PENGEMBANGAN APLIKASI BERBASIS WEB SAFETY MANAGEMENT SYSTEM UNTUK MENINGKATKAN KESADARAN KARYAWAN TERHADAP SAFETY

Disusun oleh:

Crownsius Okgiria Sianipar

012201705020

Tugas Akhir

Diajukan kepada Fakultas Komputer

President University

Sebagai salah satu syarat untuk

Memperoleh gelar sarjana komputer

Bidang sistem informasi, fakultas ilmu komputer

Cikarang, Bekasi, Indonesia

2023

PENGEMBANGAN APLIKASI BERBASIS WEB SAFETY MANAGEMENT SYSTEM UNTUK MENINGKATKAN KESADARAN KARYAWAN TERHADAP SAFETY

Oleh

Crownsius Okgiria

012201705020

Disetujui:

Ronny Juwono, S.Pd., M. T Pembimbing Tugas Akhir	Ronny Juwono, S.Pd., M. T Kepala Program Studi Sistem Informasi

Rila Mandala, Ph. D
Dekan Fakultas Ilmu Komputer

STATEMENT OF ORIGINALITY

In my capacity as an active student of President University and as the author of the thesis/final project/business plan (underline that applies) stated below:

Name : Crownsius Okgiria Sianipar

Student ID : 012201705020

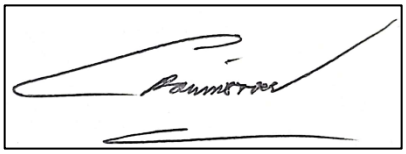
Study program : Information System

Faculty : Faculty of Computing

I Hereby declare that my thesis/final project/business plan entitled "**Pengembangan Aplikasi Berbasis Web Safety Management System Untuk Meningkatkan Kesadaran Karyawan Terhadap Safety**" is to the best of my knowledge and belief, an original piece of work based on sound academic principles. If there is any plagiarism detected in this thesis/final project/business plan, I am willing to be personally responsible for the consequences of these acts of plagiarism, and will accept the sanctions against these acts in accordance with the rules and policies of President University.

I also declare that this work, either in whole or in part, has not been submitted to another university to obtain a degree.

Cikarang, 26 September 2023



Crownsius Okgiria Sianipar

SCIENTIFIC PUBLICATION APPROVAL FOR ACADEMIC INTEREST

As an academic community member of the President's University, I, the undersigned:

Name : Crownsius Okgiria Sianipar

Student Id : 012201705020

Study program : Information System

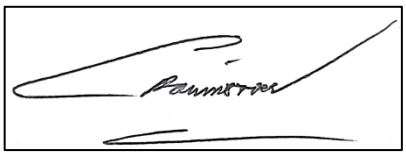
For the purpose of development of science and technology, certify, and approve to give President University a non-exclusive royalty-free right upon my final report with the title:

PENGEMBANGAN APLIKASI BERBASIS WEB SAFETY MANAGEMENT SYSTEM UNTUK MENINGKATKAN KESADARAN KARYAWAN TERHADAP SAFETY

With this non-exclusive royalty-free right, President University is entitled to converse, to convert, to manage in a database, to maintain, and to publish my final report. There are to be done with the obligation from President University to mention my name as the copyright owner of my final report.

This statement I made in truth.

Cikarang, 26 September 2023



Crownsius Okgiria Sianipar

ADVISOR APPROVAL FOR JOURNAL/INSTITUTION'S REPOSITORY

As an academic community member of the President's University, I, the undersigned:

Advisor Name : Ronny Juwono

Employee ID : 20160700622

Study Program : Information System

Faculty : Faculty of Computing

Declare that following thesis :

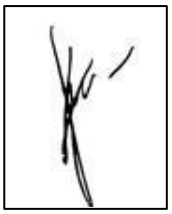
Title of thesis : Pengembangan Aplikasi Berbasis Web Safety Management System Untuk Meningkatkan Kesadaran Karyawan Terhadap Safety

Thesis author : Crownsius Okgiria Sianipar

Student ID : 012201705020

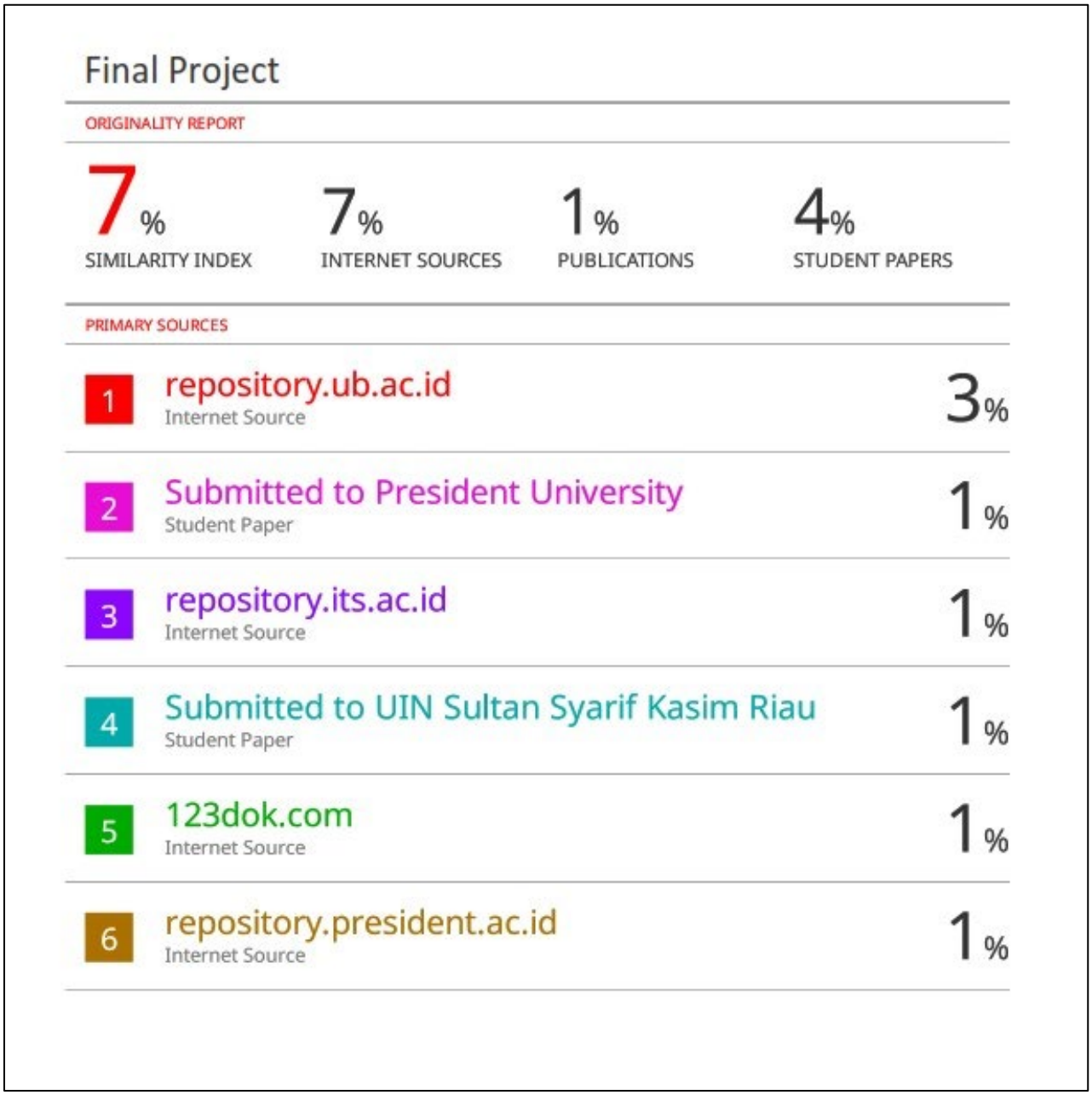
Will be published in journal / institution's repository / proceeding / unpublish.

Cikarang, 26 September 2023



Ronny Juwono, S.Pd., M. T

SIMILIARITY INDEX REPORT



Final Project

ORIGINALITY REPORT

7%	7%	1%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	repository.ub.ac.id Internet Source	3%
2	Submitted to President University Student Paper	1%
3	repository.its.ac.id Internet Source	1%
4	Submitted to UIN Sultan Syarif Kasim Riau Student Paper	1%
5	123dok.com Internet Source	1%
6	repository.president.ac.id Internet Source	1%

ABSTRAK

Keselamatan kerja adalah sarana utama untuk pencegahan kecelakaan, cacat, kematian sebagai akibat kecelakaan kerja. Hampir tidak ada perusahaan yang bebas dari potensi bahaya ataupun kecelakaan kerja. Kecelakaan kerja merupakan suatu kejadian yang tidak direncanakan dan tidak terkontrol atau terkendali yang disebabkan oleh faktor manusia, situasi lingkungan, mesin atau gabungan dari ketiganya yang terjadi pada saat proses kerja yang memungkinkan menghasilkan luka, kesakitan, kematian, dan kerusakan properti atau kejadian yang tidak diinginkan.

Salah satu upaya yang dapat dilakukan untuk mengatasi hal tersebut adalah dengan membuat catatan inspeksi untuk setiap potensi kecelakaan kerja yang dapat terjadi. Dengan mencatat data berupa foto, area atau tempat dimana inspeksi dilaporkan, dan jenis potensi bahaya karyawan dapat lebih mudah untuk menganalisa setiap potensi bahaya penyebab kecelakaan kerja di lingkungan perusahaan. Namun pada umumnya proses pendataan inspeksi masih dilakukan pada media kertas, atau menggunakan program office sederhana seperti MS Word atau MS Excel. Akan tetapi hal tersebut tidak efisien, Kesulitan dalam merangkai data sesuai dengan format yang telah ditentukan akan menjadi faktor yang membuat pengguna menjadi malas dalam melakukan pendataan inspeksi tersebut.

Penelitian ini ditujukan untuk mengembangkan aplikasi safety management system berbasis Web menggunakan bahasa pemrograman php dan framework Laravel. Aplikasi ini memiliki fitur yang dapat membantu pengguna dalam mengelola data inspeksi yang dapat digunakan dimana saja dan kapan saja. Pengembangan aplikasi ini menggunakan metode RAD.

Metode ini dipilih karena membutuhkan pendekatan secara langsung dengan pengguna. Dari hasil pengujian yang telah dilakukan, aplikasi ini dapat membantu pengguna untuk mengelola data inspeksi yang dilakukan sehari-hari, menganalisa data dari hasil inspeksi, dan mengawasi kemajuan dari hasil inspeksi yang sedang dievaluasi. Semua data akan dirangkum dan ditampilkan berupa laporan atau dalam bentuk grafik.

DEDIKASI

Saya dedikasikan laporan ini untuk kedua orang tua saya yang tidak pernah berhenti untuk mendukung saya baik secara moril maupun secara materi. Terima kasih sebesar-besarnya atas pengorbanan kalian selama ini.

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa yang telah melimpahkan rahmat dan hidayah Nya sehingga laporan tugas akhir yang berjudul "Pengembangan aplikasi berbasis web safety management system untuk meningkatkan kesadaran karyawan terhadap safety" ini dapat terselesaikan. Pertama-tama, saya juga ingin mengambil kesempatan ini untuk berterima kasih dan menyampaikan terima kasih yang tulus kepada:

1. **Bapak Ronny Juwono, S.Pd., M.T** selaku kepala program studi IS dan pembimbing yang telah memberikan nasihat, dukungan, dan bimbingan dari awal pengembangan hingga akhir tugas akhir ini. Saya mendapat banyak manfaat dari bimbingannya sehingga saya dapat menyelesaikan tugas akhir ini.
2. **Dekan** dan **para dosen Fakultas Ilmu Komputer** yang telah mengajar dan memberi saya ilmu selama saya berkuliah di President University.
3. **Bapak Jogi Jumogi Sianipar,** dan **Ibu Riama Rosiana Siregar** selaku orangtua penulis, tiada ucapan dan perbuatan yang pantas untuk membalas semua kasih sayang, materi, dukungan, dan doa yang telah diberikan kepada saya.
4. Adik saya, **Excel Brave Joy Okgiria Sianipar** dan **Juwita Maharani Novgiria Sianipar** yang telah mendukung saya untuk menyelesaikan tugas akhir ini.
5. **Bagastama Putra, Dian Prasetyo, Edi Jumanto, Ilham Prasetyo, Ricky Ega Kusuma, Rizky Fajar Maulian** dan **Zaenurohman** yang telah menjadi kawan seperjuangan yang selalu saling mendukung dari awal mulai perkuliahan hingga saat ini.
6. **Apriyanti Sihotang** yang bersedia menjadi teman berdiskusi dan memberikan saran dan masukan dalam proses pengerjaan tugas akhir.

7. Teman-teman seperjuangan lainnya di **IS Evening Class 2017** yang telah menjadi tempat untuk mengembangkan bakat penulis dan tempat bertukar pikiran.
8. Seluruh pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat langsung maupun tidak langsung dalam proses pengerjaan tugas akhir ini.

Penulis menyadari bahwa dalam penyusunan tugas akhir ini masih jauh dari kata sempurna, sehingga saran dan kritik yang membangun sangat penulis harapkan. Akhir kata penulis berharap tugas akhir ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

DAFTAR ISI

Hal

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Hampir tidak ada perusahaan yang bebas dari potensi bahaya ataupun kecelakaan kerja. Saat ini banyak individu yang belum dapat menerapkan keselamatan kerja pada lingkungan perusahaan. Menerapkan keselamatan kerja perlu dilakukan untuk meningkatkan keamanan serta menghindari adanya kecelakaan kerja yang dapat merugikan baik pihak karyawan maupun perusahaan. Keselamatan kerja merupakan sarana utama untuk pencegahan kecelakaan, cacat, kematian sebagai akibat kecelakaan kerja [1].

Salah satu upaya yang dapat dilakukan untuk mengatasi hal tersebut adalah dengan membuat catatan inspeksi untuk setiap potensi kecelakaan kerja yang dapat terjadi. Dengan mencatat data berupa foto, area atau tempat dimana inspeksi dilaporkan, dan jenis potensi bahaya karyawan dapat lebih mudah untuk menganalisa setiap potensi bahaya penyebab kecelakaan kerja di lingkungan perusahaan. Namun pada umumnya proses pendataan inspeksi masih dilakukan pada media kertas, atau menggunakan program office sederhana seperti MS Word atau MS Excel. Akan tetapi hal tersebut tidak efisien, Kesulitan dalam merangkai data sesuai dengan format yang telah ditentukan akan menjadi faktor yang membuat pengguna menjadi malas dalam melakukan pendataan inspeksi tersebut [2].

Untuk mengatasi masalah tersebut, dibutuhkan suatu sistem yang dapat membantu dan memudahkan dalam melakukan pendataan inspeksi terhadap potensi penyebab

kecelakaan kerja di perusahaan [3]. Dengan sistem ini, pengguna dapat dengan mudah melakukan pendataan inspeksi berupa gambar, area tempat inspeksi dilaporkan, dan jenis potensi bahaya dari inspeksi tersebut menggunakan berbagai jenis perangkat elektronik seperti *smartphone*, *laptop* atau *PC (personal computer)*. Sistem juga memiliki beberapa level user untuk menentukan otoritas dalam penggunaan sistem, salah satunya adalah user safety team yang bertugas untuk menindaklanjuti laporan sebelum diteruskan kepada departemen terkait. Hal ini diperlukan untuk mencegah user membuat inspeksi lalu menunjuk departemen yang bertanggung jawab secara sembarangan dan mencegah adanya kesalahan data inspeksi.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah dipaparkan diatas, maka rumusan masalah dapat ditentukan sebagai berikut:

1. Bagaimana cara merancang sebuah aplikasi safety management system berbasis web.
2. Bagaimana cara mengelola keamanan lingkungan kerja di perusahaan.
3. Bagaimana cara mencatat setiap potensi yang dapat menyebabkan terjadinya kecelakaan kerja.
4. Bagaimana cara mengawasi setiap evaluasi terhadap potensi bahaya yang dilaporkan.
5. Bagaimana cara menganalisa statistik data inspeksi yang telah berjalan.

