *user* tiap tahapan berbeda.

Dengan penelitian yang telah dilakukan [20], sebuah aplikasi manajemen klinik berbasis web dengan notifikasi *real-time* berhasil dikembangkan. Dalam penelitian tersebut, notifikasi *real-time* diterapkan melalui WhatsApp sebagai media pemberitahuan dari klinik kepada wali pasien, sehingga dapat membantu pegawai klinik dalam mengelola data pasien. Namun, dalam sebuah sistem diperlukan data *log* sebagai bentuk pencatatan aktivitas. Dengan *log*, pengguna dapat melakukan pemantauan bahkan pelacakan apabila terdapat data yang *error* atau tidak valid.

Berdasarkan proses implementasi MongoDB Change Streams sebagai teknologi untuk menampilkan *log* secara otomatis tanpa perlu *custom log* manual pada *collection* beserta penerapan Websocket sebagai *tools* yang digunakkan untuk *realtime notification* dengan melalui proses testing maka menghasilkan sebuah aplikasi yang bertujuan untuk mengelola data khususnya data-data pribadi yang dilengkapi dengan berbagai macam fitur utama yaitu menampilkan riwayat pengelolaan data yang diambil dari *log* dari *collection* dan dapat menampilkan notifikasi *message* kepada *user* dan antar *user* setelah melakukan pengelolaan data secara *realtime*. Sehingga aplikasi yang dilengkapi fitur-fitur tersebut diharapkan dapat membantu *user* dalam pengelolaan data serta melakukan *tracking record* data sebagai bentuk penerapan keamanan dan pengawasan apabila terdapat data yang tidak valid atau bersifat mencurigakan.

IV. KESIMPULAN

Kesimpulan dari penelitian yang berjudul "Pengelolaan Perlindungan Data Pribadi Menggunakkan MongoDB Change Streams Untuk Sistem Notifikasi Real-Time" bahwa pemanfaatan fitur MongoDB Change Streams bagi proses pengelolaan data dapat membantu *user* untuk mengetahui keseluruhan riwayat operasi yang telah dilakukan pada sebuah *collection* karena dengan menggunakkan MongoDB Change Streams dapat menampilkan *log* dari aktivitas pengelolaan data pada sebuah *collection* serta dengan pemanfaatan websocket juga dapat membantu sistem untuk mengirimkan *message* dalam bentuk notifikasi yang terjadi secara *realtime* sehingga setiap proses pengelolaan data selain memiliki *log* sebagai riwayat pengelolaan namun juga setiap proses pengelolaan data yang dilakukan akan menampilkan notifikasi *message* secara *realtime*. Maka dengan penggunaan MongoDB Change Streams dan Websocket pada aplikasi dapat membantu mencegah proses pengelolaan data agar terhindar dari kegiatan pencurian data khususnya pembobolan data seperti mengubah dan menghapus data tanpa konfirmasi karena seluruh aktivitas pengelolaan data akan tercatat pada riwayat pengelolaan yang berasal dari *log* serta setiap proses pengelolaan data akan menampilkan notifikasi *message* secara *realtime* sehingga setiap proses pengelolaan data dapat terpantau dengan jelas.

DAFTAR PUSTAKA

[1] H. Wijayanto, D. Daryono, and S. Nasiroh, "Analisis Forensik Pada Aplikasi Peduli Lindungi Terhadap Kebocoran Data Pribadi," *J. Teknol. Inf. dan Komun.*, vol. 9, no. 2, p. 11, 2021, doi: 10.30646/tikomsin.v9i2.572.

[2] K. Kaur, I. Gupta, and A. K. Singh, "A Comparative Evaluation of Data Leakage/Loss Prevention Systems (DLPS)," pp. 87–95, 2017, doi: 10.5121/csit.2017.71008.

[3] M. Raihan, "Perlindungan Data Diri Konsumen Dan Tanggungjawab Marketplace Terhadap Data Diri Konsumen (Studi Kasus: Kebocoran Data 91 Juta Akun Tokopedia)," *J. Inov. Penelit.*, vol. 3, no. 10, pp. 7847–7856, 2023.

[4] R. Maududy and D. Rizal Nursyamsi, "Pengembangan Real-Time Monitoring dan Data Logging Berbasis Web Pada Proses Robot Painting untuk Meningkatkan Efisiensi Produksi," *Informatics Digit. Expert*, vol. 5, no. 2, pp. 89–94, 2024, doi: 10.36423/index.v5i1.1586.

[5] K. Aryasa and Y. Elly Kurniawan, "Implementasi Firebase Realtime Database Untuk Aplikasi Pemesanan Menu Berbasis Android," *Semin. Nas. Sist. Inf. dan Teknol. Inf. 2019*, pp. 71–78, 2019.

