Implementasi Aplikasi *Backend* untuk Sistem Informasi Deteksi Cacat Produksi Kain Tekstil Menggunakan PHP dengan Kerangka Kerja Laravel

Kumara Ris Panji Bagaskara, Nur Ichsan Utama, dan Oktariani Nurul Pratiwi

Program Studi S1 Sistem Informasi
Universitas Telkom, Bandung
E-mail: panjibagas@student.telkomuniversity.ac.id,
nichsan@telkomuniversity.ac.id, onurulp@telkomuniversity.ac.id

Abstrak

Produksi tekstil terus meningkat seiring dengan investasi modal asing dan tingginya permintaan akan kain tekstil. Dalam menghadapi tantangan ini, diperlukan teknologi mutakhir untuk memastikan kualitas produksi kain. Fokus penelitian ini beralih kepada pengembangan *backend* sistem untuk menyajikan hasil dari model *machine learning* klasifikasi objek yang mendeteksi cacat. Kerangka kerja Laravel dan bahasa PHP dipilih sebagai landasan pengembangan, MariaDB berperan sebagai *database*, serta *Waterfall* sebagai metode pengembangannya. Penelitian ini bertujuan meningkatkan akurasi dan resolusi dalam deteksi cacat, menciptakan standar kualitas produk yang jelas, dan menggantikan penggunaan inspeksi manual yang rentan terhadap subjektivitas dan kesalahan. Dengan PT Gracia Mega Karya yang menjadi fokus utama penelitian ini, penelitian ini diharapkan dapat cacat memberikan kontribusi signifikan dalam meningkatkan efisiensi dan kualitas produksi kain tekstil. Sebagai hasil, dikembangkan sebuah *website* laporan dinamakan "*Defector*" yang menyajikan informasi visual terkait deteksi cacat dalam produksi kain tekstil. *Website* ini dirancang untuk mempermudah pengambilan keputusan terkait perbaikan dan pengembangan proses produksi secara efektif.

Kata kunci : *Backend*, Cacat, Laravel, MariaDB, PHP, Tekstil

Pendahuluan

Pengembangan *backend* sangat penting dalam pengembangan *website* karena berperan dalam membangun sistem yang kuat dan efisien untuk mendukung kualitas layanan dan informasi berbasis *web*. Pengembangan *backend* mencakup pengelolaan konten dinamis, basis data, serta integrasi dengan antarmuka *frontend*, yang memastikan kelancaran operasional dan pengelolaan informasi yang efektif. Hal ini memungkinkan transformasi digital yang lebih baik dan peningkatan kualitas layanan di era digital. *Back*end yang kokoh juga memungkinkan sistem untuk menangani tantangan seperti keamanan, skalabilitas, dan kemajuan teknologi yang cepat [1].

Dalam konteks yang lebih spesifik, seperti industri tekstil, pengembangan *backend* memainkan peran penting dalam mendukung implementasi model *machine learning*, termasuk *deep learning*, yang digunakan untuk mendeteksi cacat pada kain. *Backend* yang kuat dan efisien memungkinkan pengolahan data yang besar dan kompleks dari sensor atau kamera secara cepat dan konsisten, yang sangat penting untuk analisis *real-time*. Dengan pengelolaan data yang optimal dan keamanan yang terjamin, model *deep learning* dapat diintegrasikan dengan lebih lancar ke dalam sistem produksi, meningkatkan kualitas hasil dengan mengurangi jumlah produk cacat. *Backend* juga mendukung skalabilitas sistem, memungkinkan pengolahan data dalam jumlah besar seiring meningkatnya volume produksi dan permintaan pasar, serta menjaga keamanan data sensitif terkait proses produksi [2].

Berdasarkan data dari Badan Statistika Nasional, industri tekstil di Kabupaten Bandung selama empat tahun terakhir menunjukkan angka yang tinggi dalam proyek penanaman modal asing, dengan peningkatan realisasi investasi pada tahun 2021-2022 sebesar 145,41%. Ini mencerminkan upaya yang dilakukan oleh pemerintah Jawa Barat untuk meningkatkan produksi tekstil, yang bertujuan untuk memberikan kontribusi signifikan ter-

DOI : http://dx.doi.org/10.32409/jikstik.23.4.3662

hadap perekonomian nasional. Peran krusial industri tekstil di Jawa Barat tergambar jelas, mengingat wilayah ini merupakan pusat produksi tekstil dan garmen di Indonesia. Kontribusi besar sektor industri tekstil di Jawa Barat tidak hanya berdampak positif pada perekonomian regional dan nasional, tetapi juga menciptakan peluang pekerjaan bagi penduduk setempat [3].

Selain itu, pengembangan *backend* yang efektif dalam sistem deteksi cacat berbasis *web* menawarkan berbagai keuntungan dibandingkan metode manual tradisional. Inspeksi visual konvensional sering kali kurang konsisten dan subjektif, sementara sistem otomatisasi yang didukung oleh *backend* yang handal mampu memberikan hasil yang lebih akurat dan konsisten [4].

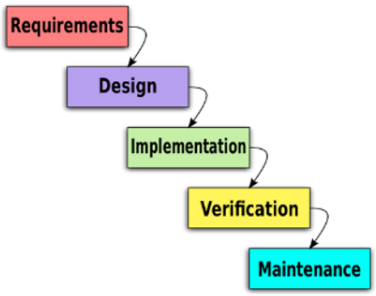
Kesalahan manusia dalam proses manual dapat berdampak negatif pada kualitas produk, sementara otomatisasi dapat mengurangi risiko tersebut dan meningkatkan efisiensi produksi. Dalam industri tekstil, penggunaan sistem berbasis *web* tidak hanya mengurangi kesalahan manusia tetapi juga meningkatkan presisi dalam otomatisasi, memberikan keunggulan kompetitif yang signifikan bagi perusahaan [5].