1.3 Tujuan Penelitian

Tujuan utama penelitian ini dilakukan secara spesifik untuk menyelesaikan permasalahan sebagai berikut:

1. Untuk mengembangkan sebuah aplikasi pengelola HSE berbasis web.
2. Untuk mencatat setiap potensi yang dapat menyebabkan terjadinya kecelakaan kerja.
3. Untuk menganalisa, merencanakan dan mengendalikan HSE dalam perusahaan.

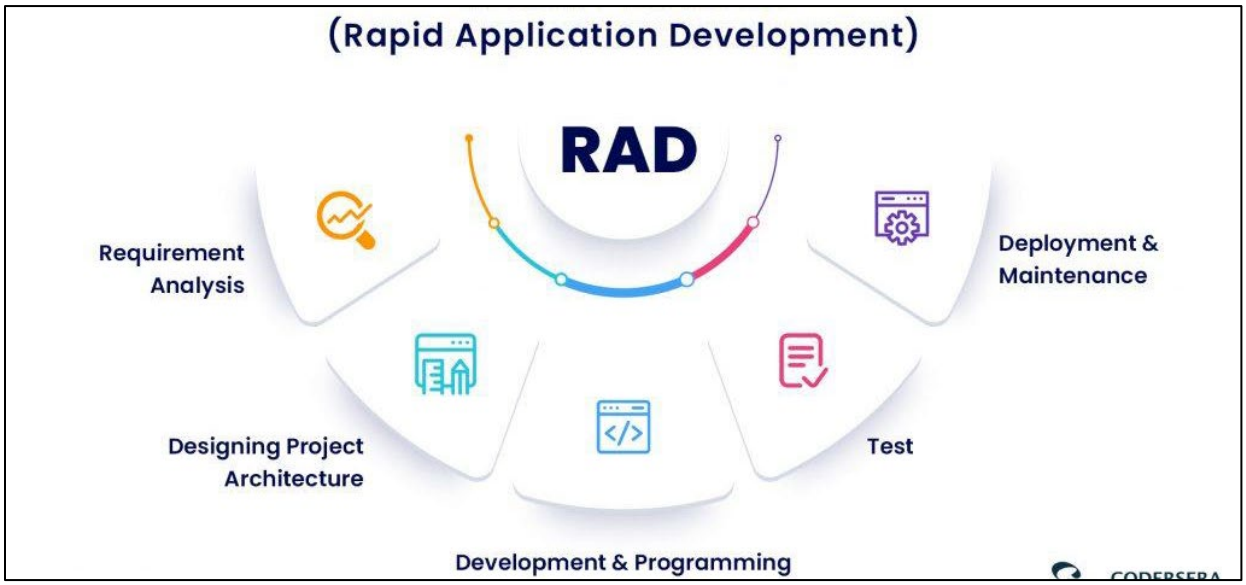
1.4 Batasan Masalah

Penelitian ini berfokus pada pengembang aplikasi yang dapat membantu penggunanya untuk mengelola, menganalisa, merencanakan, dan mengendalikan lingkungan kerja. Aplikasi dapat diakses dari berbagai jenis perangkat. Berdasarkan latar belakang yang telah dijelaskan diatas, terdapat batasan dalam penelitian ini, diantaranya:

1. Pengguna harus memiliki koneksi internet untuk mengakses aplikasi ini.
2. Perangkat harus mempunyai browser dengan status sudah terupdate ke versi terbaru agar dapat menjalankan aplikasi ini dengan baik.
3. Pengguna harus memiliki akun untuk dapat masuk ke dalam aplikasi ini.

1.5 Metodologi Penelitian

Metodologi yang dipilih penulis untuk mengembangkan aplikasi ini adalah metode Rapid Application Development (RAD). Metodologi pengembangan ini memprioritaskan pembuatan prototype yang cepat dan menerima umpan balik yang cepat dari pengguna, sehingga perubahan kebutuhan sistem dapat diimplementasikan ke dalam sistem dengan cepat dibandignkan siklus pengembangan dan pengujian aplikasi secara tradisional yang terlalu terikat dengan jadwal pengembangan yang telah ditentukan. Dalam metologi ini persyaratan sistem dibagi menjadi komponen-komponen kecil, sehingga dapat melakukan perubahan selama proses pengembangan [4]. Dengan begitu, pengembang dapat membuat beberapa iterasi dan pembaruan perangkat lunak dengan cepat tanpa perlu memulai jadwal pengembangan dari awal. Metodologi RAD memiliki 4 tahap, seperti yang di tunjukkan pada Gambar 1.1.



1. Fase 1: Requirement Planning

Pada tahap awal metode pengembangan ini, tim pengembang, pengguna perangkat lunak, dan anggota tim lainnya berkomunikasi untuk menunjukan tujuan proyek pengembangan, mendefinisikan kebutuhan sistem perangkat lunak, serta mengidentifikasi masalah yang ada saat ini dan potensi masalah yang nantinya akan dihadapi selama pengembangan perangkat lunak.

2. Fase 2: User Design Phase

   Fase ini adalah inti dari metodologi RAD dan di tahap inilah yang membedakannya RAD dari metodelogi lainnya. Selama dalam tahap ini, pengunna bekerja bahu membahu dengan pengembang untuk memastikan kebutuhan mereka terpenuhi di setiap langkah dalam proses desain perangkat lunak, di mana pengguna dapat menguji setiap prototipe produk pada setiap tahap untuk memastikan aplikasi memenuhi harapan mereka.

3. Fase 3: Construction Phase

   Di tahap ini tim pengembang, programmer, dan penguji bekerja sama untuk memastikan semuanya bekerja dengan lancar dan hasil akhirnya memenuhi persyaratan kebutuhan perangkat lunak yang dirancang. Tahap ini mencakup pemrogramman aplikasi dan pengujian aplikasi, mulai dari unit test, integration test, dan system testing.

4. Fase 4: Cutover Phase

   Di tahap akhir metodelogi RAD ini, perangkat lunak akan diuji mulai dari fungsionalitasnya dan performanya menggunakan use case test untuk memastikan perangkat lunak yang dikembangkan bekerja sesuai dengan yang

diharapkan. Ditahap ini juga para pengembang masih terus mencari bugs di aplikasi yang telah dikembangkan.

1.6 Sistematika Pembahasan

Penelitian ini terdiri dari 7 BAB, yang diantaranya:

1. BAB I: PENDAHULUAN

   Bab pertama ini mengenalkan tujuan dari penelitian ini, yang berisikan latar belakang masalah, rumusan masalah, tujuan penelitian, batasan masalah, metodologi penelitian dan sistematika pembahasan penelitian ini.

2. BAB II: STUDI PUSTAKA

   Studi pustaka mendeskripsikan kajian kepustakaan yang berkaitan dengan pengembangan aplikasi *Safety Management System*, yang diguanakan sebagai referensi dalam melakukan penelitian.

3. BAB III: ANALISIS SISTEM

   Bab ini memuat pengalian fungsionalistas, proses bisnis dan desain arsitektur sistem berdasarkan dari kebutuhan sistem secara mendetail.

4. BAB IV: DESAIN SISTEM

   Bab ini memberi penjelasan tentang desain dari aplikasi yang dikembangkan, basis data sistem, arsitektur sistem, UI/UX dan komponen-komponen sistem sesuai dengan kebutuhan sistem.

5. BAB V: IMPLEMENTASI SISITEM

   Pembahasan yang dicakup bab ini adalah pengembangan antarmuka proses penerapan dan penjelasan dari algoritma yang digunakan pada sistem ini.

6. BAB VI: PENGUJIAN DAN EVALUASI SISTEM

   Bab ini memuat proses pengujian dari aplikasi *Safety Management System* yang kemudian dilakukan analisa terhadap hasil yang didapatkan.

7. BAB VII: KESIMPULAN DAN SARAN

   Bab ini mencakup kesimpulan yang diperoleh dari aplikasi yang dikembangkan, hasil dan kemungkinan karya masa depan yang dapat dikembangakan.

BAB II

STUDI KEPUSTAKAAN

2.1 HSE (Health Safety Environment)

HSE adalah singkatan dari health, safety, and environment yang merupakan serangkaian proses dan prosedur yang mengidentifikasi potensi bahaya pada lingkungan kerja tertentu. Pengembangan praktik HSE dilakukan untuk mengurangi dan/atau menghilangkan bahaya serta melatih karyawan untuk pencegahan kecelakaan atau respons terhadap sesuatu yang mengancam [6].

2.2 Sistem Informasi

Sistem informasi adalah seperangkat komponen yang saling berhubungan yang digunakan untuk mengumpulkan, menyimpan, memproses, dan mengirimkan data dan informasi digital. Pada intinya, ini adalah kumpulan perangkat keras, perangkat lunak, data, manusia, dan proses yang bekerja sama untuk mengubah data mentah menjadi informasi berguna. IS mendukung berbagai tujuan bisnis seperti peningkatan layanan pelanggan atau peningkatan efisiensi [7].

2.3 PHP

PHP (singkatan rekursif untuk PHP: Hypertext Preprocessor) adalah bahasa skrip tujuan umum open source yang banyak digunakan dan sangat cocok untuk pengembangan web dan dapat disematkan ke dalam HTML [8].

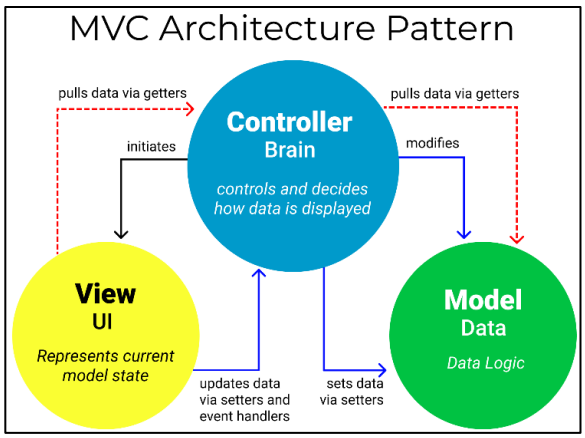
2.4 Laravel

Laravel adalah kerangka aplikasi web dengan sintaksis yang ekspresif dan elegan. Kami telah meletakkan fondasinya — memberi Anda kebebasan untuk berkreasi tanpa harus memikirkan hal-hal kecil [9].

2.5 Model View Controller

Model–view–controller (MVC) adalah pola desain perangkat lunak yang biasa digunakan untuk mengembangkan antarmuka pengguna yang membagi logika program terkait menjadi tiga elemen yang saling berhubungan [10], diantaranya:

1. *Model* merupakan *backend* yang berisi semua logika data. Bertugas untuk mengelola data. Baik data berasal dari database, API, atau objek JSON, dan lain sebagainya.
2. *View* sebagai *frontend* atau antarmuka pengguna grafis *(GUI)*. Bertugas untuk menampilkan apa yang akan dilihat pengguna pada layar.
3. *Controller* yaitu otak aplikasi yang mengontrol bagaimana data ditampilkan. Bertugas untuk menarik, memodifikasi, dan menyediakan data kepada pengguna. Controller adalah penghubung antara *View* dan *Model*.



Pola Arsitektur MVC mengubah pengembangan aplikasi yang kompleks menjadi proses yang lebih mudah dikelola. Hal ini memungkinkan beberapa pengembang untuk mengerjakan aplikasi secara bersamaan.

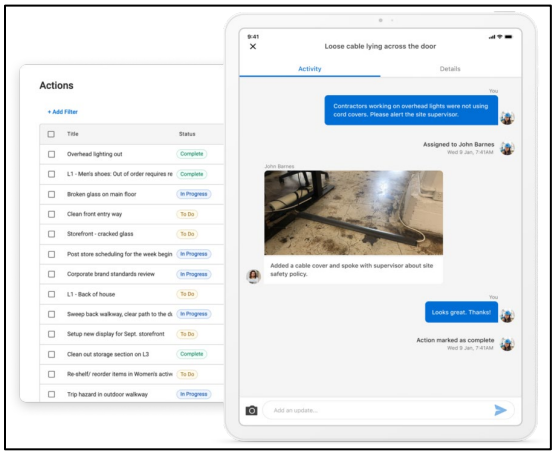
Pola MVC membantu penulis memecah kode frontend dan backend menjadi komponen terpisah. Dengan cara ini, akan lebih mudah untuk mengelola dan membuat perubahan pada kedua sisi tanpa mengganggu satu sama lain [14].

2.6 Aplikasi serupa

Bagian ini membahas aplikasi yang ada yang terkait dengan atau mirip dengan *Safety Management System* dalam beberapa fungsinya, tetapi berbeda dalam hal fitur aplikasi dan bagaimana fungsionalitas diimplementasikan.

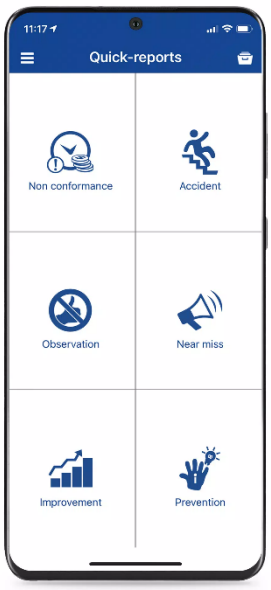
2.6.1 Safety Culture

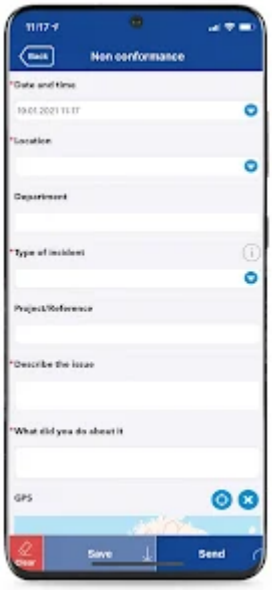
Safety Culture memiliki fitur lengkap dalam hal inspeksi dan audit terhadap keselamatan kerja pada lingkungan kerja. Namun aplikasi ini tidak mudah untuk digunakan oleh pengguna yang awam dikarenakan terlalu banyak menu dan langkah-langkah yang perlu dipelajari oleh user.



2.6.2 HSEQ+

HSEQ+ memiliki beberapa fitur yang sama, seperti pencatatan data inspeksi. Aplikasi ini juga memiliki tampilan yang simple. Namun aplikasi ini hanya dapat diakses melalui perangkat android saja.





G

2.6.3 Perbandingan Aplikasi Serupa

Tabel 2.1 Perbandingan Aplikasi Serupa

No	Fitur	Safety Culture	HSEQ+	Safety Management System
1	*Cross-platform* *Transactions*	✓	✗	✓
2	*Insepction with image*	✓	✓	✓
3	*Random inspections program*	✗	✗	✓
4	*Interactive dashboard*	✗	✗	✓

BAB III

ANALISIS SISTEM

3.1. Gambaran Umum Sistem

Safety Management System dikembangkan untuk membantu menganalisa, merencanakan, dan mengendalikan proses inspeksi terhadap potensi bahaya kecelakaan kerja di perusahaan secara mandiri melalui aplikasi berbasis web yang dapat diakses dengan berbagai perangkat elektronik seperti, smartphone, tablet, laptop, pc, dan perangkat lainnya yang mendukung aplikasi browser di perangkat tersebut. Fitur utama dari aplikasi ini adalah untuk mencatat seluruh proses laporan inspeksi yang telah dimasukkan ke dalam sistem sehingga proses evaluasi terhadap inspeksi tersebut dapat di tracking dengan baik. Berdasarkan data-data inspeksi yang telah dicatat, sistem akan membuat rangkuman data yang ditampilkan dalam bentuk grafik interaktif yang dapat diakses langsung pada grafik tersebut untuk menampilkan setiap data secara detail. Fitur tambahan lainnya adalah pengguna dapat melihat rangkuman data tersebut dalam periode waktu tertentu.