[6] Y. Darnita and M. Muntahanah, "Rancang Bangun Aplikasi Mobile Penjadwal Perkuliahan Dengan Firebase Dengan Realtime Notification," *Pseudocode*, vol. 8, no. 1, pp. 58–65, 2021, doi: 10.33369/pseudocode.8.1.58-65.

[7] N. Amalia, B. Ismanto, M. F. Kurniawan, and D. J. S. HS, "Implementasi Notifikasi Realtime pada Aplikasi Informasi Akademik Berbasis Android menggunakan Metode Agile," *KLIK Kaji. Ilm. Inform. dan Komput.*, vol. 3, no. 2, pp. 121–127, 2022, [Online]. Available: https://djournals.com/klik

[8] N. Hikmah, A. Suradika, and R. A. Ahmad Gunadi, "Metode Agile Untuk Meningkatkan Kreativitas Guru Melalui Berbagi Pengetahuan (Knowledge Sharing) (Studi Kasus: Sdn Cipulir 03 Kebayoran Lama, Jakarta," *Instruksional*, vol. 3, no. 1, p. 30, 2021, doi: 10.24853/instruksional.3.1.30-39.

[9] Shama PS and Shivamanth A, "A Review of Agile Software Development Methodologies," *Int. J. Adv. Stud. Comput. Sci. Eng.*, vol. 4, no. 11, pp. 1–6, 2015, [Online]. Available: www.ijascse.org

[10] S. Naik, R. D. Dandagwhal, C. N. Wani, and S. K. Giri, "A review on various aspects of auxetic materials," *AIP Conf. Proc.*, vol. 2105, no. 05, pp. 90–92, 2019, doi: 10.1063/1.5100689.

[11] A. Maharani, "Perancangan Data Base Kasir Dan Persediaan Barang Menggunakan Mongodb," *J. Data Min. dan Sist. Inf.*, vol. 3, no. 1, p. 32, 2022, doi: 10.33365/jdmsi.v3i1.1941.

[12] R. Maulana, "Implementasi Web Socket Pada Sistem Pelayanan Pasien Rawat Jalan Pada Puskesmas Kabupaten Gowa," *J. INSTEK (Informatika Sains dan Teknol.*, vol. 6, no. 1, p. 130, 2021, doi: 10.24252/instek.v6i1.20555.

[13] Z. Ramadhan, S. R. Akbar, and G. E. Setyawan, "Implementasi Sistem Monitoring Daya Listrik Berbasis Web dan Protokol Komunikasi Websocket," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 3, no. 1, pp. 205–211, 2019.

[14] A. Fahrezi, F. N. Salam, G. M. Ibrahim, R. R. Syaiful, and A. Saifudin, "Pengujian Black Box Testing pada Aplikasi Inventori Barang Berbasis Web di PT. AINO Indonesia," *Log. J. Ilmu Komput. dan Pendidik.* , vol. 1, no. 1, pp. 1–5, 2022, [Online]. Available: https://journal.mediapublikasi.id/index.php/logic

[15] A. C. Praniffa, A. Syahri, F. Sandes, U. Fariha, Q. A. Giansyah, and M. L. Hamzah, "Pengujian Black Box Dan White Box Sistem Informasi Parkir Berbasis Web Black Box and White Box Testing of Web-Based Parking Information System," *J. Test. dan Implementasi Sist. Inf.*, vol. 1, no. 1, pp. 1–16, 2023.

[16] A. Fikri, H. Hozairi, and M. Muhsi, "Analisis Pengujian Sistem Informasi Mui Kabupaten Pamekasan Menggunakan Metode Blackbox Functional Testing," *J. Mnemon.*, vol. 5, no. 2, pp. 158–164, 2022, doi: 10.36040/mnemonic.v5i2.4803.

[17] D. I. Permatasari, "Pengujian Aplikasi menggunakan metode Load Testing dengan Apache JMeter pada Sistem Informasi Pertanian," *J. Sist. dan Teknol. Inf.*, vol. 8, no. 1, p. 135, 2020, doi: 10.26418/justin.v8i1.34452.

[18] W. Tejaya, S. Rahman, and A. Munir, "Pengujian Website Invitees Menggunakan Metode Load Testing Dengan Apache Jmeter," *KHARISMA Tech*, vol. 18, no. 1, pp. 99–112, 2023, doi: 10.55645/kharismatech.v18i1.305.

[19] J. Abraham, I. E. Ismail, S. Kom, and M. Kom, "Unit Testing dan User Acceptance Testing pada Sistem Informasi Pelayan Kategorial Pelayanan Anak," pp. 1–7, 2021.

[20] S. Hakim, M. A. Yaqin, and M. Jasri, "Sistem Informasi Manajemen di Klinik Az-Zainiyah Berbasis Web Terintegrasi Notifikasi Real Time Menggunakan WhatsApp," *J. Res. Investig. Educ.*, pp. 17–25, 2024, doi: 10.37034/residu.v2i1.165.