Sejalan dengan konsep keseluruhan dari *website Defector*, pendataan otomatis secara *real-time* di *website* menawarkan banyak keuntungan dibandingkan dengan pendataan manual di atas kertas, terutama dalam hal meningkatkan kualitas dan keandalan data. Data yang dimasukkan secara manual sering kali rentan terhadap kesalahan manusia seperti ketidaktepatan, inkonsistensi, dan kesalahan pengetikan, yang dapat menyebabkan masalah serius, termasuk kesalahan operasional dan konsekuensi hukum yang tidak diinginkan. Misalnya, kesalahan data di institusi keuangan dapat menyebabkan kerugian finansial yang signifikan atau bahkan mengakibatkan konsekuensi hukum yang berat [6]. Selain itu, proses pendataan manual cenderung tidak efisien dan memakan waktu, serta tidak selalu dapat diotomatisasi dengan mudah karena kompleksitas tertentu dalam prosedur bisnis. Sebaliknya, pendataan otomatis di *website* dapat mengurangi risiko kesalahan tersebut dengan menerapkan validasi data dan pemeriksaan otomatis secara *real-time*, memastikan data yang dihasilkan lebih akurat dan konsisten. Pendataan otomatis juga memungkinkan pembaruan data secara langsung, mendukung pengambilan keputusan yang lebih cepat dan berbasis data, yang pada akhirnya memberikan keunggulan kompetitif dan meningkatkan kepatuhan terhadap peraturan [7].

Dengan teori yang sama mengenai kebutuhan akan peningkatan efisiensi dan presisi ini, penelitian ini bertujuan untuk mengembangkan *backend* untuk sistem deteksi cacat otomatis berbasis *web* yang mampu mengintegrasikan teknik inspeksi visual secara *real-time*. Dengan menggunakan metode pengembangan *Waterfall* dan kerangka kerja Laravel berbasis PHP, penelitian ini bertujuan untuk mempercepat proses pengembangan dan mengatasi kompleksitas serta karakteristik unik dari setiap masalah yang dihadapi [8][9]. Selain itu, Laravel merupakan *platform* yang ideal untuk pengembangan *website* di industri tekstil karena menawarkan fitur lengkap seperti Eloquent ORM untuk pengelolaan data, *Blade Templating Engine* untuk tampilan dinamis, serta dukungan API untuk integrasi sensor dan visualisasi data dengan *Google Charts*. Laravel juga bersifat o*pen-source*, sehingga lebih hemat biaya, skalabel, dan didukung oleh komunitas yang besar untuk kemudahan pemeliharaan [10]. Diharapkan, sistem ini dapat menciptakan standar kualitas produk yang lebih baik dalam konteks pengawasan mutu industri tekstil, dengan menggantikan proses inspeksi manual yang subjektif dan repetitif, serta memberikan kontribusi signifikan dalam meningkatkan efisiensi dan kualitas produksi kain tekstil di PT Gracia Mega Karya. Dengan pendekatan ini, penelitian ini tidak hanya akan membuktikan keunggulan otomatisasi berbasis *web* dalam konteks deteksi cacat, tetapi juga memperkuat pentingnya pengembangan *backend* yang kuat dalam menghadapi tantangan modern di industri tekstil.

Metode Penelitian

Metode Pengembangan *Waterfall* Proses perancangan perangkat lunak dengan metode *Waterfall* adalah pendekatan sistematis yang bertujuan meningkatkan kualitas perangkat lunak [11]. Metode *Waterfall* memiliki kelebihan berupa proses kerja yang terstruktur, memungkinkan perencanaan kebutuhan dengan baik, dan menyediakan jadwal pengembangan yang lebih jelas [12].



Gambar 1: Metode Pengembangan Waterfall [13]

Dari Gambar 1 terdapat beberapa tahapan dalam metode *Waterfall* meliputi:

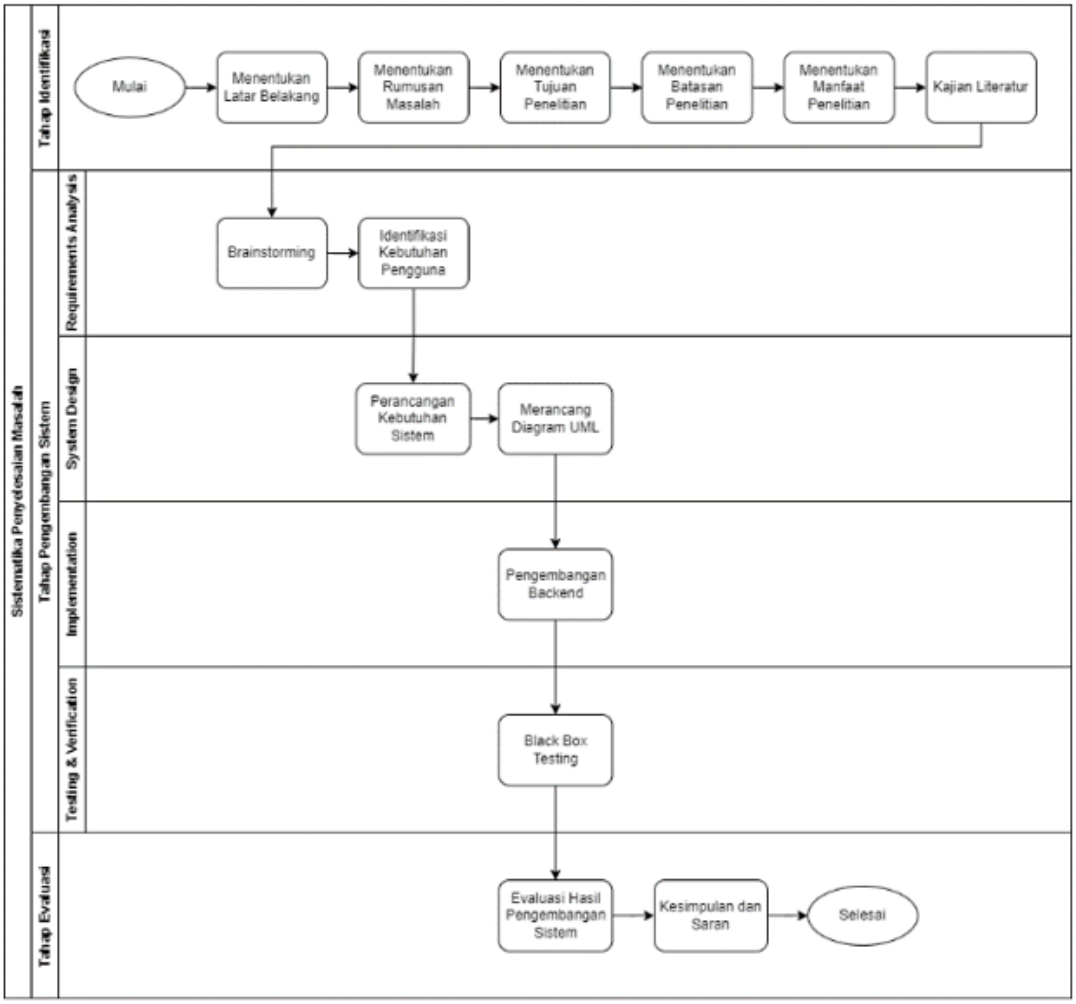
1. *Requirements*: Menganalisis kebutuhan sistem, baik fungsional maupun nonfungsional.
2. *Design*: Merancang sistem, termasuk database dan diagram UML.
3. *Implementation*: Mengubah desain menjadi sistem melalui coding, dalam hal ini menggunakan PHP dengan Laravel dan MariaDB.