3.2. Kebutuhan Fungsional Sistem

Kebutuhan fungsional mendefinisikan apa saja yang seharusnya dapat dilakukan oleh sistem. Dalam penelitian ini, kebutuhan fungsional diklarifikasi berdasarkan studi kepustakaan terhadap HSE (Health Safety Environment) yang telah dikaji di bab sebelumnya. Setiap kebutuhan fungsional akan diberikan kode SRS_SMS_F_X. SRS merupakan singkatan dari System Requirement Spesification, SMS merupakan singkatan

dari Safety Management System, F merupakan kebutuhan fungsional dan X merupakan nomor dari definisi kebutuhan. Kebutuhan fungsional pada Safety Management System dapat dilihat pada tabel 3.1.

Tabel 3.1 Daftar Kebutuhan Fungsional

No	**Kode Fungsi**	**Deskripsi Kebutuhan**
1.	SRS_SMS_F_01	Pengguna dapat melakukan login ke dalam sistem.
2.	SRS_SMS_F_02	Pengguna dapat melihat statistik data inspeksi
3.	SRS_SMS_F_03	Pengguna dapat mengupload hasil inspeksi berupa gambar dan detail informasi ke dalam sistem *(live inspections).*
4.	SRS_SMS_F_04	Pengguna dapat mengupload hasil inspeksi kepada departemen sesuai dengan pemilihan acak dari sistem berupa gambar dan detail informasi ke dalam sistem *(random inspections)*.
5.	SRS_SMS_F_05	Pengguna dapat mengubah rincian inspeksi yang tercatat secara individual.
6.	SRS_SMS_F_06	Pengguna dapat melakukan approval terhadap inspeksi yang telah masuk ke dalam sistem.
7.	SRS_SMS_F_07	Pengguna dapat melaporkan proses evaluasi terhadap inspeksi ke dalam sistem.

8.	SRS_SMS_F_08	Pengguna dapat melaporkan bahwa proses evaluasi telah selesai dikerjakan.
9.	SRS_SMS_F_09	Pengguna dapat melakukan closing untuk inspeksi yang tercatat secara individual.
10.	SRS_SMS_F_10	Pengguna dapat menambahkan data user ke dalam sistem.
11.	SRS_SMS_F_11	Pengguna dapat mengubah detail informasi user secara individual.
12.	SRS_SMS_F_12	Pengguna dapat menonaktifkan user secara individual.
13.	SRS_SMS_F_13	Pengguna dapat menghapus user secara individual.
14.	SRS_SMS_F_14	Pengguna dapat menambahkan data feedback ke dalam sistem
15.	SRS_SMS_F_15	Pengguna dapat melihat rincian data informasi feedback secara individual.
16.	SRS_SMS_F_16	Pengguna dapat mentrigger sistem agar melakukan pengacakan target inspeksi kepada user.
17.	SRS_SMS_F_17	Pengguna dapat menerjemahkan halaman menggunakan *Google Translate API*.

3.3 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional mendefinisikan apa saja kebutuhan yang tidak berhubungan langsung dengan pengguna, namun memberikan kualitas terhadap sistem dan kepuasan terhadap pengguna. Kebutuhan non-fungsional pada Safety Management System dapat dilihat pada Tabel 3.2.

Tabel 3.2 Daftar Kebutuhan Non-Fungsional

No	**Parameter**	**Deskripsi Kebutuhan**
1	Usabilitas	Rancangan antarmuka mudah digunakan pengguna berdasarkan *effectiveness, efficiency,* dan *satisfaction* [11].
2	Kompatibilitas	Perangkat harus mempunyai browser *(Google Chrome, Mozilla Firefox, Microsoft Edge, dsb).*

3.4 Kebutuhan Perangkat Keras dan Perangkat Lunak

Untuk mencapai tujuan dari penelitian ini, berikut beberapa teknologi berupa perangkat keras dan pendukung yang diperlukan dalam mengembangkan aplikasi Safety Management System ini:

1. Sistem Operasi

   Sistem ini dikembangkan menggunakan *Personal Computer* yang menjalankan sistem operasi Windows® 10 64-Bit Pro versi 22H2 (OS Build 19045.3086).

2. Lingkup pengembangan

   Visual Studio Code adalah sebuah Integrated Development System (IDE) yang didesain untuk mengembangkan berbagai aplikasi khususnya berbasis aplikasi berbasis web.

3. PHP

   Hypertext Preprocessor atau PHP adalah bahasa penulisan skrip *open-source* yang banyak digunakan dalam pemrograman atau pengembangan website (web development). Bahasa ini umumnya dijalankan dalam komunikasi sisi server, dan saat ini didukung oleh hampir semua sistem. Pengembangan ini menggunakan PHP versi 8.1.17

4. Composer

   Composer adalah alat manajemen ketergantungan open source untuk PHP, dibuat terutama untuk memfasilitasi distribusi dan pemeliharaan paket PHP

sebagai komponen aplikasi individual. Ini telah mengubah ekosistem PHP secara dramatis dengan menciptakan dasar bagi pengembangan PHP modern, dengan aplikasi berbasi komponen dan *framework*.

5. Laravel

Laravel adalah framework berbasis bahasa pemrograman PHP yang bisa digunakan untuk membantu proses pengembangan sebuah website agar lebih maksimal. Dengan menggunakan Laravel, website yang dihasilkan akan lebih dinamis. Pengembangan ini menggunakan Laravel versi 10.14.1.

6. Valet

Valet adalah lingkungan pengembangan yang sangat cepat untuk Laravel. Salah satu alasannya begitu terkenal adalah karena berjalan tanpa perangkat lunak server web (Apache dan Nginx). Ini menggunakan DnsMasq di Mac OS, dan DNS Akrilik di Windows 10, Valet memproksi semua permintaan di domain *.test untuk menunjuk ke situs yang diinstal di mesin lokal Anda. Teknologi ini memungkinkan sistem dapat diakses dari manapun seperti sistem yang sudah dihosting.

7. Livewire

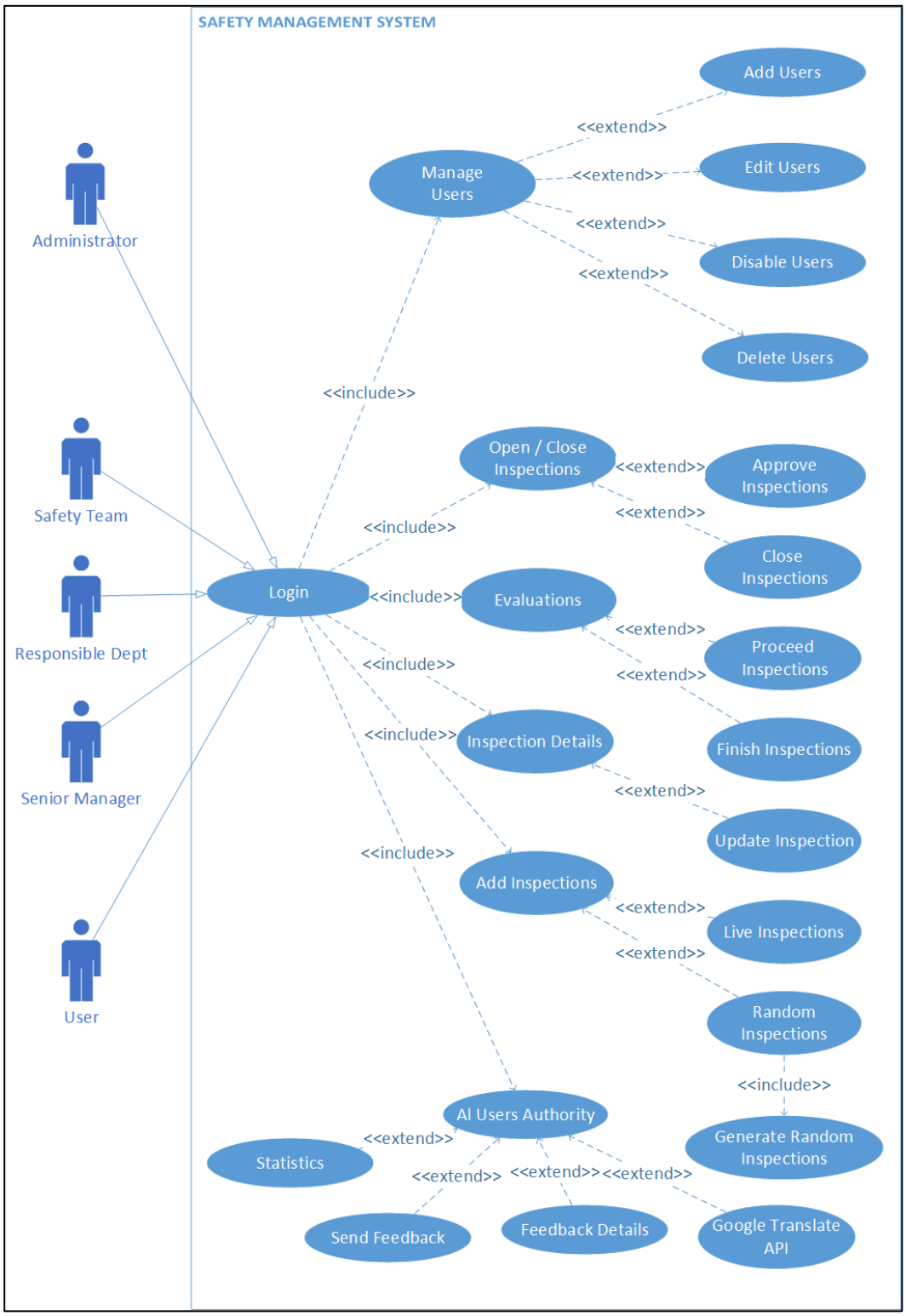
Livewire adalah *full-stack framework* untuk Laravel yang membuat pembuatan antarmuka dinamis menjadi sederhana, tanpa mengubah cara kerja ketika menggunakan *framework* Laravel.

8. XAMPP

XAMPP adalah *cross-platform web server* yang dikembangkan oleh Apache Friends. XAMPP terdiri dari Apache HTTP Server MySQL database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP merupakan singkatan dari X (tempat sistem operasi apapun), Apache, MySQL, PHP dan Perl. Program ini tersedia dalam GNU General Public License dan bebas, merupakan web server yang mudah digunakan yang dapat melayani tampilan halaman web yang dinamis.

3.5 Use Case Diagram

Use Case Diagram merupakan sebuah diagram yang digunakan menggambarkan serangkaian tindakan (use case) yang harus atau dapat dilakukan sistem (subject) dan memvisualisasikan interaksi dari berbagai hal tersebut dengan pengguna (actor). Use Case Diagram mewakili bagian inti dari suatu sistem dan alur kerja di antara mereka. Dengan bantuan use case diagram, kita dapat mengetahui kondisi sebelum dan sesudah interaksi dengan pengguna (actor). Use Case Diagram untuk aplikasi *System Management System* dapat dilihat pada Gambar 3.1.



3.6 Use Case Narrative

Use Case Narrative menggambarkan urutan kejadian yang terjadi setelah setiap tindakan atau use case, dan bagaimana aktor atau pengguna akan berinteraksi dengan sistem untuk menyelesaikan tugasnya dalam deskripsi tekstual.

Tabel 3.3 *Use Case Login*

Use Case Name	*Login*	
Priority	*High*	
Primary System Actor	All User	
Description	*Actor* melakukan login ke dalam sistem.	
Precondition	Tidak ada	
Trigger	*Actor* ingin masuk kedalam sistem.	
Typical Course of Event	*Actor Action*	*System Response*
	1. *Actor* memasukkan *username* dan *password* yang sudah terdaftar dalam sistem. 3. *Actor* menekan tombol "*Login*".	2. Sstem melakukan validasi terhadap setiap variabel yang diinput. 4. Sistem melakukan validasi terhadap *username* dan *password* yang ada di database. Jika data valid user akan diarahkan ke halaman *Dashboard*. Jika

		tidak valid, maka akan diarahkan kembali ke halaman *Login*.
Post Condition	Jika use case berhasil, sistem akan mengalihkan tampilan ke halaman *Dashboard*.	

Tabel 3.4 *Use Case Statistics*

Use Case Name	Inspection Details	
Priority	*High*	
Primary System Actor	All User	
Description	*Actor* melihat statistik inspeksi yang tercatat.	
Precondition	Tidak ada	
Trigger	*Actor* ingin melihat statistik inspeksi.	
Typical Course of Event	*Actor Action*	*System Response*
	1. Actor masuk ke halaman *Dashboard*.	2. Sistem mengambil data-data inspeksi kemudian menampilkan statistik data inspeksi.
Post Condition	Sistem menampilkan rincian data inspection.	

Tabel 3.5 *Use Case Live Inspections*

Use Case Name	*Live Inspections*	
Priority	*High*	
Primary System Actor	All User	
Description	*Actor* melaporkan inspeksi.	
Precondition	Tidak ada	
Trigger	*Actor* ingin melaporkan inspeksi.	
Typical Course of Event	*Actor Action*	*System Response*
	1. *Actor* masuk ke menu *Live Inspections*. 3. *Actor* melakukan input data. 5. Actor menekan tombol "*Submit*".	2. Sistem menampilkan form *Live Inspections*. 4. Sistem melakukan validasi terhadap setiap variabel yang diinput. 6. Sistem menyimpan inspeksi kedalam sistem. 7. Sistem mengalihkan tampilan ke halaman *My Inspections.*
Post Condition	Data inspeksi ditambahkan ke database.	

	Status inspeksi menjadi “Pending”. Sistem mengalihkan tampilan ke halaman *My Inspections.*

Tabel 3.6 *Use Case Random Inspections*

Use Case Name	*Random Inspections*	
Priority	*High*	
Primary System Actor	All User	
Description	*Actor* melaporkan inspeksi.	
Precondition	Tidak ada	
Trigger	*Actor* ingin melaporkan inspeksi.	
Typical Course of Event	*Actor Action*	*System Response*
	1. *Actor* masuk ke menu *Random Inspections.* 3. *Actor* melakukan input data. 5. Actor menekan tombol “*Submit*”.	2. Sistem menampilkan form *Random Inspections*. 4. Sistem melakukan validasi terhadap setiap variabel yang diinput. 6. Sistem menyimpan inspeksi kedalam sistem. 7. Sistem mengalihkan

		tampilan ke halaman *My Inspections.*
Post Condition	Data inspeksi ditambahkan ke database. Status inspeksi menjadi "Pending". Sistem mengalihkan tampilan ke halaman *My Inspections.*	

Tabel 3.7 *Use Case Update Inspection*

Use Case Name	*Update Inspection*	
Priority	*High*	
Primary System Actor	Safety Team	
Description	*Actor* melakukan perubahan terhadap data inspeksi.	
Precondition	Tidak ada	
Trigger	*Actor* ingin melakukan perubahan terhadap data inspeksi.	
Typical Course of Event	*Actor Action*	*System Response*
	1. Actor masuk ke menu *My Inspections*. 3. *Actor* memilih data	2. Sistem menampilkan data inspections terbaru. 4. Sistem menampilkan

	inspections yang ingin diproses. 5. *Actor* melakukan input data untuk mengubah rincian inspeksi. 7. *Actor* menekan tombol "Submit".	data inspection details. 6. Sistem melakukan validasi terhadap setiap variabel yang diinput. 8. Sistem menyimpan data inspection ke dalam sistem. 9. Sistem mengalihkan tampilan ke halaman *My Inspections.*
Post Condition	Data disimpan ke database.	

Tabel 3.8 *Use Case Approve Inspections*

Use Case Name	*Approve Inspections*
Priority	*High*
Primary System Actor	Safety Team
Description	*Actor* melakukan approval dan review terhadap data inspeksi.
Precondition	Tidak ada
Trigger	*Actor* ingin melakukan approval dan review terhadap

	data inspeksi.	
Typical Course of Event	*Actor Action*	*System Response*
	1. Actor masuk ke menu *Notifications*. 3. *Actor* memilih data inspections yang ingin diproses. 5. *Actor* melakukan input data lanjutan dan mengubah rincian inspeksi (jika diperlukan). 7. *Actor* menekan tombol "Submit".	2. Sistem menampilkan data inspections terbaru. 4. Sistem menampilkan data inspection details. 6. Sistem melakukan validasi terhadap setiap variabel yang diinput. 8. Sistem menyimpan data inspection ke dalam sistem. 9. Sistem mengalihkan tampilan ke halaman *My Inspections.*
Post Condition	Data disimpan ke database. Status inspeksi berganti menjadi "Open".	

Tabel 3.9 *Use Case Proceed Inspections*

Use Case Name	*Proceed Inspections*

Priority	*High*	
Primary System Actor	Responsible Department	
Description	*Actor* melaporkan proses evaluasi terhadap data inspeksi.	
Precondition	Tidak ada	
Trigger	*Actor* ingin melaporkan proses evaluasi terhadap data inspeksi.	
Typical Course of Event	*Actor Action*	*System Response*
	1. *Actor* masuk ke menu *Notifications*. 3. *Actor* memilih data inspections yang ingin diproses. 5. *Actor* melakukan input data lanjutan. 7. *Actor* menekan tombol "Submit".	2. Sistem menampilkan data inspections terbaru. 4. Sistem menampilkan data inspection details. 6. Sistem melakukan validasi terhadap setiap variabel yang diinput. 8. Sistem menyimpan data inspection ke dalam sistem. 9. Sistem mengalihkan

		tampilan ke halaman *My Inspections.*
Post Condition	Data disimpan ke database. Status inspeksi berganti menjadi “In Progress”.	

Tabel 3.10 *Use Case Finish Inspections*

Use Case Name	*Finish Inspections*	
Priority	*High*	
Primary System Actor	Responsible Department	
Description	*Actor* melaporkan proses evaluasi terhadap data inspeksi.	
Precondition	Tidak ada	
Trigger	*Actor* ingin melaporkan proses evaluasi terhadap data inspeksi.	
Typical Course of Event	*Actor Action*	*System Response*
	1. *Actor* masuk ke menu *Notifications*. 3. *Actor* memilih data inspections yang ingin diproses.	2. Sistem menampilkan data inspections terbaru. 4. Sistem menampilkan data inspection details.

	5. *Actor* melakukan input data lanjutan. 7. *Actor* menekan tombol "Submit".	6. Sistem melakukan validasi terhadap setiap variabel yang diinput. 8. Sistem menyimpan data inspection ke dalam sistem. 9. Sistem mengalihkan tampilan ke halaman *My Inspections.*
Post Condition	Data disimpan ke database. Status inspeksi berganti menjadi "Finished".	

Tabel 3.11 *Use Case Close Inspections*

Use Case Name	*Close Inspections*
Priority	*High*
Primary System Actor	Safety Team
Description	*Actor* melakukan closing terhadap data inspeksi.
Precondition	Tidak ada

Trigger	*Actor* ingin melakukan closing terhadap data inspeksi.	
Typical Course of Event	*Actor Action*	*System Response*
	1. *Actor* masuk ke menu *Notifications*. 3. *Actor* memilih data inspections yang ingin diproses. 4. *Actor* menekan tombol "Close Inspections".	2. Sistem menampilkan data inspections terbaru. 4. Sistem menampilkan data inspection details. 5. Sistem menyimpan data inspection ke dalam sistem. 6. Sistem mengalihkan tampilan ke halaman *My Inspections.*
Post Condition	Data disimpan ke database. Status inspeksi berganti menjadi "Closed".	

Tabel 3.12 *Use Case Add User*

Use Case Name	*Add User*
Priority	*High*
Primary System Actor	Administrator

Description	*Actor* menambahkan data user.	
Precondition	Tidak ada	
Trigger	*Actor* ingin melakukan closing terhadap data inspeksi.	
Typical Course of Event	*Actor Action*	*System Response*
	1. *Actor* masuk ke menu *User Registration*. 3. *Actor* melakukan input data 5. *Actor* menekan tombol "Submit".	2. Sistem menampilkan form *User Registration*. 4. Sistem melakukan validasi terhadap setiap variabel yang diinput. 6. Sistem menyimpan data inspection ke dalam sistem. 7. Sistem mengalihkan tampilan ke halaman *Dashboard*..
Post Condition	Data user ditambahkan ke database. Sistem mengalihkan tampilan ke halaman *Dashboard.*	

Tabel 3.13 *Use Case Edit User*

Use Case Name	*Edit User*

Priority	*High*	
Primary System Actor	Adminsitrator, Auth User	
Description	*Actor* menambahkan data user.	
Precondition	Tidak ada	
Trigger	*Actor* ingin melakukan closing terhadap data inspeksi.	
Typical Course of Event	*Actor Action*	*System Response*
	1. *Actor* masuk ke menu *Profile*. 3. *Actor* memilih data user yang ingin diedit. 5. *Actor* melakukan input data 7. Actor menekan tombol "Submit".	2. Sistem menampilkan form *Profile*. 4. Sistem menampilkan rincian data user. 6. Sistem melakukan validasi terhadap setiap variabel yang diinput. 8. Sistem menyimpan data inspection ke dalam sistem. 9. Sistem mengalihkan tampilan ke halaman *Dashboard*..
Post Condition	Data user ditambahkan ke database.	

	Sistem mengalihkan tampilan ke halaman *Dashboard.*	

Tabel 3.14 *Use Case Disable User*

Use Case Name	*Disable User*	
Priority	*High*	
Primary System Actor	Adminsitrator	
Description	*Actor* menonaktifkan user.	
Precondition	Tidak ada	
Trigger	*Actor* ingin menonaktifkan user.	
Typical Course of Event	*Actor Action*	*System Response*
	1. *Actor* masuk ke menu *All users*. 3. *Actor* memilih data user yang ingin dinonaktifkan. 5. *Actor* menekan tombol disable user.	2. Sistem menampilkan form *Profile*. 4. Sistem menampilkan rincian data user. 6. Sistem menyimpan data users ke dalam sistem. 7. Sistem mengalihkan tampilan ke halaman

		Dashboard..
Post Condition	Status user menjadi disabled. Sistem mengalihkan tampilan ke halaman *Dashboard.*	

Tabel 3.15 *Use Case Delete User*

Use Case Name	*Delete User*	
Priority	*High*	
Primary System Actor	Adminsitrator	
Description	*Actor* menghapus data user.	
Precondition	Tidak ada	
Trigger	*Actor* ingin menghapus data user.	
Typical Course of Event	*Actor Action*	*System Response*
	1. *Actor* masuk ke menu *All Users*. 3. *Actor* memilih data user yang ingin dihapus. 5. *Actor* menekan tombol	2. Sistem menampilkan form *Profile*. 4. Sistem menampilkan rincian data user. 4. Sistem melakukan

	delete user.	validasi terhadap *users* dan *inspections* yang ada di database. Jika tidak terdapat data inspeksi dengan pic atau created_by user yang akan dihapus, maka sistem akan menghapus data user tersebut, jika sebaliknya maka penghapusan akan dibatalkan dan tampilan dialihkan kembali ke halaman *Profile Details.*
Post Condition	Jika use case berhasil, sistem akan menghapus data user tersebut. Sistem akan mengalihkan tampilan ke halaman Dashboard.	

Tabel 3.16 *Use Case Send Feedback*

Use Case Name	*Send Feedback*

Priority	*High*	
Primary System Actor	All User	
Description	*Actor* mengirim feedback.	
Precondition	Tidak ada	
Trigger	*Actor* ingin mengirim feedback.	
Typical Course of Event	*Actor Action*	*System Response*
	1. *Actor* masuk ke menu *Send Feedback.* 3. *Actor* melakukan input data. 5. Actor menekan tombol "*Submit*".	2. Sistem menampilkan form *Send Feedback*. 4. Sistem melakukan validasi terhadap setiap variabel yang diinput. 6. Sistem menyimpan inspeksi kedalam sistem. 7. Sistem mengalihkan tampilan ke halaman *All Feedback.*
Post Condition	Data feedback ditambahkan ke database. Sistem mengalihkan tampilan ke halaman *All Feedback.*	

Tabel 3.17 *Use Case Feedback Details*

Use Case Name	*Feedback Details*	
Priority	*High*	
Primary System Actor	All User	
Description	*Actor* melihat data inspeksi.	
Precondition	Tidak ada	
Trigger	*Actor* ingin melihat data inspeksi.	
Typical Course of Event	*Actor Action*	*System Response*
	1. Actor masuk ke halaman *All Feedback.*	2. Sistem menampilkan data *Feedback*.
	3. Actor memilih data feedback yang ingin dilihat.	4. Sistem menampilkan rincian data feedback.
Post Condition	Sistem menampilkan rincian data feedback.	

Tabel 3.18 *Use Case Generate Random Inspections*

Use Case Name	*Generate Random Inspections*
Priority	*High*
Primary System Actor	All User

Description	*Actor* melakukan trigger terhadap sistem untuk memilih target departemen kepada user secara acak.	
Precondition	Tidak ada	
Trigger	*Actor* ingin agar sistem melakukan pemilihan target departemen secara acak untuk melakukan Random Inspections.	
Typical Course of Event	*Actor Action*	*System Response*
	1. Actor masuk ke menu *Send Feedback*. 3. Actor menekan tombol *"Generate Random Inspections"*.	2. Sistem menampilkan form *Send Feedback*. 4. Sistem melakukan pemilihan target departemen kepada user secara acak.
Post Condition	Data user akan memiliki data target departemen. Sistem mengalihkan tampilan ke halaman *Dashboard.*	

Tabel 3.19 *Use Case Google Translate API*

Use Case Name	*Google Translate API*

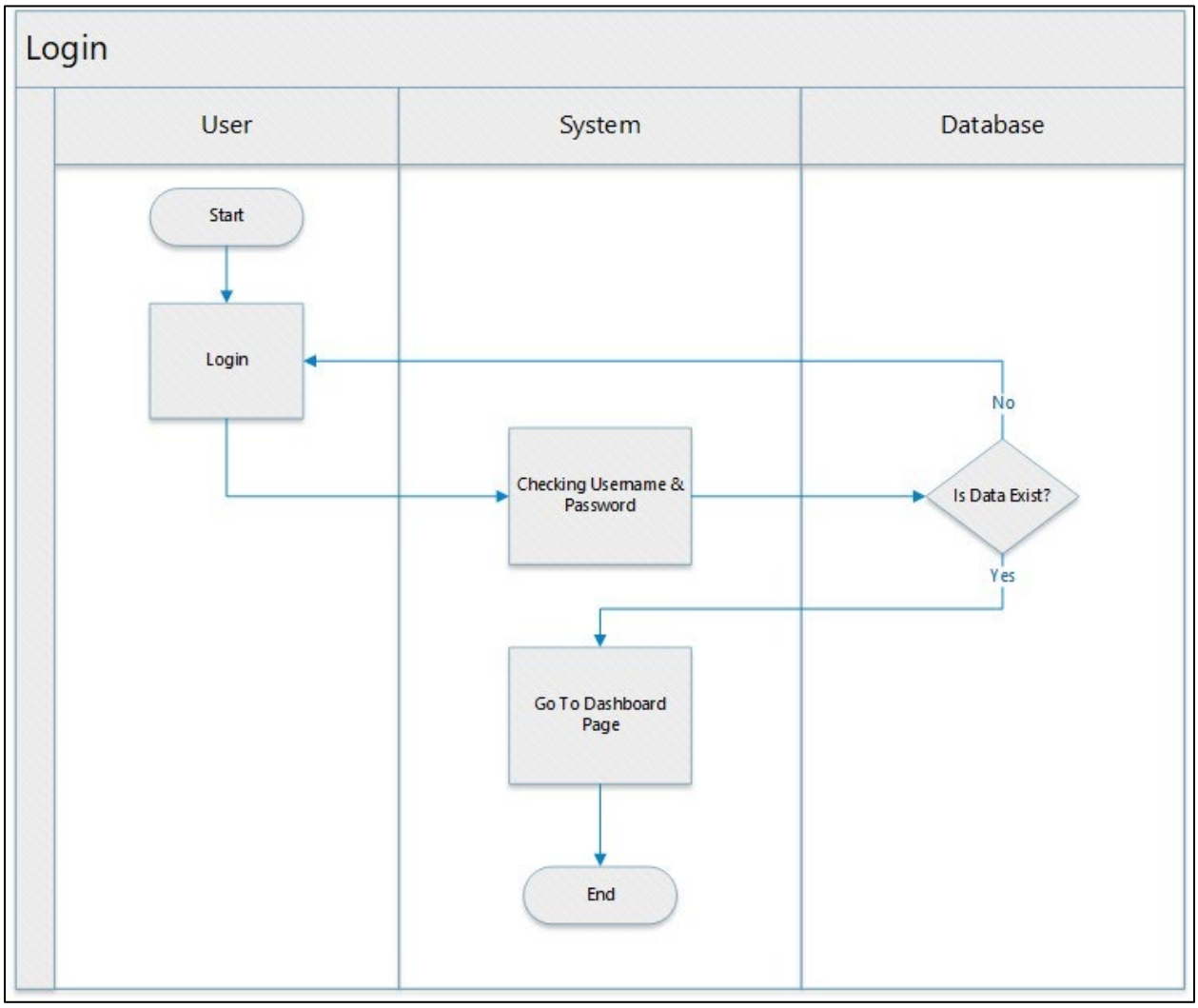
Priority	*High*	
Primary System Actor	All User	
Description	*Actor* melakukan trigger kepada sistem untuk menerjemahkan halaman.	
Precondition	Tidak ada	
Trigger	*Actor* ingin menerjemahkan halaman.	
Typical Course of Event	*Actor Action*	*System Response*
	1. Actor masuk ke menu *Sidebar Menu*. 3. Actor memilih bahasa pada opsi "Select Language"	2. Sistem menampilkan menu *Sidebar Menu*. 4. Sistem memanggil service dari Google Translate API dan memproses penerjemahan konten halaman.
Post Condition	Sistem akan mengaplikasikan Bahasa terjemahan yang dipilih untuk semua halaman di sistem.	

3.7 Swimlane Diagram

Swimlane Diagram adalah jenis diagram alur yang menggambarkan siapa yang melakukan apa dalam suatu urutan proses. Diagram ini menggunakan metafora jalur di kolam renang, diagram *swimlane* memberikan kejelasan dan akuntabilitas dengan menempatkan langkah-langkah proses dalam "*swimlane*" horizontal atau vertikal dari aktor atau subjek tertentu. Diagram ini dapat menunjukkan koneksi, komunikasi dan *handoff* antar jalur, dan dapat berfungsi untuk menyorot redundansi dan inefisiensi yang ada dalam suatu proses.