4. *Verification*: Menguji sistem dengan pengujian black box untuk memastikan fungsionalitas berjalan baik.
5. *Maintenance*: Tahap pemeliharaan dilakukan setelah program diterapkan. Kesalahan atau bug yang ditemukan selama penggunaan dan pengujian mungkin memerlukan pembaruan kode [14].

Dalam penelitian ini, proses hanya dilaksanakan hingga tahap *Testing and Verification*. Hal ini disebabkan karena tahap *Maintenance* memerlukan penggunaan *website* dalam jangka waktu yang panjang, yang berada di luar cakupan dan durasi penelitian ini.

Sistematika Penyelesaian Masalah

Pada Gambar 2 menggambarkan sistematika pelaksanaan penelitian ini, metode yang digunakan adalah metode *Waterfall*, yang terdiri dari beberapa tahapan yang diterapkan selama pengerjaan tugas akhir. Pada Gambar 2, tahap awal dimulai dengan Tahap Identifikasi, di mana penelitian ini difokuskan pada penentuan latar belakang, perumusan masalah, tujuan penelitian, batasan penelitian, manfaat penelitian, dan kajian literatur. Selanjutnya, penelitian masuk ke Tahap Pengembangan Sistem dengan menerapkan metode *Waterfall*.



Gambar 2: Sistematika Penyelesaian Masalah

Tahap pertama adalah *Requirements Analysis*, yang melibatkan *brainstorming* dengan pihak PT Gracia Mega Karya untuk mengidentifikasi kebutuhan pengguna dan mendapatkan gambaran umum mengenai *website*. Tahap berikutnya, *System Design*, berfokus pada perancangan kebutuhan sistem dan menghasilkan diagram UML. Setelah itu, pada tahap *Implementation*, penulis mulai mengembangkan *website* untuk deteksi cacat.

Setelah tahap pengembangan selesai, dilakukan tahap *Testing and Verification*, di mana penulis menguji *website* yang telah dibuat menggunakan *Black Box Testing* untuk memastikan fungsionalitasnya sesuai dengan rencana pengembangan awal. Pada tahap akhir, yaitu Tahap Evaluasi, penulis mengevaluasi pengembangan sistem serta menyusun kesimpulan dan saran untuk laporan penelitian yang telah dibuat.

Alasan Pemilihan Metode

Dalam penelitian ini, penulis menerapkan metode pengembangan *website* dengan pendekatan *Waterfall*. Metode ini dipilih karena telah terbukti efektif dalam berbagai penelitian untuk mengembangkan sistem berbasis *web* dengan berbagai tujuan. Keunggulan utama dari metode *Waterfall* adalah pendekatannya yang sistematis dan berurutan, yang memungkinkan transformasi proses manual menjadi sistem terkomputerisasi yang lebih cepat dan efisien [15]. Selain itu, model *Waterfall* memiliki kelebihan karena kesederhanaannya dan kemudahan dalam pemahaman. Dengan sifatnya yang kaku, setiap fase dalam model ini dapat diselesaikan secara berurutan, sehingga mempermudah pengelolaan dan pengaturan tugas [16].

Proses pengembangan yang lebih terstruktur dalam *Waterfall* juga membantu menjaga kualitas perangkat lunak serta memudahkan proses pemeliharaan. Dari sisi pengguna, metode ini memberikan keuntungan dalam hal perencanaan dan persiapan kebutuhan data serta proses dari awal, sehingga pengembangan dengan model *Waterfall* dapat berjalan lancar tanpa masalah [17].

Hasil dan Pembahasan

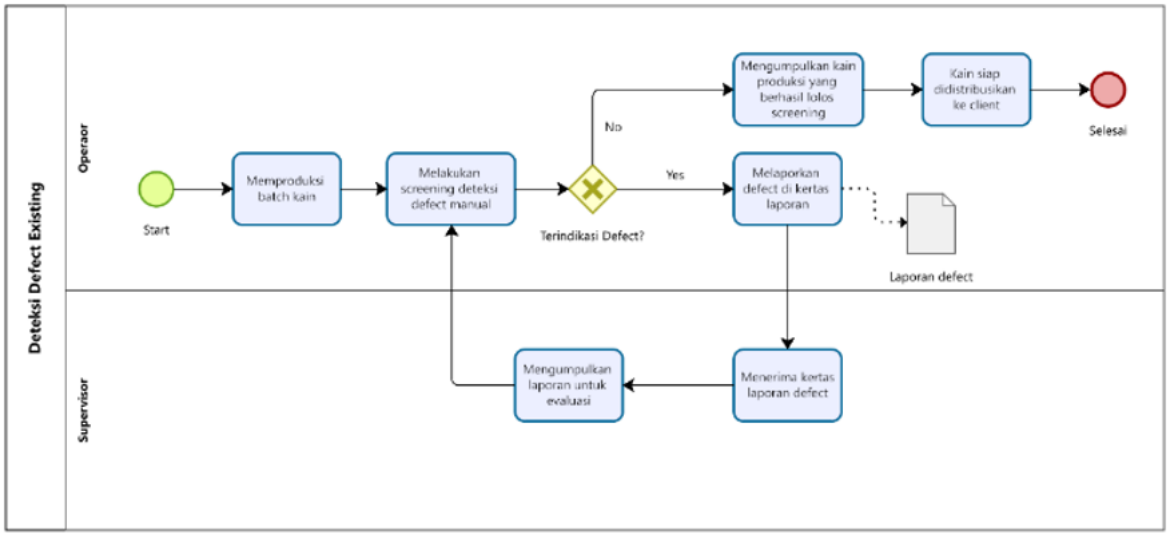
Analisis Proses Bisnis

Analisis proses bisnis dilakukan berdasarkan hasil diskusi yang telah dilakukan sebelumnya. Perancangan proses bisnis sangat penting untuk menggambarkan bagaimana alur kerja dari *website Defector* dalam mencapai tujuan utama yang diinginkan oleh *user*. Penggambaran proses bisnis dijadikan dua jenis, proses bisnis *existing* dan proses bisnis *targeting*. Proses bisnis *existing* mengelola data cacat pabrik untuk menggambarkan cara kerja dari proses pengelolaan data cacat yang sedang berlangsung di PT Gracia Mega Karya yang dilakukan oleh aktor operator dan supervisor, sebelum adanya *website Defector*.