3.7.1 Halaman Login

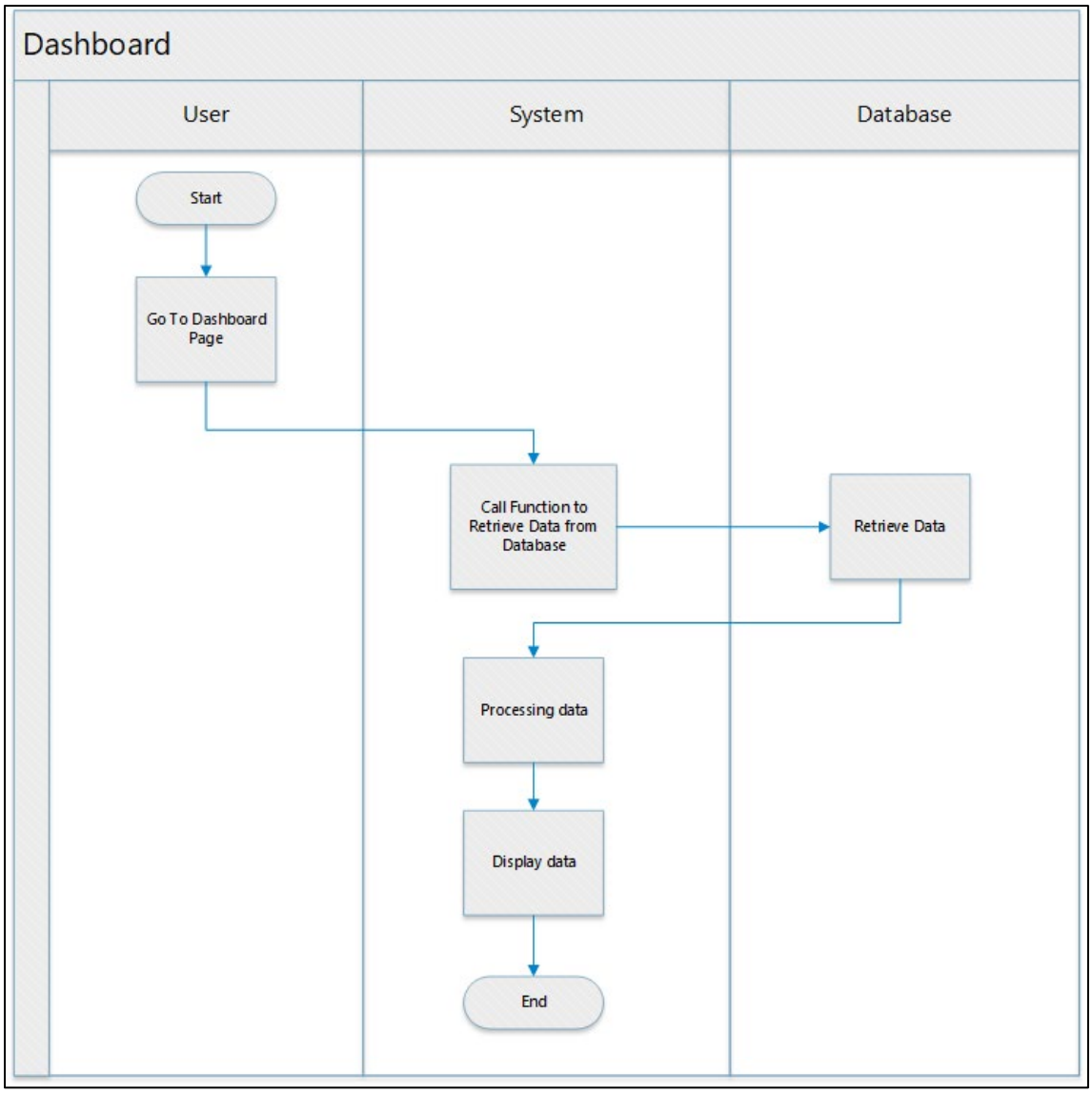
Halaman ini menampilkan seluruh data yang telah dirangkum dan sejarah transaksi dalam periode waktu tertentu. Pada Gambar 3.2 dijelaskan bagaimana



proses validasi data yang ketika user login sebelum masuk ke halaman *Dashboard*.

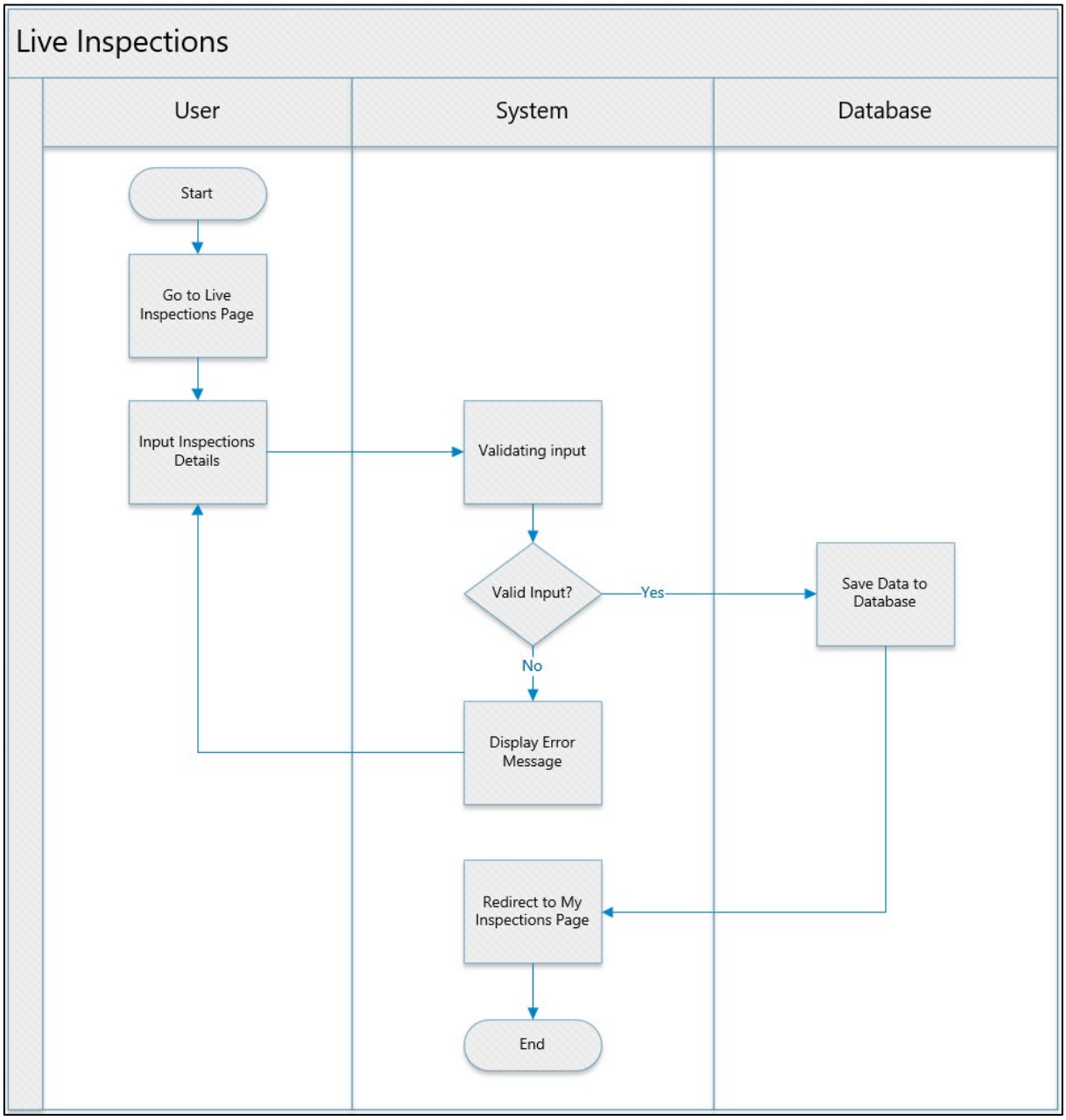
3.7.2 Halaman Dashboard

Halaman ini menampilkan seluruh data yang telah dirangkum dan sejarah transaksi dalam periode waktu tertentu. Pada Gambar 3.3 dijelaskan bagaimana proses pengambilan data yang dibutuhkan untuk di tampilkan pada halaman *Dashboard*.



ections

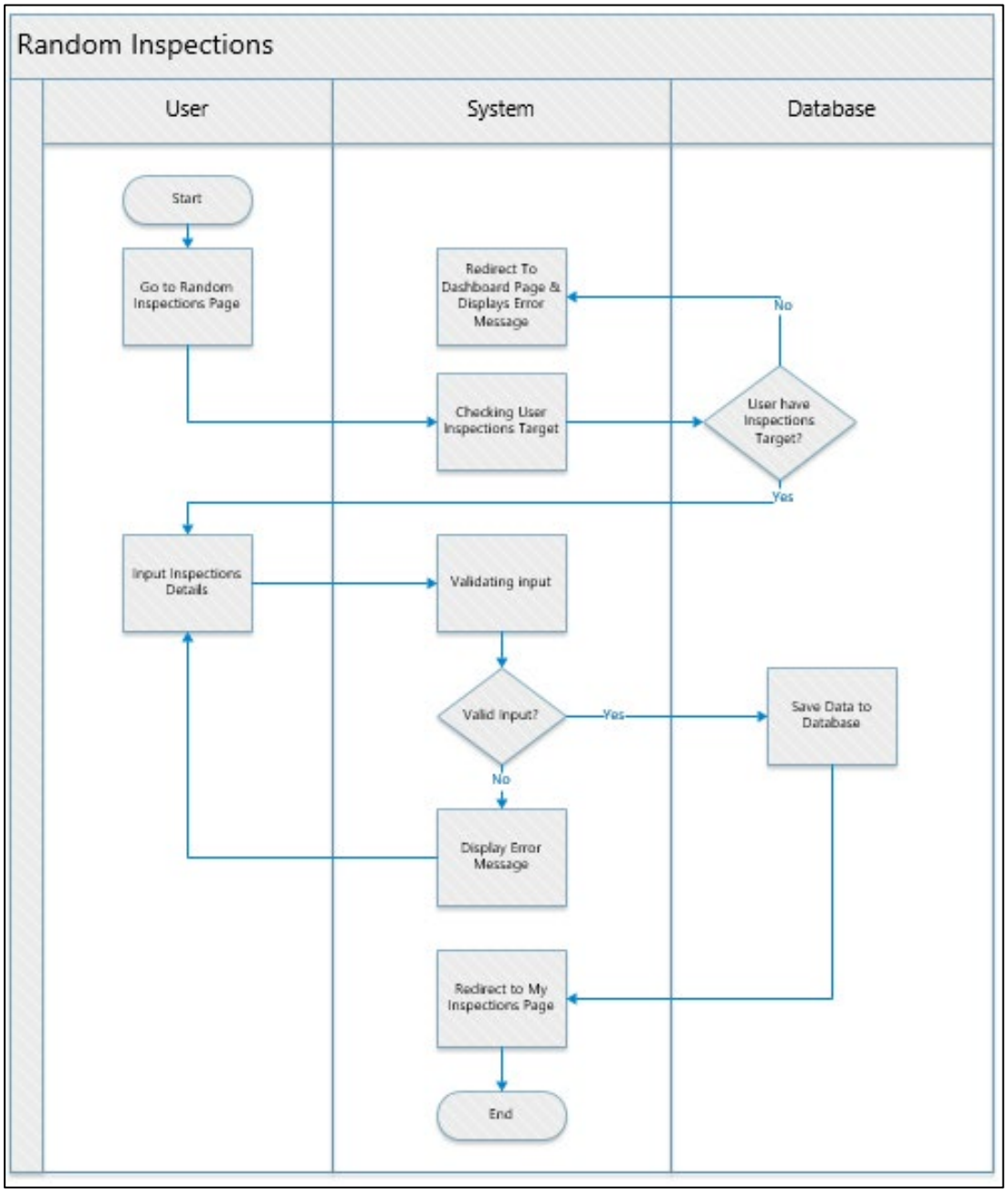
Fitur ini memungkinkan pengguna untuk menambahkan inspeksi baru. Pada Gambar 3.4 dijelaskan bagaimana proses penyimpanan data inspeksi ke dalam basis data.



Gambar 3.4 Swimlane Diagram Fitur *Live Inspections*

3.7.4 Fitur Random Inspections

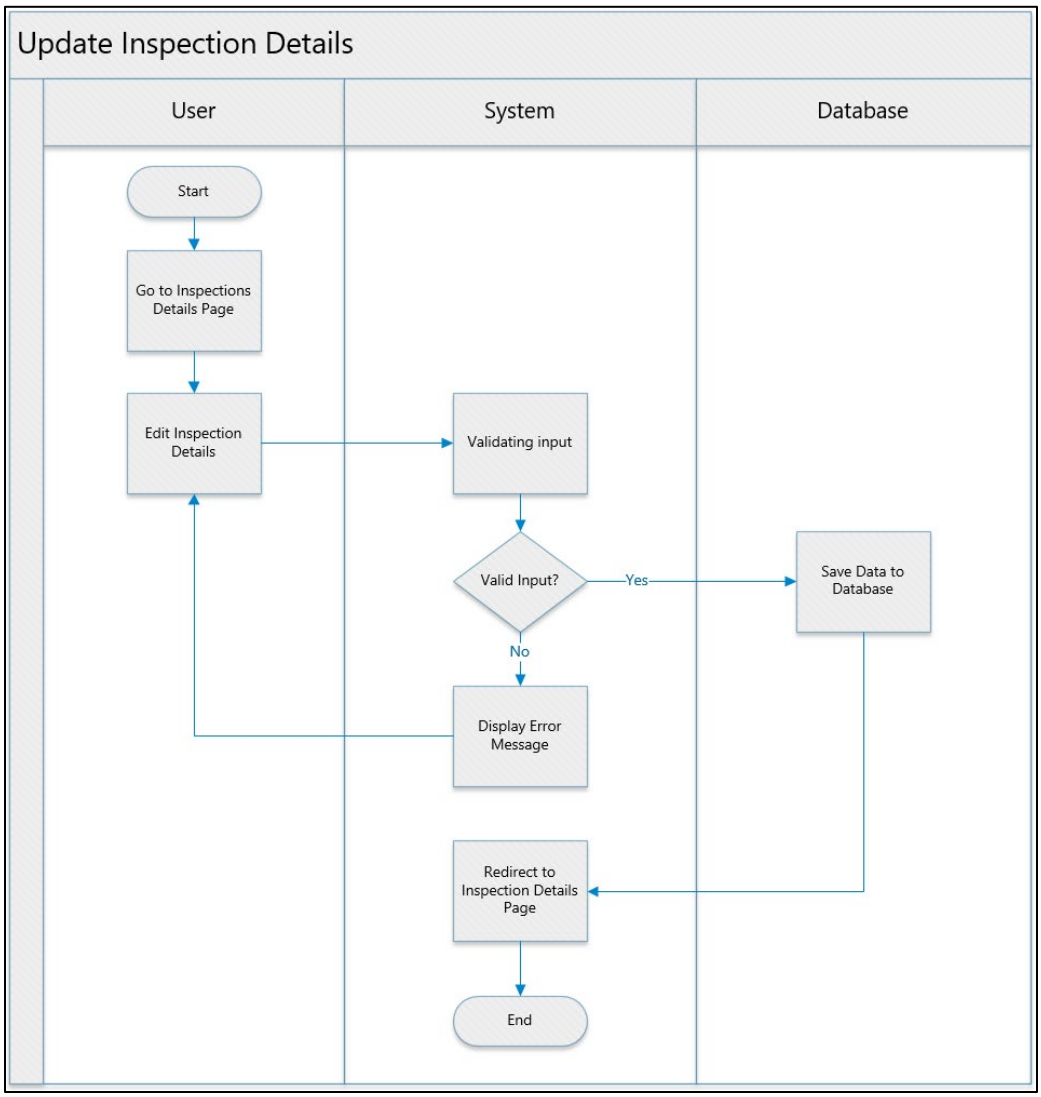
Fitur ini mirip dengan fitur Live Inspections, perbedaan fitur ini adalah sistem telah memilih secara acak departemen mana yang harus di inspeksi oleh user. Pada Gambar 3.5 dijelaskan bagaimana proses penyimpanan data inspeksi ke



dalam basis data.

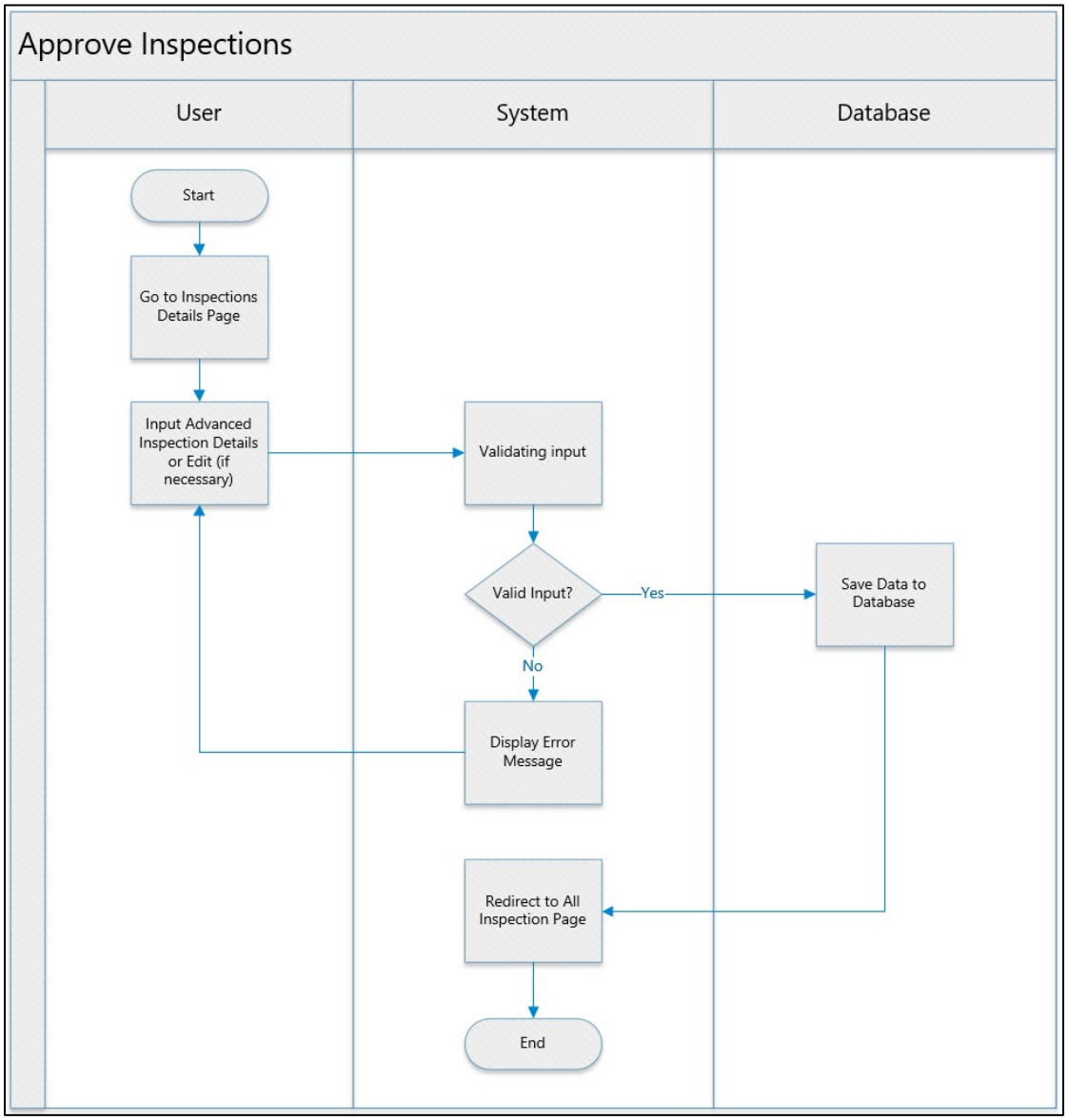
3.7.5 Fitur Update Inspection Details

Fitur ini memungkinkan pengguna untuk mengubah rincian data inspeksi yang telah masuk ke dalam sistem. Fitur ini hanya dapat dilakukan oleh *author*, *safety team* dan *admin*. Fitur ini tidak dapat digunakan jika status Inspection sudah menjadi "*in progress*" dan seterusnya. Pada Gambar 3.6 dijelaskan bagaimana proses update inspection details.


Update Inspection Details
User
System
Database
Start
Go to Inspections Details Page
Edit Inspection Details
Validating input
Valid Input?
Yes
No
Save Data to Database
Display Error Message
Redirect to Inspection Details Page
End

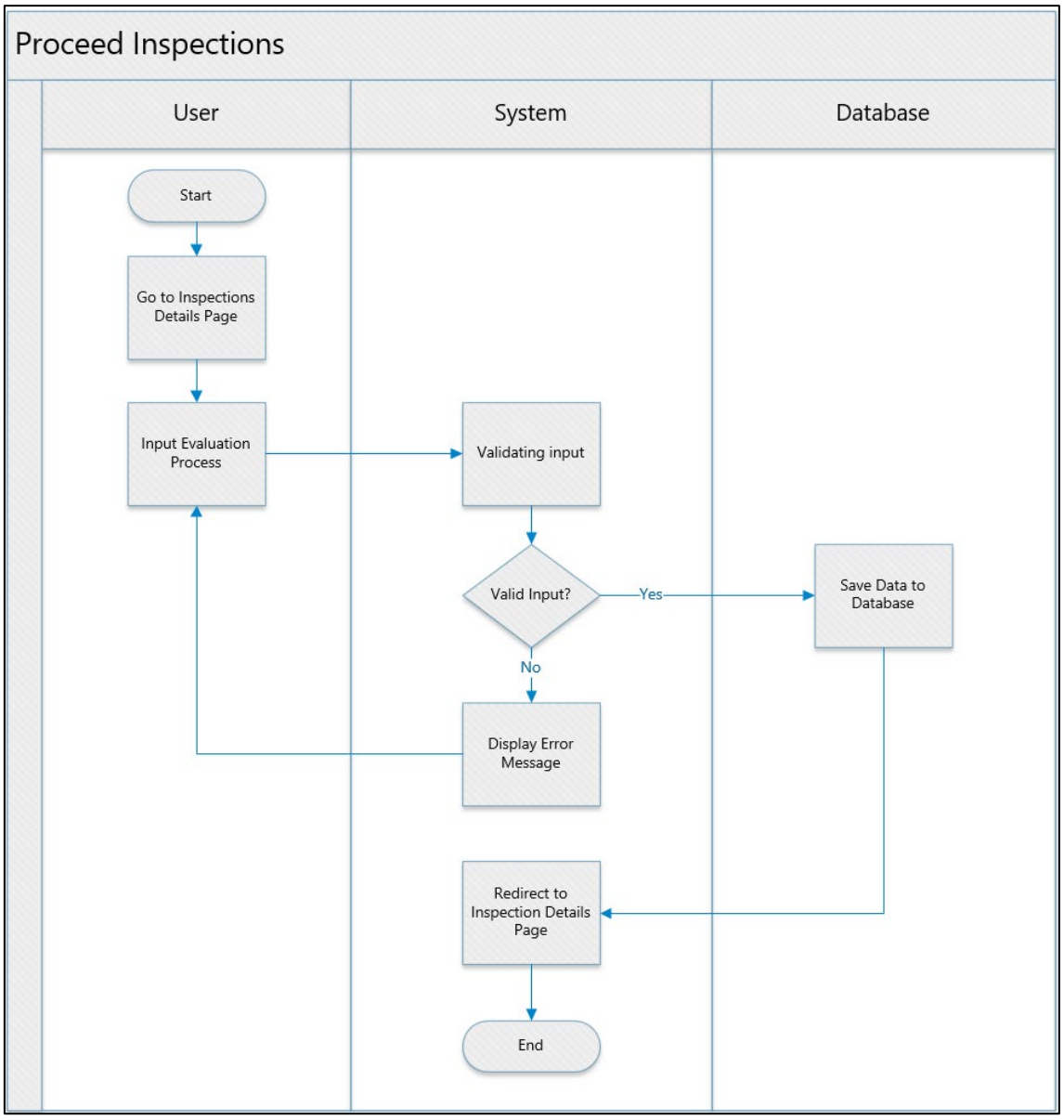
3.7.6 Fitur Approve Inspection

Fitur ini memungkinkan pengguna untuk melakukan approval terhadap inspeksi yang telah masuk ke dalam sistem. Fitur ini hanya dapat dilakukan oleh user dengan level "*safety team*". Pada Gambar 3.7 dijelaskan bagaimana proses *Approve Inspection*.



3.7.7 Fitur Proceed Inspection

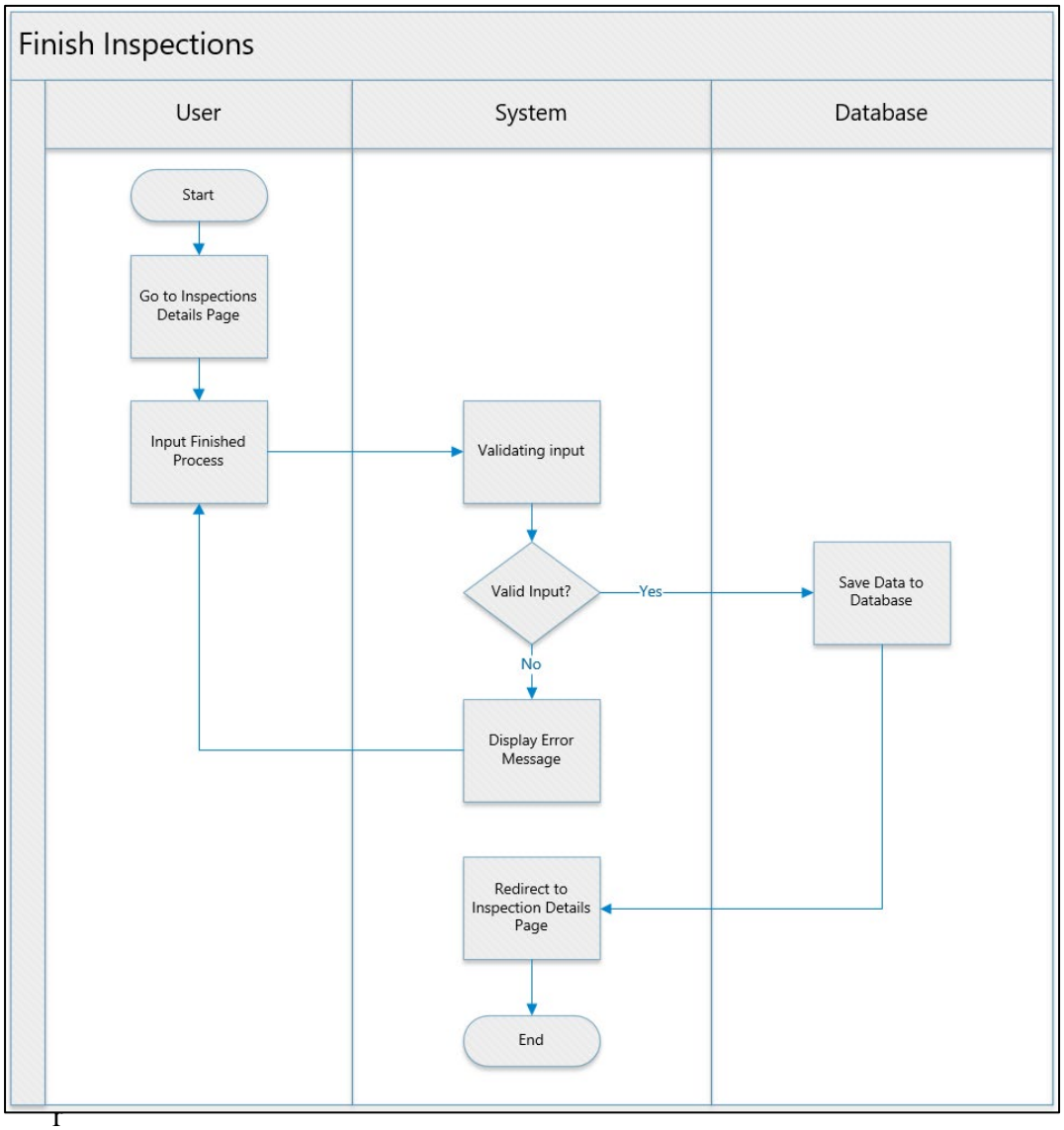
Fitur ini memungkinkan pengguna dapat melaporkan proses evaluasi terhadap inspeksi yang telah masuk ke dalam sistem. Fitur ini hanya dapat dilakukan oleh PIC *(Person In Charge)* yang telah dipilih oleh *safety team.* Pada Gambar 3.8 dijelaskan bagaimana proses *Proceed*



Inspection.

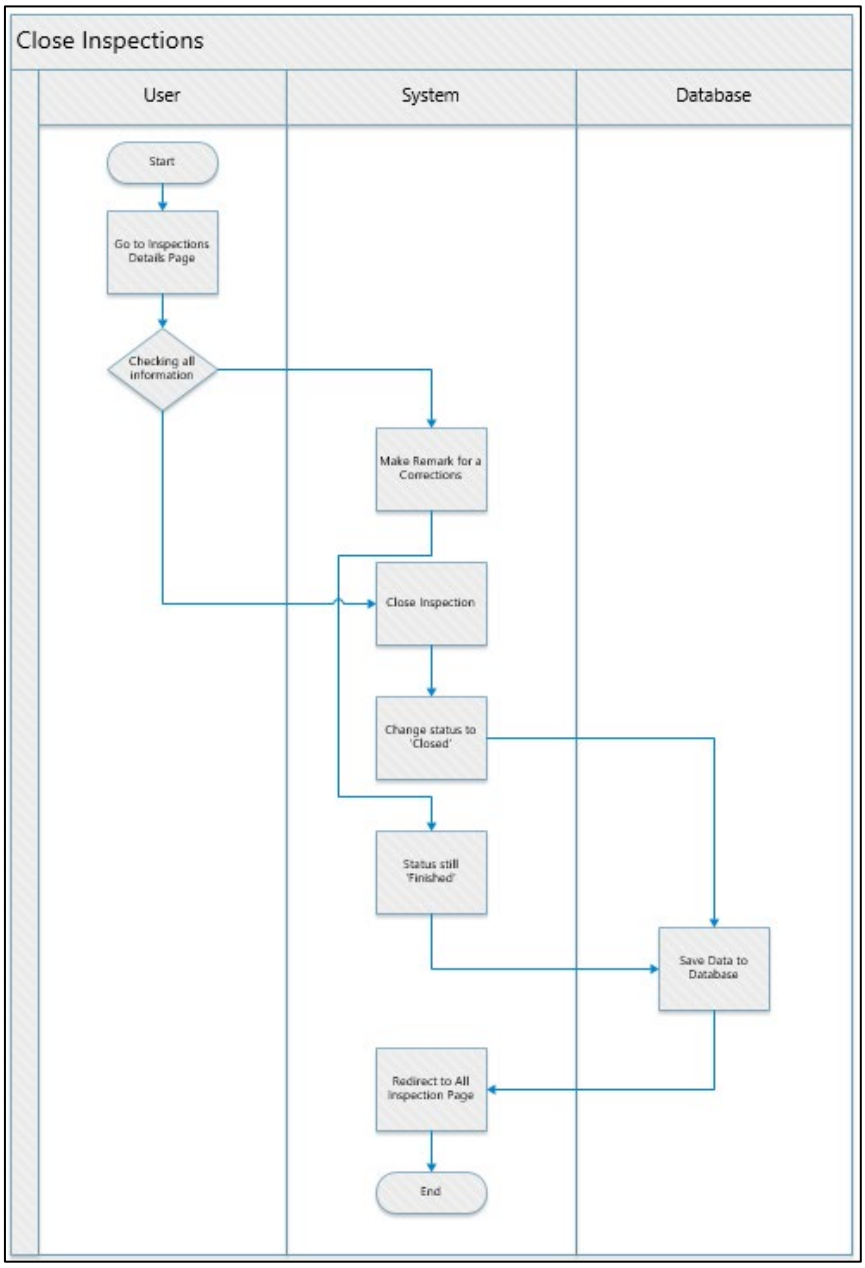
3.7.8 Fitur Finish Inspection

Fitur ini memungkinkan pengguna untuk dapat melaporkan hasil evaluasi terhadap inspeksi yang telah dikerjakan sebelumnya. Fitur ini hanya dapat dilakukan oleh PIC *(Person In Charge)* yang telah dipilih oleh *safety team.* Pada Gambar 3.9 dijelaskan bagaimana proses *Finish I*