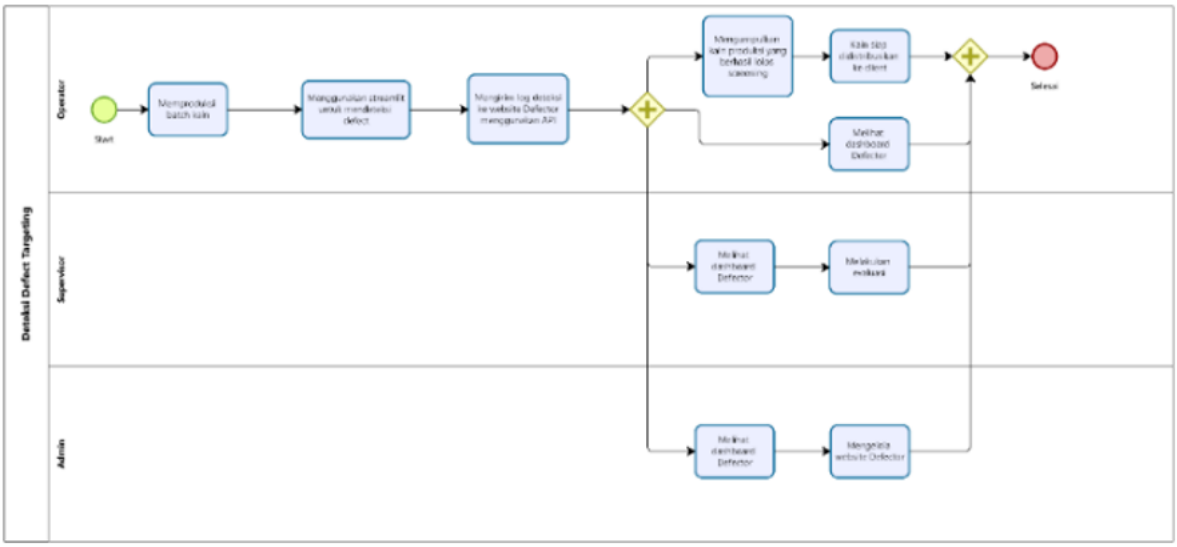
Pada Gambar 3 terlihat bahwa proses bisnis yang berjalan saat ini dimulai dengan operator yang memproduksi *batch* kain sambil melakukan deteksi cacat secara manual. Jika ditemukan cacat, operator mencatatnya di kertas laporan dan menyerahkannya kepada supervisor.

Supervisor kemudian mengumpulkan laporan tersebut untuk evaluasi. Jika tidak ditemukan cacat, operator melanjutkan proses *screening* hingga produksi kain untuk satu *ball* selesai. Setelah itu, kain yang lolos *screening* siap untuk didistribusikan ke *client*. Sementara itu, proses bisnis *targeting* mengelola data cacat untuk menggambarkan cara kerja dari proses pengelolaan data cacat yang ditargetkan setelah *website Defector* diciptakan.

Pada Gambar 4 terlihat bahwa proses bisnis yang diharapkan setelah pembuatan *website Defector* dimulai dengan operator memproduksi *batch* kain, diikuti dengan penerapan kerangka kerja Streamlit untuk mendeteksi cacat. Sistem deteksi ini memanfaatkan endpoint API untuk mengirim data cacat dalam bentuk *log* yang kemudian diubah menjadi tabel *log* di *website Defector*. Selanjutnya, operator dapat mengakses *dashboard Defector* dan mengumpulkan kain yang lolos proses *screening*, lalu mendistribusikannya ke client. Sementara itu, supervisor dapat mengakses *dashboard Defector* dan melakukan evaluasi terhadap data cacat yang dihasilkan oleh sistem deteksi. Sedangkan admin dapat melihat *dashboard* dan mengelola *website Defector*.



Gambar 3: Proses Bisnis Existing Deteksi Cacat



Gambar 4: Proses Bisnis Targeting Deteksi Cacat

Tabel 1: Daftar Aktor dan Perannya

No	Aktor	Deskripsi
1	Admin	Aktor yang memiliki hak akses penuh serta pengawasan terhadap jalannya *website* Defector. Dapat mengakses semua fitur yang tersedia yaitu *assign user*, membaca log data cacat dan *non*-cacat, mengunggah *batch* kain, serta menambah data *client* produksi kain
2	Supervisor	Aktor yang hadir untuk melakukan pengawasan terhadap proses produksi kain. Memiliki hak akses terhadap fitur *website* yang tidak jauh berbeda dengan admin. Ditujukan untuk mengawasi proses produksi kain yang dijalankan oleh operator
3	Operator	Aktor dengan tingkat peran paling dasar, hanya dapat melakukan pengelolaan terhadap *batch* kain, memilih operator dan *client* aktif untuk kelengkapan log data, serta membaca dan hapus data log cacat dan *non*-cacat.

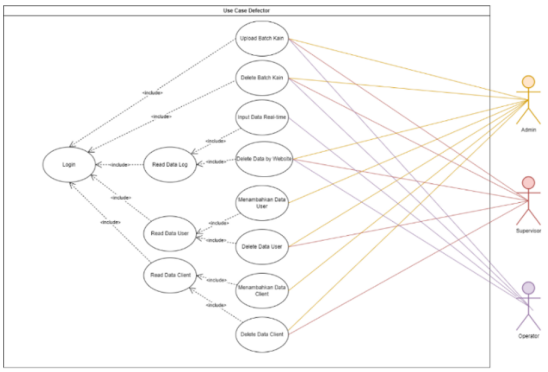
Dalam pengembangan *website Defector*, dapat diidentifikasi bahwa *website* memiliki tiga aktor didalamnya. Aktor pada *website* yaitu admin, supervisor, dan operator. Peran serta penjelasan mengenai aktor tersebut dijelaskan pada Tabel 1.