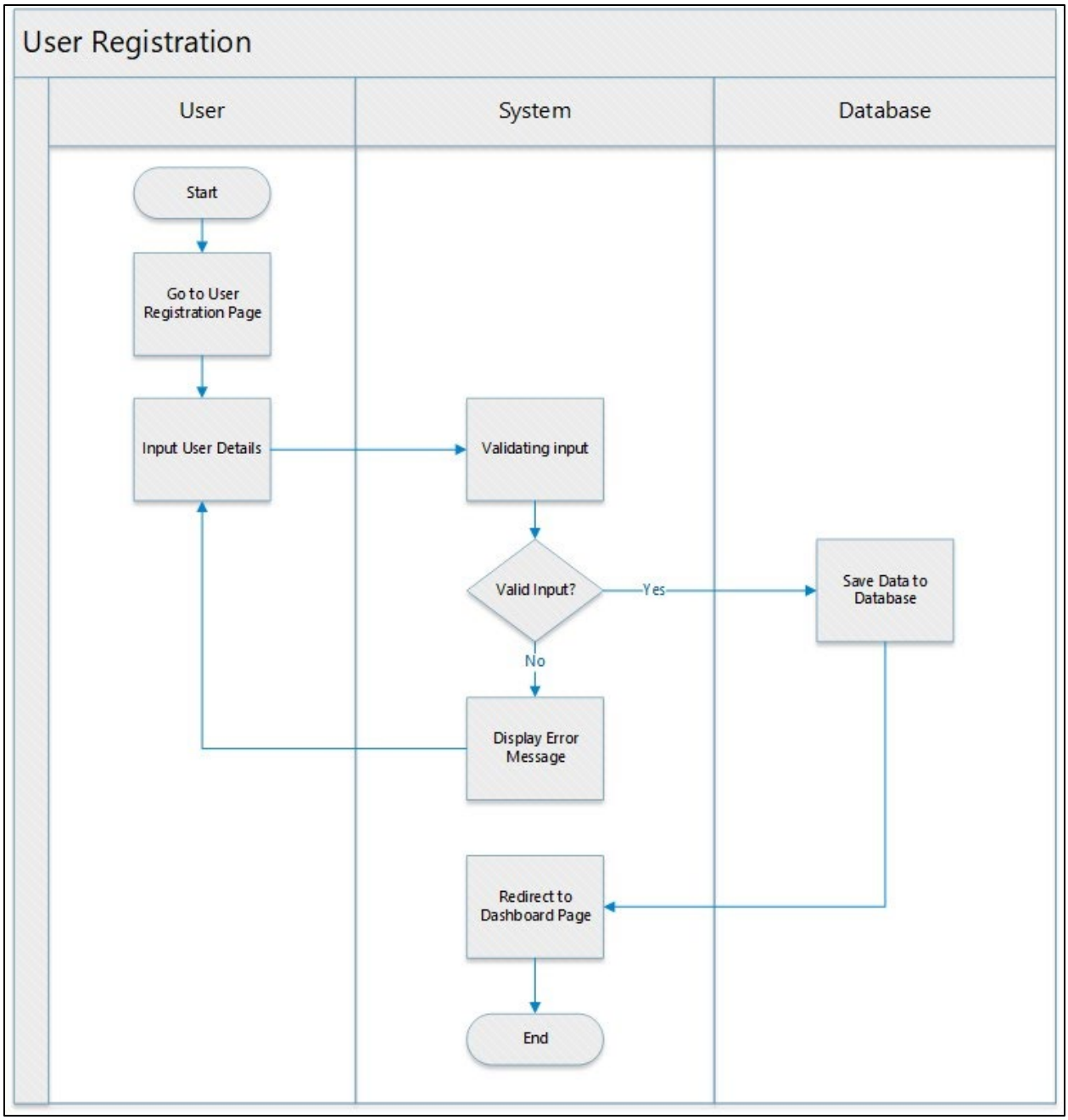
Close Inspection

Fitur ini memungkinkan pengguna untuk dapat melaporkan hasil evaluasi terhadap inspeksi yang telah dikerjakan sebelumnya. Fitur ini hanya dapat dilakukan oleh user dengan level "*safety team"*. Pada Gambar 3.10 dijelaskan bagaimtana proses *Finish Inspection.*



3.7.10 Fitur User Registration

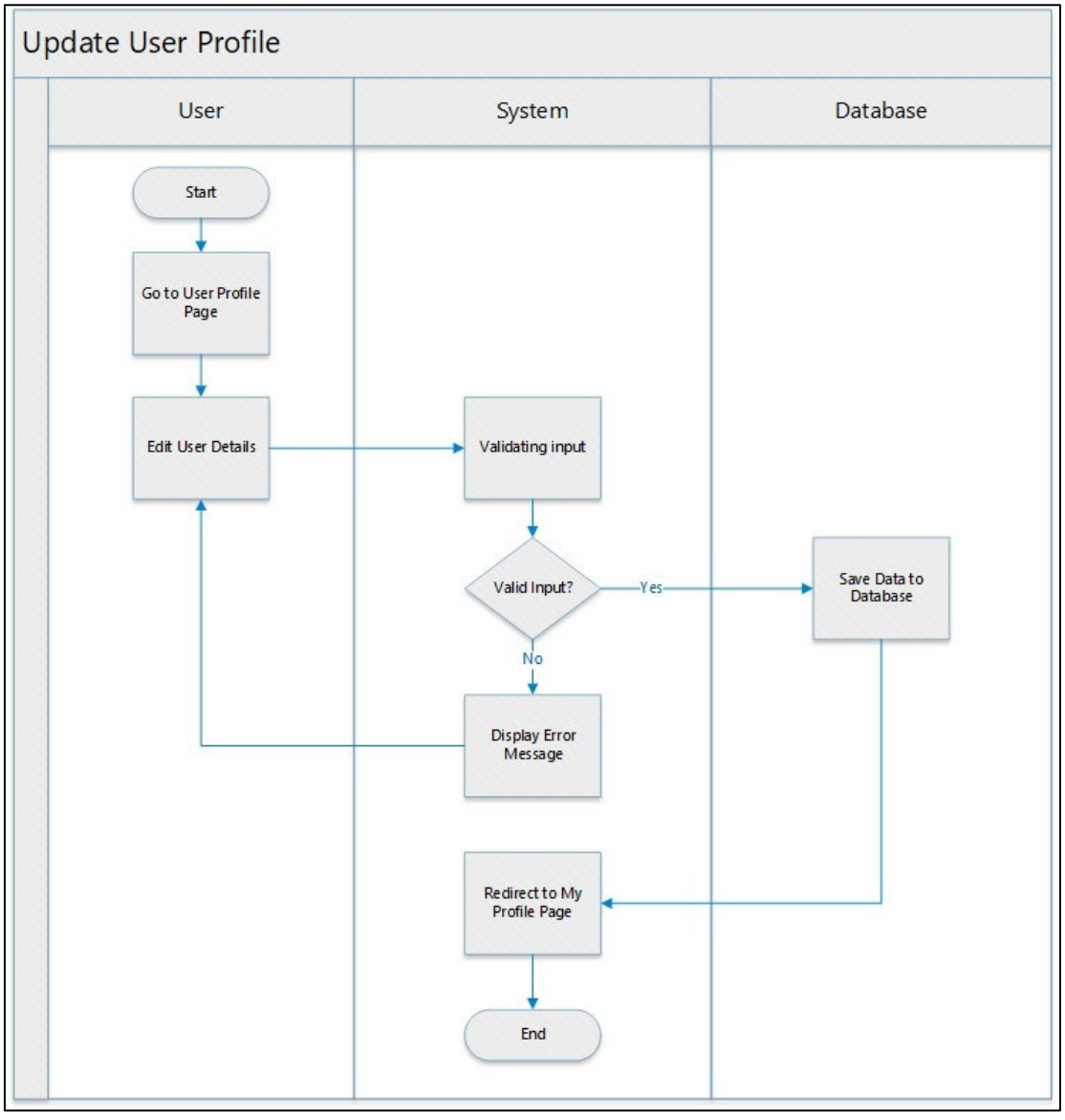
Fitur ini memungkinkan pengguna untuk menambahkan user baru. Pada Gambar 3.11 dijelaskan bagaimana proses penyimpanan data user ke dalam basis data.



3.7.11 Fitur Update User Details

Fitur ini memungkinkan pengguna untuk mengubah rincian data inspeksi yang telah masuk ke dalam sistem. User hanya dapat melakukan fitur ini terhadap data user itu sendiri, kecuali user dengan level "*administrator*". Pada Gambar 3.12 dijelaskan bagaimana proses update

user profile.

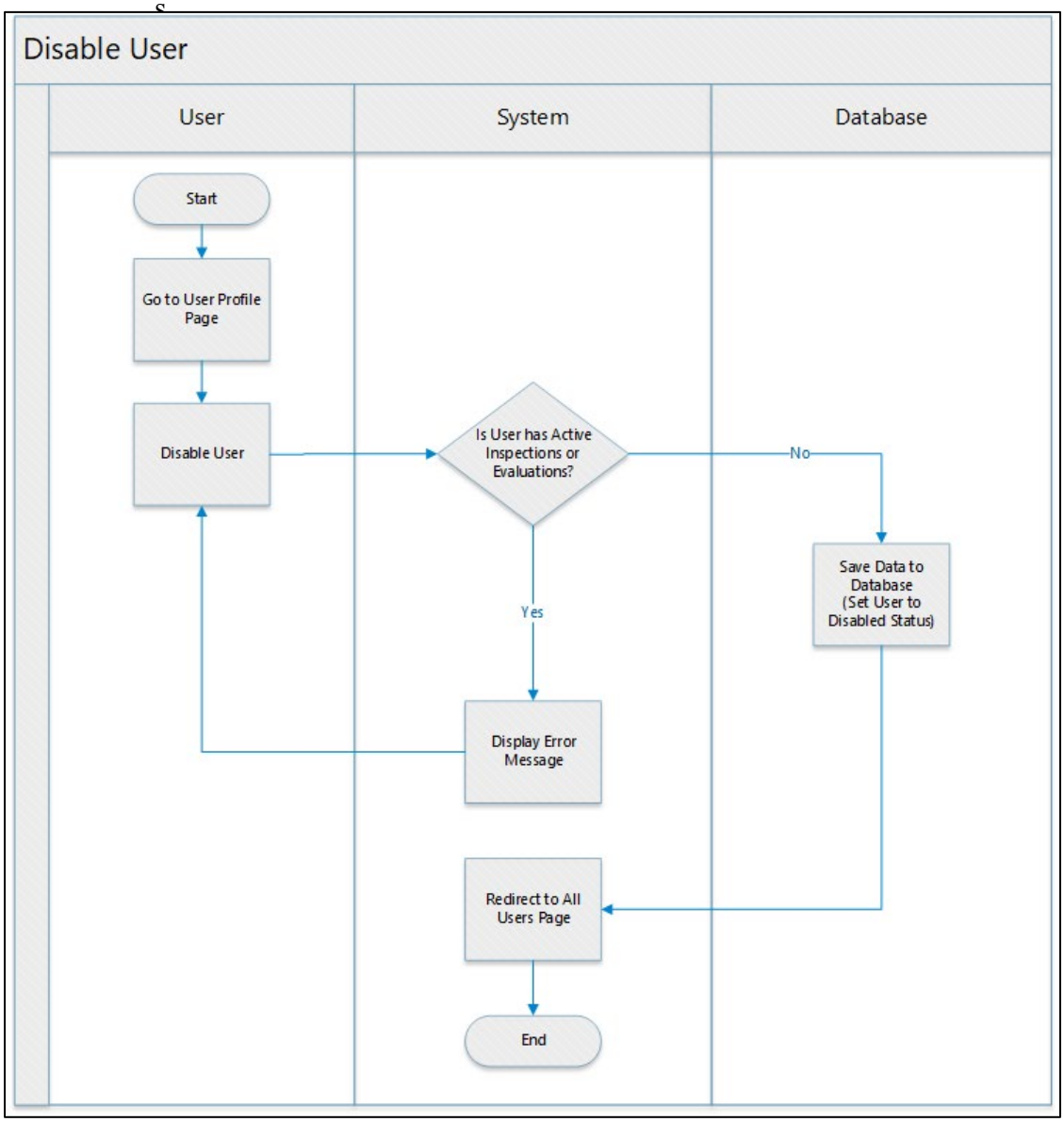


G

3.7.12 Fitur Disable User

Fitur ini memungkinkan pengguna untuk menonaktifkan user secara individual. Fitur ini hanya dapat dilakukan oleh user dengan level *"administrator"*. Pada Gambar 3.13 dijelaskan bagaimana proses disable u