Desain Sistem

Pada tahap ini, hasil diskusi dengan pihak pabrik menghasilkan spesifikasi teknis yang rinci untuk pengembangan *website* deteksi cacat. Diagram visualisasi telah dibuat berdasarkan kebutuhan yang dikumpulkan sebelumnya, termasuk *use case diagram* untuk menggambarkan interaksi antara aktor dengan sistem, *sequence diagram* untuk memetakan urutan interaksi antar objek, *class diagram* untuk menampilkan struktur statis sistem, *deployment diagram* untuk menunjukkan interaksi perangkat keras dan perangkat lunak, serta ERD untuk menggambarkan hubungan antar entitas data. Diagram-diagram ini memfasilitasi komunikasi dengan tim *frontend* dan memastikan bahwa semua kebutuhan sistem dapat dipenuhi dalam implementasi akhir.

1. *Use Case Diagram*

Use case diagram pada Gambar 5 menggambarkan terdapat interaksi antara aktor yaitu admin, supervisor, dan operator. Diagram ini menunjukkan beberapa fungsi yang dapat dilakukan oleh *website* untuk mendukung proses pemantauan produksi kain yang bersifat cacat.

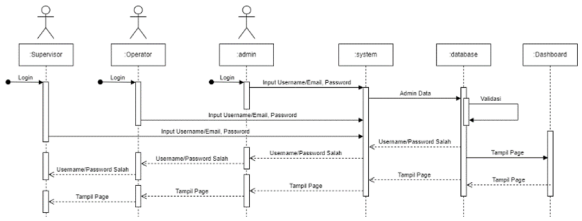


Gambar 5: Use Case Diagram

Admin memiliki kemampuan untuk mengelola *batch* kain, dan mengelola data lain seperti data *user* dan data *client*, termasuk menghapus data-data tersebut. Supervisor memiliki kewenangan yang hampir sama dengan admin, namun tidak dapat menambahkan *user* baru di *website Defector*. Sedangkan operator hanya dapat menginput dan menghapus data cacat *real-time* serta *batch* kain.

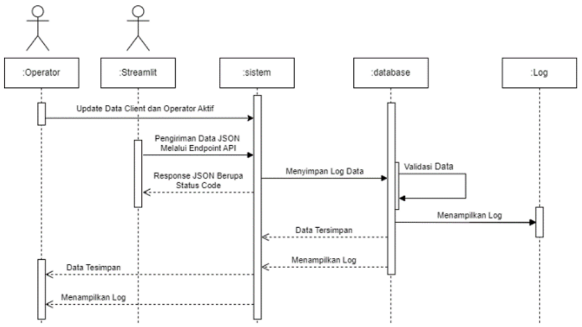
2. *Sequence Diagram*

Sequence Diagram menjadi tahap untuk mengilustrasikan urutan tindakan atau rangkaian langkah-langkah secara berurut sebagai respon terhadap berbagai fitur dalam sistem pengembangan *website*.



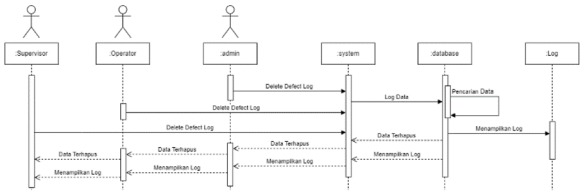
Gambar 6: Sequence Diagram Login

Gambar 6 menggambarkan proses yang terjadi ketika pengguna melakukan login ke dalam sistem. Aktor, yang bisa berupa admin, supervisor, atau operator, memulai dengan memasukkan informasi *login* seperti *username* dan *password*. Sistem kemudian memvalidasi data tersebut melalui interaksi dengan *database* pengguna. Jika data yang dimasukkan benar, sistem akan mengarahkan pengguna ke halaman *dashboard* yang sesuai dengan perannya, memberikan akses ke fitur yang relevan. Proses ini menunjukkan urutan respons sistem terhadap permintaan *login*.



Gambar 7: Sequence Diagram Menambahkan Data Log Real-Time

Gambar 7 menjelaskan alur interaksi ketika operator memasukkan data cacat secara *real-time*. Operator menetapkan client aktif serta operator yang bertanggung jawab untuk *batch* kain yang sedang diproduksi. Data *log* dikirimkan melalui API dan diterima oleh sistem. Kemudian, sistem menyimpan data tersebut ke dalam *database* dan menampilkannya di tabel *log* pada halaman *dashboard*. Diagram ini menggambarkan alur data mulai dari pengiriman hingga penyimpanan dan penyajian data secara *real-time*.

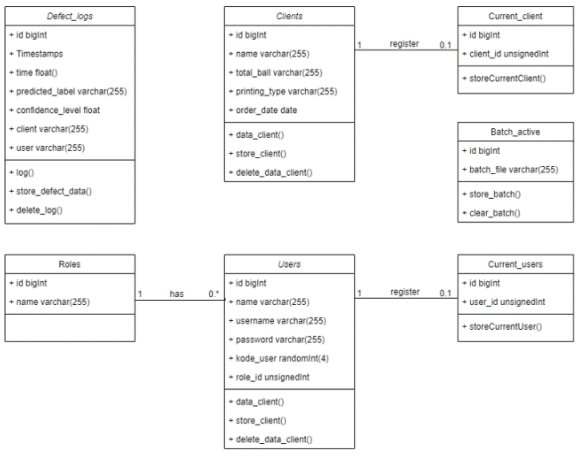


Gambar 8: Sequence Diagram Menghapus Data Log dari Tabel Log

Gambar 8 memaparkan urutan langkah yang terjadi saat seorang pengguna menghapus data *log* dari sistem. Pengguna terlebih dahulu memilih data yang ingin dihapus dari tabel *log* pada halaman *dashboard*. Setelah itu, permintaan dikirimkan ke sistem, dan sistem menghapus data yang dipilih dari *database*. Hasil dari penghapusan ini diperbarui dan ditampilkan kembali di tabel *log*, memastikan bahwa data yang tidak diperlukan lagi sudah berhasil dihapus dari sistem. Diagram ini memaparkan urutan langkah yang terjadi saat seorang pengguna menghapus data *log* dari sistem. Pengguna terlebih dahulu memilih data yang ingin dihapus dari tabel *log* pada halaman *dashboard*. Setelah itu, permintaan dikirimkan ke sistem, dan sistem menghapus data yang dipilih dari *database*. Hasil dari penghapusan ini diperbarui dan ditampilkan kembali di tabel *log*, memastikan bahwa data yang tidak diperlukan lagi sudah berhasil dihapus dari sistem.

3. *Class Diagram*

Pada Gambar 9, *Class Cliets* menunjukkan *client* yang memesan jasa produksi kain ke pabrik. *Class* ini memiliki atribut seperti '*id*', '*name*', '*total_ball*', '*printing_type*', dan '*order_date*'. *Clients* memiliki hubungan dengan *Current_client* melalui *foreign key* '*id*'. Di dalam *Current_client* terdapat atribut '*client_id*'. *Class* ini terhubung dengan *class Clients* melalui metode '*storeCurrentClient*'.

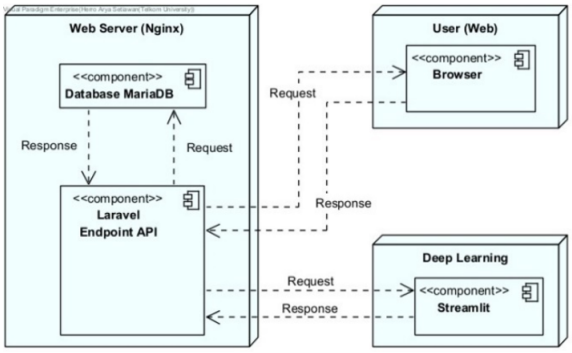


Gambar 9: Class Diagram

Kemudian terdapat *class Users* yang menunjukkan user yang dapat mengakses *website Defector*. *Class* ini memiliki atribut '*id*', '*name*', '*username*', '*password*', 'kode_*user*', dan '*role_id*'. *Class* ini memiliki relasi dengan *Current_users* yang menunjukkan *user* aktif di *website Defector*. *Current_users* memiliki atribut '*id*' dan '*user_id*', dan memiliki hubungan dengan *class Users* melalui *foreign key* '*id*'. Sedangkan *class Roles* yang juga berkaitan dengan *class Users* memiliki atribut '*id*' dan '*name*'. Sementara itu, terdapat dua *class independen* yang tidak memiliki relasi dengan *class* lainnya, yaitu *Defect_logs* dan *Batch_active*. *Defect_logs* memiliki atribut '*id*', '*timestamps*', '*time*', dan lainnya. Didukung oleh metode '*log*', '*store_defect_data*', dan '*delete_log*'. Sedangkan *class Batch_active* memiliki atribut '*id*' dan '*batch_file*', didukung oleh metode '*store_batch*' dan '*clear_batch*'.

4. *Deployment Diagram*

Deployment Diagram dapat membantu memvisualisasikan penempatan artefak perangkat lunak pada node perangkat keras dalam sistem yang sudah diterapkan (*deployed*).

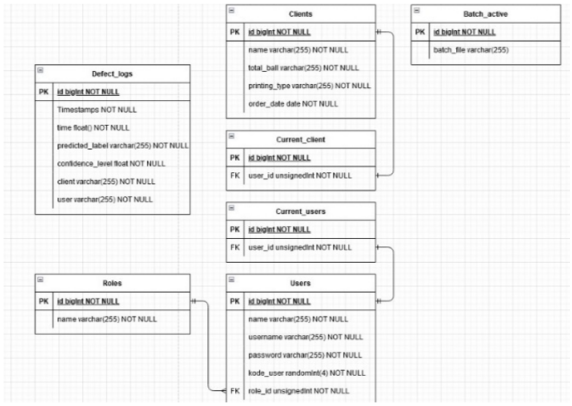


Gambar 10: Deployment Diagram

Pada Gambar 10, *User* berinteraksi dengan *website Defector* melalui *browser* untuk melakukan operasi CRUD pada data. *Website* ini berfungsi sebagai antarmuka *user*, sementara di sisi *server*, kerangka kerja Laravel dijalankan oleh *Web Server* menggunakan Nginx. Komponen Laravel seperti *routing*, *controller*, dan model mengatur logika aplikasi dan pengelolaan data, tanpa menggunakan *rate limiter* API. Komunikasi antara *browser* dan *Web Server* terjadi melalui HTTP, memungkinkan *user* untuk mengirim permintaan dan menerima respons terkait pengelolaan data. Data yang diolah oleh Laravel disimpan dalam *database* MariaDB. Sistem ini tidak menggunakan IoT Node, fokusnya adalah pengelolaan data secara manual oleh *user* melalui antarmuka *web*. Data dari kerangka kerja Streamlit dapat diunggah melalui *endpoint* API dan diolah oleh Laravel untuk ditampilkan dalam bentuk tabel *log*. Infrastruktur sistem ini mengandalkan server Nginx dan *database* MariaDB untuk mendukung operasi CRUD pada data.

7. *Entity Relationship Diagram*

Pada Gambar 11, menggambarkan struktur data dalam pengembangan sistem *website*, mencakup beberapa tabel penting. Tabel *Defect_logs* menyimpan catatan prediksi cacat pada kain dengan atribut seperti '*timestamps*', '*predicted_label*', '*confidence_level*', serta '*client*' dan '*user*' terkait.



Gambar 11: Entity Data Relationship (ERD)

Tabel *Clients* menyimpan informasi tentang *client* yang memesan jasa produksi kain, termasuk '*name*', '*total_ball*', '*printing_type*', dan '*order_date*', dengan hubungan ke *Current_client* yang mencatat *client* aktif saat ini. Tabel *Batch_active* mencatat '*batch_file*' kain yang sedang diproses. Tabel Roles mendefinisikan peran yang dimiliki oleh *user*, sementara tabel *Users* menyimpan informasi *user* yang dapat mengakses sistem, termasuk '*name*', '*username*', '*password*', 'kode_*user*' dan 'role_id' yang dihubungkan melalui *foreign key* ke tabel Roles. Tabel *Current_users* mencatat *user* yang saat ini aktif di sistem.

Implementasi

Tahap implementasi menyajikan hasil-hasil yang telah dicapai. Dari hasil ini, akan dievaluasi apakah *website* yang dikembangkan mampu memberikan *output* sesuai dengan tujuan yang diharapkan.

1. *Pseudocode*

Sesuai pada Gambar 12, fungsi '*store_defect_data()*' yang bertugas untuk memproses data *log* yang diterima dari kerangka kerja Streamlit dan kemudian menampilkannya pada *website Defector*. Fungsi ini diawali dengan proses autentikasi yang memverifikasi apakah token API yang digunakan sah. Jika token tidak valid, sistem akan mengembalikan respons *error* dengan status 400. Jika token valid,

sistem akan mengambil data dari *client* dan *user* aktif yang sudah dipilih dengan ID status 1.

```
FUNCTION store_defect_data(request, token)
    // Check if the provided token matches the application API token
    IF token equals application API token THEN
        // Retrieve current client and user data
        SET currentClient = Get CurrentClient with client information where ID is 1
        SET currentUser = Get CurrentUser with user information where ID is 1

        // Extract all data from the request
        SET data = request.getAllData()

        // Define validation rules
        SET rules = {
            'time_s' => 'required',
            'predicted_label' => 'required',
            'confidence_level' => 'required'
        }

        // Validate the data against the rules
        SET validated = Validate(data, rules)

        // Check if validation fails
        IF validated fails THEN
            RETURN JSON response with error messages and status 400
        END IF

        // Extract only specific data from the request
        SET data = request.getData(['status', 'client_id', 'current_user_id'])

        // If validation passes, proceed to save data
        IF validated passes THEN
            FOR EACH item IN data DO
                // Create a new DefectLog entry
                CREATE new DefectLog with the following fields:
                    'time_s' = item['time_s']
                    'predicted_label' = item['predicted_label']
                    'confidence_level' = item['confidence_level']
                    'client' = currentClient.client.name
                    'user' = currentUser.user.name
            END FOR

            // Return success response
            RETURN JSON response with success message and status 200
        ELSE
            // Return error response if data was not saved
            RETURN JSON response with error message and status 400
        END IF
    ELSE
        // Return error response if the token is invalid
        RETURN JSON response with error message and status 400
    END IF
END FUNCTION
```

Gambar 12: Pseudocode Metode Insert Data

Setelah itu, semua data deteksi cacat dikumpulkan dari Streamlit menggunakan fungsi 'request.getAllData()'. Data yang diambil mencakup kolom '*time_s*', '*predicted_label*', dan '*confidence_level*', yang semuanya bersifat *required*. Selanjutnya, data tersebut divalidasi, dan jika validasi gagal, sistem akan memberikan respons *error* dengan status 400. Setelah itu, data *log* cacat akan dibuat di *website* dengan kolom '*time_s*', '*predicted_label*', '*confidence_level*', '*client*', dan '*user*'. Fungsi ini berakhir dengan mengembalikan respons sukses dengan status 200 jika prosesnya berhasil, atau respons *error* dengan status 400 jika terjadi kesalahan.

2. *Verification dan Testing*

Black Box Testing dilakukan untuk memverifikasi fungsionalitas aplikasi tanpa melihat struktur kode internalnya. Pengujian ini berfokus pada *input* dan *output* dari *website*, memastikan setiap fungsi beroperasi sesuai dengan spesifikasi yang ditetapkan. Dengan proses pengujian yang dilakukan melalui tahap *staging*, diharapkan setiap *bug* atau masalah dalam aplikasi dapat diidentifikasi dan diperbaiki sebelum tahap *deployment*.

Pengujian fungsionalitas sistem dilakukan menggunakan metode *black box testing*. Pengujian ini bertujuan untuk mengevaluasi apakah fitur-fitur yang terdapat dalam sistem telah berjalan sesuai dengan spesifikasi yang ditetapkan.

Tabel 2: Pengujian Black Box

Nama Fitur	Output yang Diharapkan	Hasil	
		Sesuai	Tidak
Login	User yang sudah terdaftar dapat login	?	
Menu *Dashboard*	Masuk halaman *dashboard* beserta isinya	?	
Upload Gambar *Batch* Kain	Mengunggah *batch* motif kain di *dashboard*	?	
Hapus *Batch* Kain	Mengunggah *batch* motif kain di *dashboard*	?	
Jumlah Cacat Terdeksi	Menampilkan angka jumlah cacat yang terdeteksi	?	
Chart Total Cacat	Menampilkan jumlah total cacat dalam bentuk *chart*	?	
Menu Log	Masuk menu log	?	
Pilih Operator Aktif	Memilih operator aktif dalam pengerjaan produksi kain dengan bentuk *dropdown*	?	
Pilih Client Aktif	Memilih *client* aktif dengan bentuk *dropdown*	?	
Import Data JSON melalui *endpoint* API	Mengimport operator aktif, *client* aktif, dan log data dan mengubahnya menjadi tabel log	?	
Delete Data Log	Menghapus data log	?	
Menu Data *User*	Masuk menu data *user*	?	
Tambah *User*	*User* ditambahkan	?	
Hapus *User*	*User* terhapus	?	
Search User	Search data *user* dengan nama, jabatan, dan kode	?	
User Columns	Menampilkan data *user* berdasarkan nama, jabatan, dan kode	?	
Jumlah *User Columns*	Menampilkan data *user* dengan jumlah 10,25,50 baris per halaman	?	
Menu Data *Client*	Masuk menu data client	?	
Tambah *Client*	*Client* ditambahkan	?	
Hapus *Client*	*Client* terhapus	?	
Search Client	Search data *client* dengan nama, *ball*, *printing*, dan tanggal *order*	?	
Client Columns	Menampilkan data *client* berdasarkan nama, *ball, printing,* dan tanggal *order*	?	
Jumlah *Client Columns*	Menampilkan data *client* dengan jumlah 10,25,50 baris per halaman	?	
Exit (Logout)	Keluar dari *website*	?	

Pada fitur *login*, sistem diuji untuk memastikan bahwa hanya pengguna yang telah terdaftar dapat

mengakses sistem. Hasil pengujian menunjukkan bahwa fungsi login berjalan dengan baik, memungkinkan pengguna yang terverifikasi untuk masuk ke dalam sistem.

Selanjutnya, pengujian pada menu dashboard menunjukkan bahwa pengguna dapat mengakses halaman *dashboard* dengan lancar dan mendapatkan informasi yang ditampilkan sesuai dengan yang diharapkan. Fitur unggah gambar batch kain juga diuji untuk memeriksa kemampuan sistem dalam menerima dan menyimpan gambar dengan benar, di mana pengujian ini menunjukkan hasil yang sesuai tanpa ada kendala dalam proses unggah gambar.

Fitur lain yang diuji adalah penghapusan *batch* kain dari sistem, di mana pengguna dapat menghapus data *batch* kain dengan benar. Selain itu, sistem diuji untuk memverifikasi penampilan jumlah cacat pada *batch* kain dan menampilkan data tersebut dengan tepat. Hasil pengujian menunjukkan bahwa jumlah cacat ditampilkan dengan benar sesuai dengan data yang ada.

Fitur *chart* total cacat yang menampilkan grafik jumlah cacat juga diuji dan menunjukkan hasil yang sesuai, di mana grafik yang ditampilkan sistem sesuai dengan data yang telah dikumpulkan. Terakhir, pengujian dilakukan pada fitur pengelolaan data pengguna dan *client*. Sistem berhasil menampilkan dan mengelola data pengguna serta *client* dengan baik dan sesuai dengan spesifikasi yang ditetapkan.

Penutup

Penelitian ini berhasil merancang *website Defector* berbasis PHP menggunakan kerangka kerja Laravel yang mengintegrasikan sistem deteksi cacat di PT. Gracia Mega Karya dengan metode pengembangan *Waterfall*. *Website* ini menampilkan data cacat yang diperoleh dari *deep learning* melalui kerangka kerja Streamlit, dengan kolom data seperti *time*, *predicted label* (*defect* atau *non-defect*), dan *confidence level*.

Hasil penelitian menunjukkan bahwa sistem ini mendukung deteksi cacat secara *real-time* pada produksi kain tekstil, sehingga mempermudah pendataan dan pemantauan hasil produksi, serta memungkinkan evaluasi proses produksi yang lebih efisien. Pengujian menggunakan metode *black box testing* membuktikan bahwa *website* ini efektif sebagai alat bagi pengguna untuk memantau proses deteksi cacat dalam produksi kain.

Peneliti memberikan dua saran terkait pengembangan *website* sistem deteksi cacat. Yang pertama yaitu untuk mempertimbangkan penggunaan kerangka kerja dan metode pengembangan lainnya untuk penerapan membangun *website* lebih lanjut. Seperti halnya kerangka kerja Django dan metode Agile, serta mengembangkan sistem yang mampu mendeteksi jenis cacat secara lebih rinci. Hal ini akan memungkinkan *user* atau peneliti untuk melakukan analisis yang lebih mendalam terhadap penyebab cacat dan memungkinkan evaluasi perusahaan yang lebih komprehensif.

Daftar Pustaka

[1] Grady Andersen , "Exploring the Connection Between Back-End Development and Machine Learning ", MoldStud, Januari 2024.

[2] Tamás Czimmermann,Gastone Ciuti, Mario Milazzo, Marcello Chiurazzi,Stefano Roccella,Calogero Maria Oddo and Paolo Dario, "Visual-based defect detection and classification approaches for industrial applications—A Survey", Sensors, 20(5):1459, https://doi.org/10.3390/s20051459, 2020.

[3] Z. Alim dan S. E. Fitria, "Analisis Pengaruh Antara Relative Advantage dan Competitive Pressure Terhadap Adoption E-Commerce pada UMKM di Kota Bandung (Studi Pada Kawasan Tekstil Cigondewah) ", e-Proceeding of Management : Vol.7, No.1 April 2020

[4] A. Yuni Kristanto dan R. Rumita, "Analisis Penyebab Cacat Kain dengan Menggunakan Metode Failure Mode and Effect Analysis (FMEA) dan *Fault Tree Analysis* (FTA)", Industrial Engineering Online Journal, vol. 5, no. 1, 2016.

[5] G. Cöşkun and H. F. Akpinarli, "*C*o-design approach in textile printing and a case study of an e-commerce company website", Tekstil ve Muhendis, vol. 26, no. 116, hlm. 415–430, doi: 10.7216/1300759920192611613, 2019.

[6] M. Armstrong and S. Taylor, "Armstrong's handbook of human resource management practice: A guide to the theory and practice of people management", Kogan Page Publishers, 2023.

[7] C. Cichy and S. Rass, "An Overview of Data Quality Frameworks", IEEE Access, vol. 7, hlm. 24634–24648, doi: 10.1109/ACCESS.2019.2899751, 2019.

[8] D. Purnama Sari dan R. Wijanarko, "Implementasi Framework Laravel pada Sistem Informasi Penyewaan Kamera (Studi Kasus Di Rumah Kamera Semarang)", Informatika dan RPL, vol. 2, no. 1, hlm. 32–36, 2019.

[9] Agariadne Dwinggo Samala dan Bayu Ramadhani Fajri, "Rancang Bangun Aplikasi E-Sertifikat Berbasis Web Menggunakan Metode Pengembangan Waterfall", Jurnal Teknik Informatika, vol. 13, no. 2, 2020.

[10] Ashish Sutradhar, Md. Harunur Rashid Bhuiyan, Faria Tashnim Mazumder, Pritom Goswami, Saiful Islam Salim, Tarik Reza Toha, and Shaikh Md. Mominul Alam., "Devising a Dust and Noise Pollution Monitoring System for Textile Industry", dalam ACM International Conference Proceeding Series, Association for Computing Machinery, doi: 10.1145/3491371.3491381, hlm. 77–82, Des 2021.

[11] I. T. Maulana, "Penerapan Metode SDLC (System Development Life Cycle) Waterfall pada E-Commerce Smartphone", Jurnal Ilmiah Sistem Informasi dan Ilmu Komputer, vol. 2, no. 2, hlm. 1–6, 2022.

[12] W. Nugraha, M. Syarif, dan W. S. Dharmawan, "Penerapan Metode SDLC Waterfall dalam Sistem Informasi Inventori Barang Berbasis Desktop", JUSIM (Jurnal Sistem Informasi Musirawas), vol. 3, no. 1, hlm. 22–28, doi: 10.32767/jusim.v3i1.246, Jun 2018.

[13] D. Hughey, "The Traditional Waterfall Approach", diakses daring pada: https://www.umsl.edu/~hugheyd/is6840/waterfall.html, 2009.

[14] A. Ardiansyah dan S. Aji, "Pengembangan Sistem Informasi Penjualan Handphone Menggunakan Metode Waterfall", Jurnal Sistem Informasi Akuntansi, Vol. 1 No. 1, DOI: https://doi.org/10.31294/jasika.v1i1.386, 2021.

[15] H. Babatunde and D. Andrew Omideyi, "A web based mobile archival system using waterfall model approach", UNIZIK Journal of Engineering and Applied Sciences, vol. 3, no. 4, hlm. 1081–1101, 2024.

[16] L. Zhi Xin and S. Abd Ishak, "Design and Development of Web-Based Foodbank Management System by Using Waterfall Approach", Applied Information Technology and Computer Science, vol. 4, no. 1, hlm. 524–543, doi: 10.30880/aitcs.2023.04.01.030, 2023.

[17] Agus Deka Pujawan, Raihan Abiyu Rendra, Jaenal Arifin, and Cahyadi Agustin, "Design of Information System Vaccination Report Data Logging Web-Based Using Waterfall (Case Study at *Bandung* Health Office)", JATISI (Jurnal Teknik Informatika Dan Sistem Informasi), 9(1), 110-125, https://doi.org/10.35957/jatisi.v9i1.1440, 2022.