s

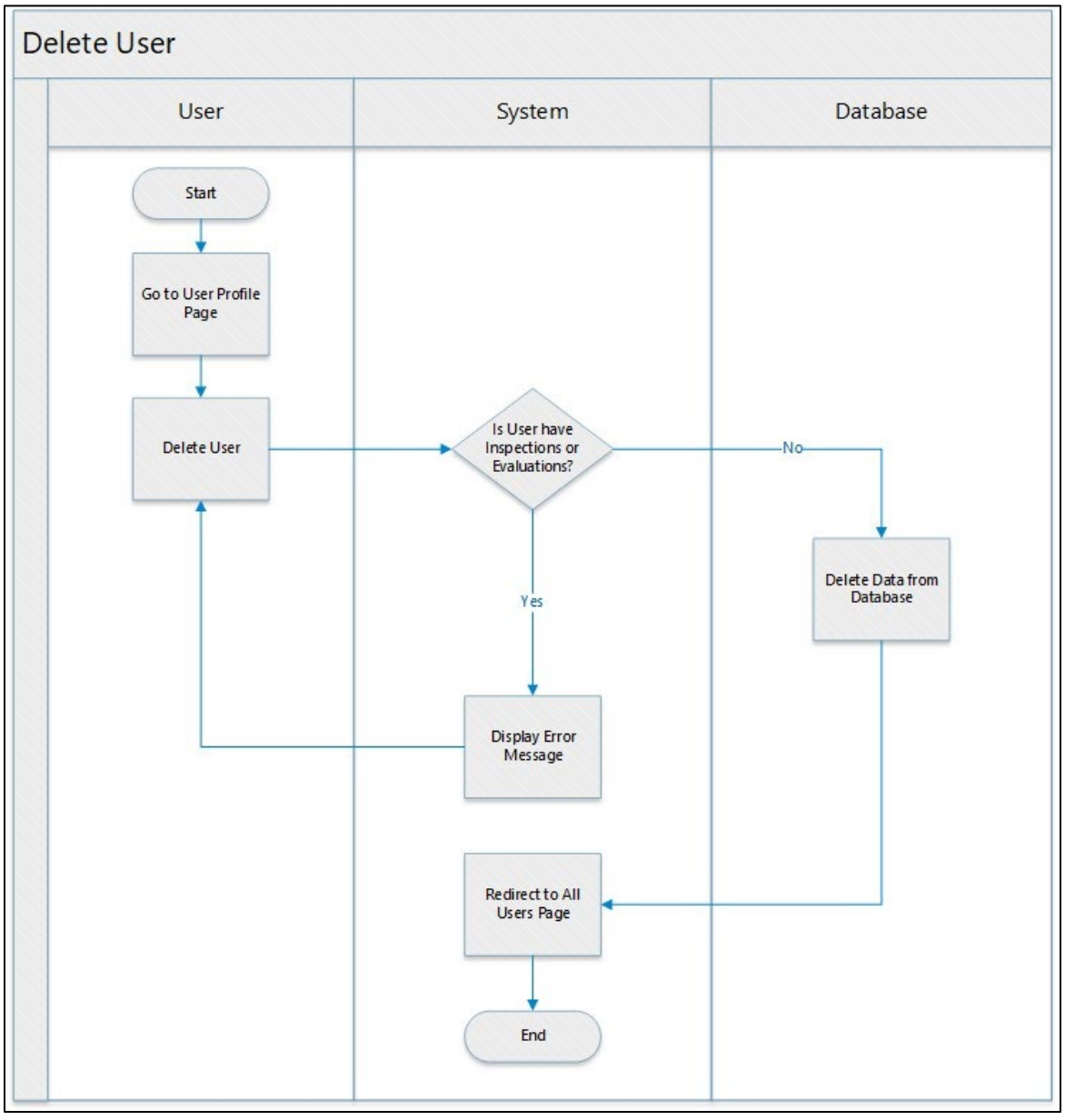


3.7.13 Fitur Delete User

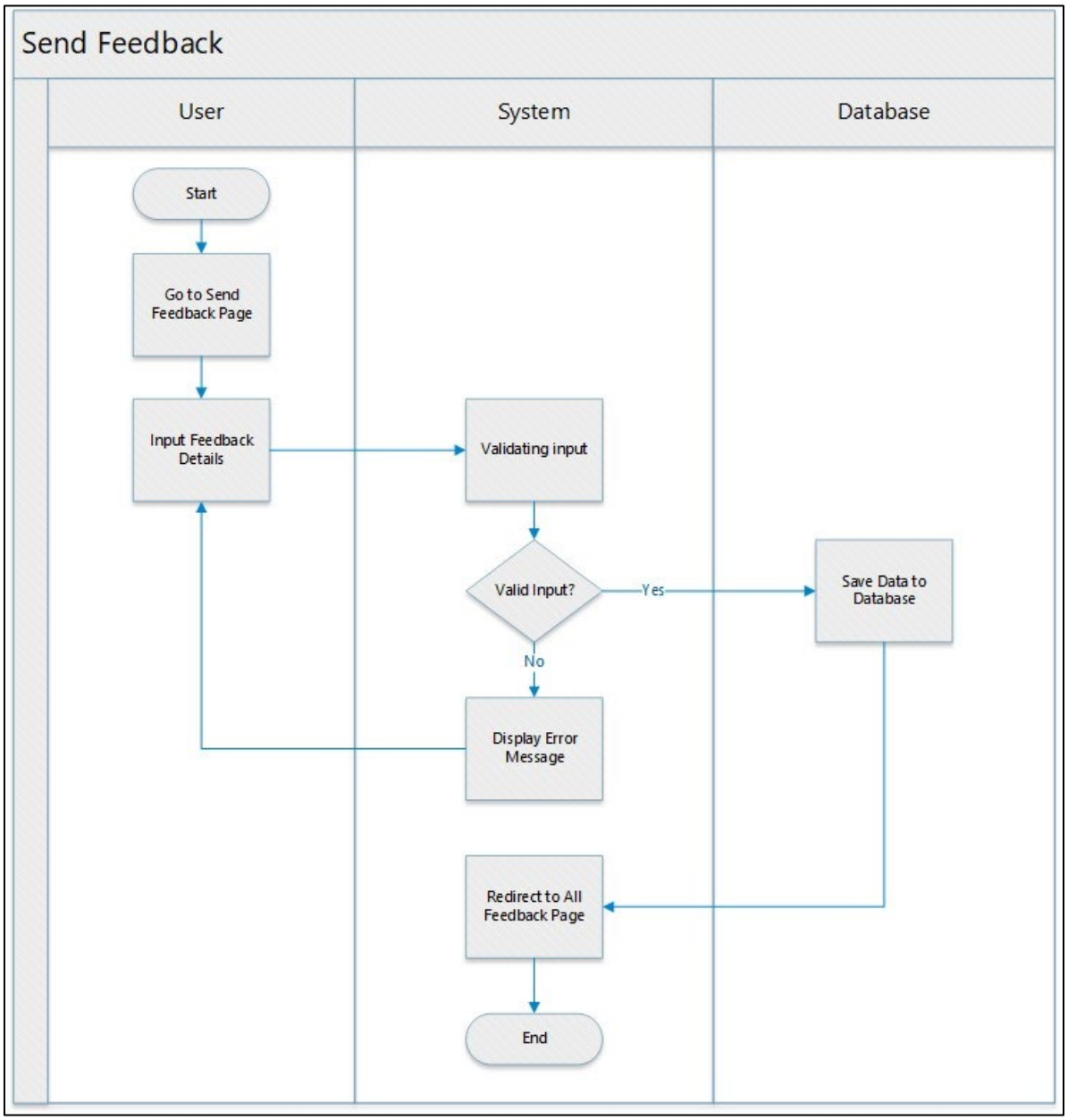
Fitur ini memungkinkan pengguna untuk menambahkan inspeksi baru. Fitur ini hanya dapat dilakukan oleh user dengan level *"administrator"*. Pada Gambar 3.14 dijelaskan bagaimana proses

penghapusan data user dari basis data.

3.7.14 Fitur Send Feedback



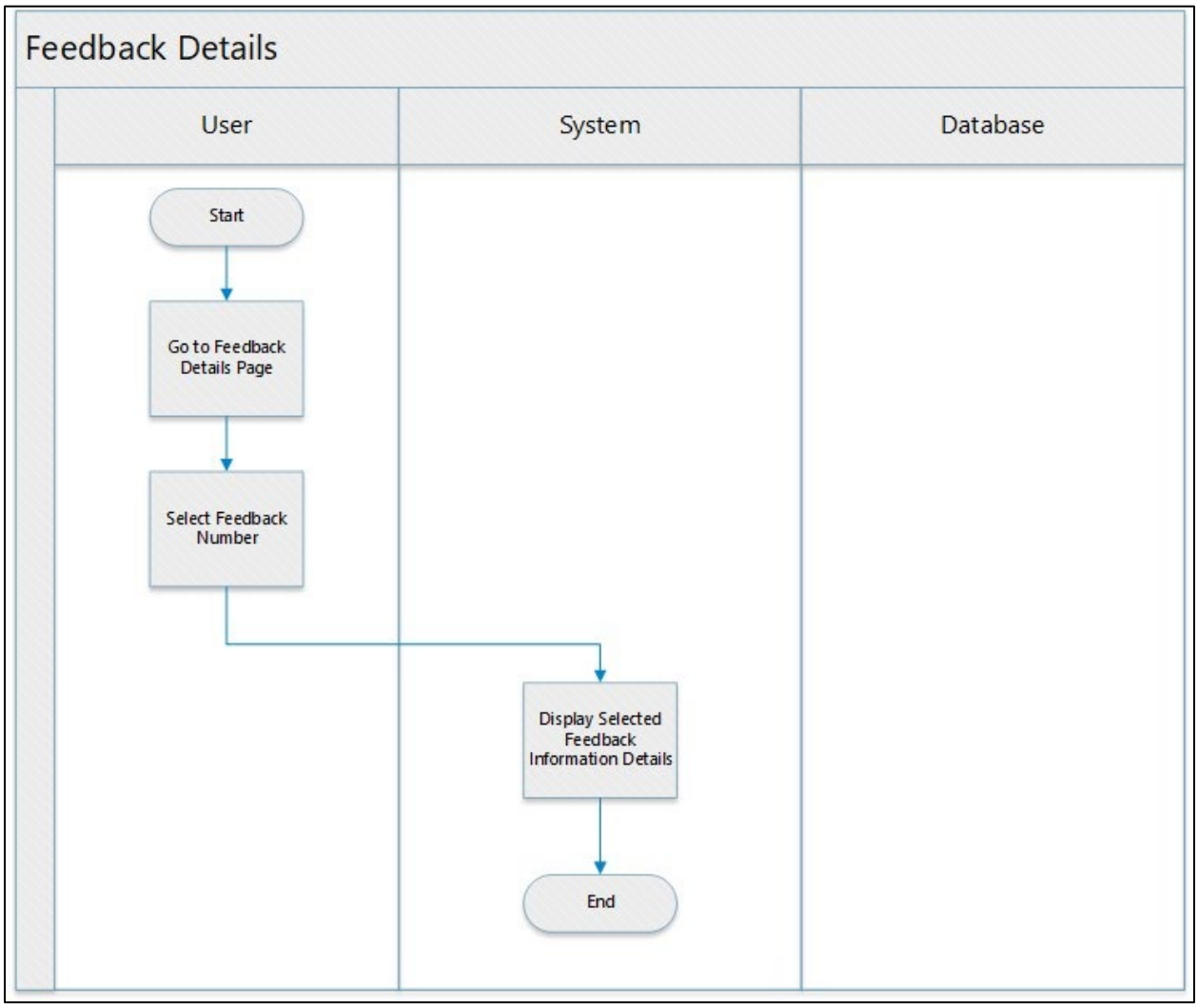
Fitur ini memungkinkan pengguna untuk menambahkan data feedback baru. Pada Gambar 3.15 dijelaskan bagaimana proses penyimpanan data feedback



ke dalam basis data.

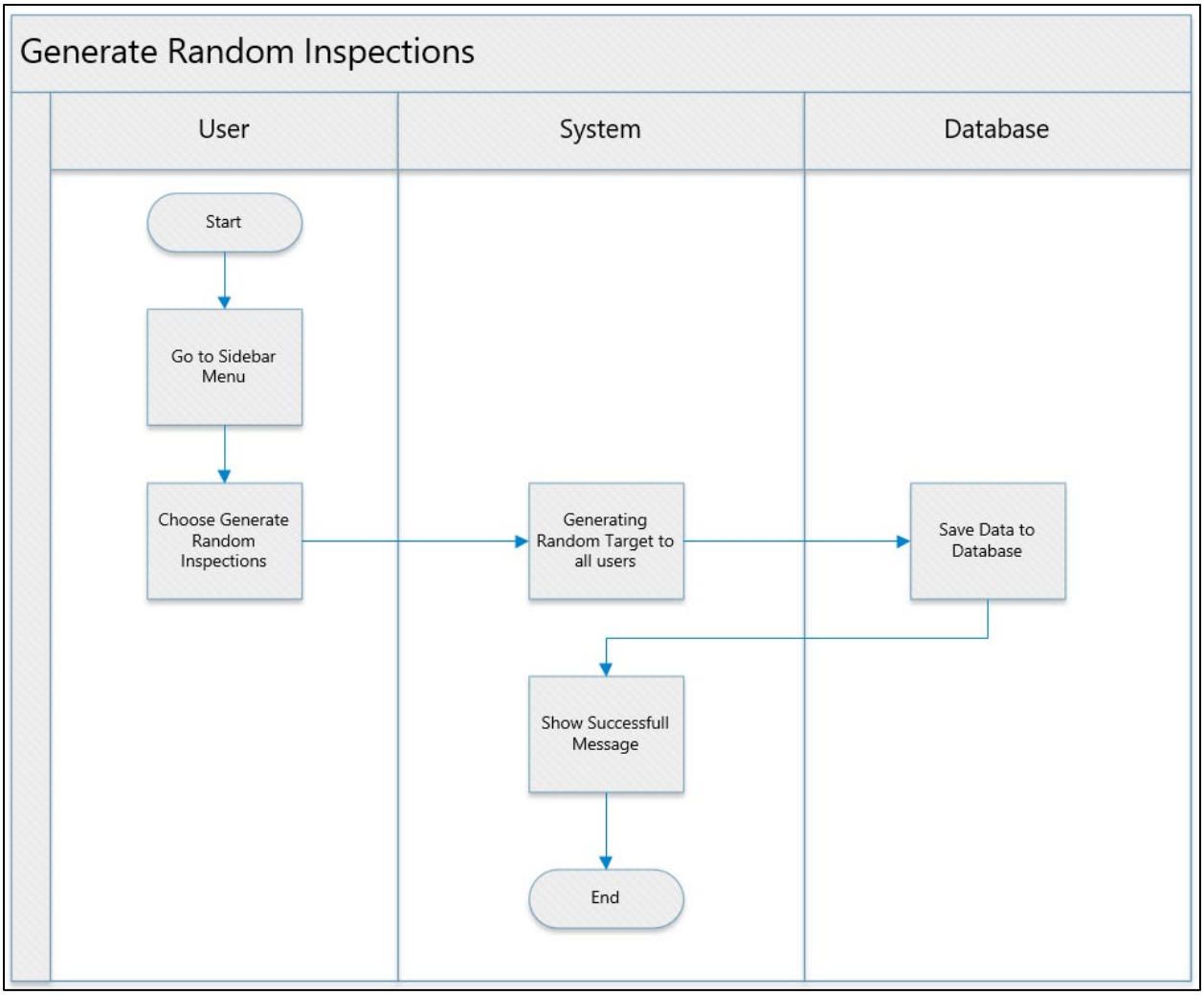
3.7.15 Halaman Feedback Details

Halaman ini menampilkan detail informasi mengenai feedback secara individual. Pada Gambar 3.16 dijelaskan bagaimana proses pengambilan data yang dibutuhkan untuk tampilan halaman *Feedback Details*.



3.7.16 Halaman Generate Random Inspections

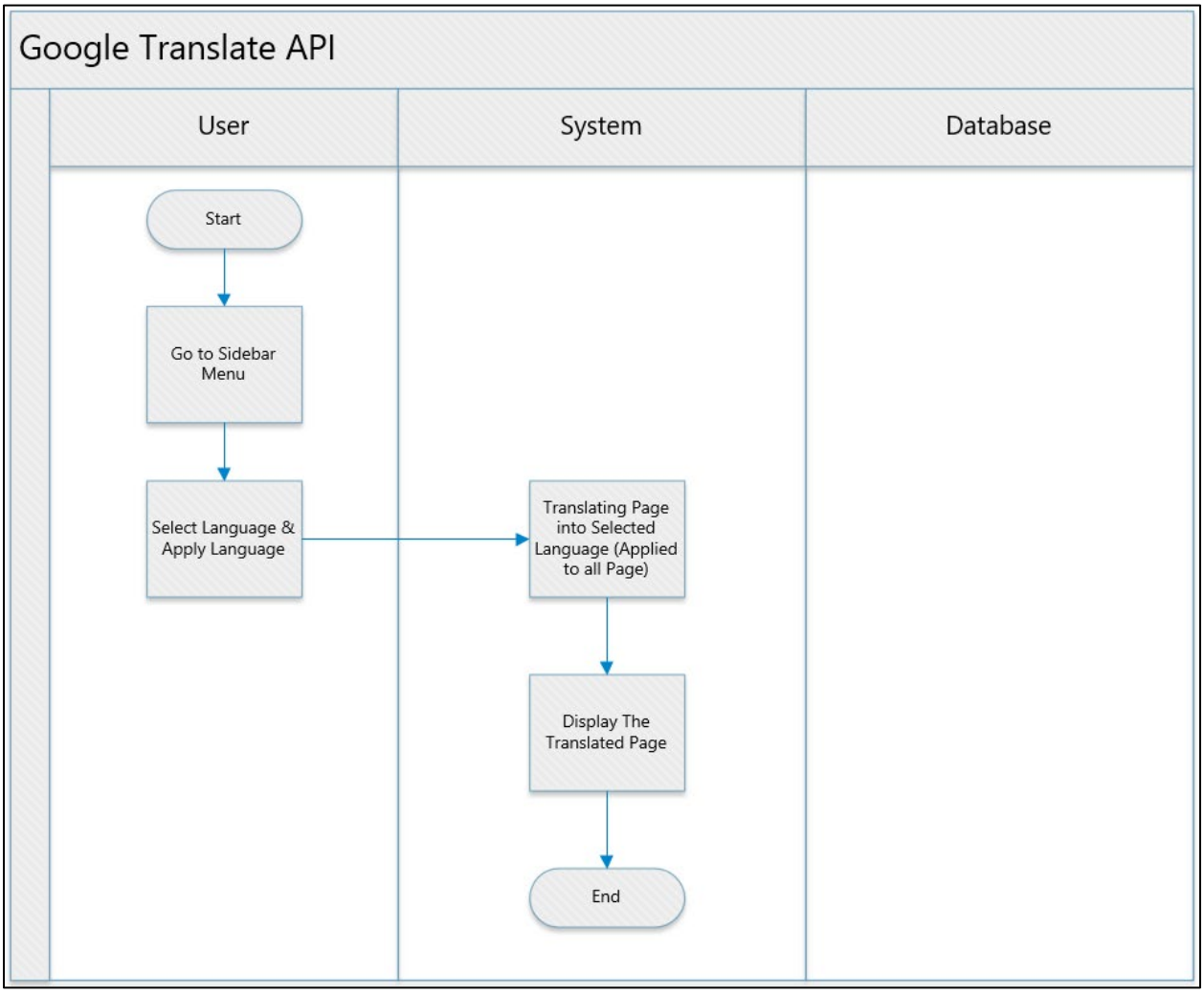
Fitur ini memungkinkan pengguna dapat melakukan trigger kepada sistem untuk melakukan pemilihan departemen yang akan menjadi *target inspeksi* secara acak kepada masing-masing user. Pada Gambar 3.17 dijelaskan bagaimana proses



G *generate random inspections*.

3.7.17 Fitur Menerjemahkan Halaman *(Google Translate API)*

Fitur ini memungkinkan pengguna dapat menerjemahkan halaman ke bahasa lain. Fitur ini didukung oleh *Google Translate* sebagai *engine* dari fitur tersebut. Pada Gambar 3.18 dijelaskan bagaimana proses menerjemahkan



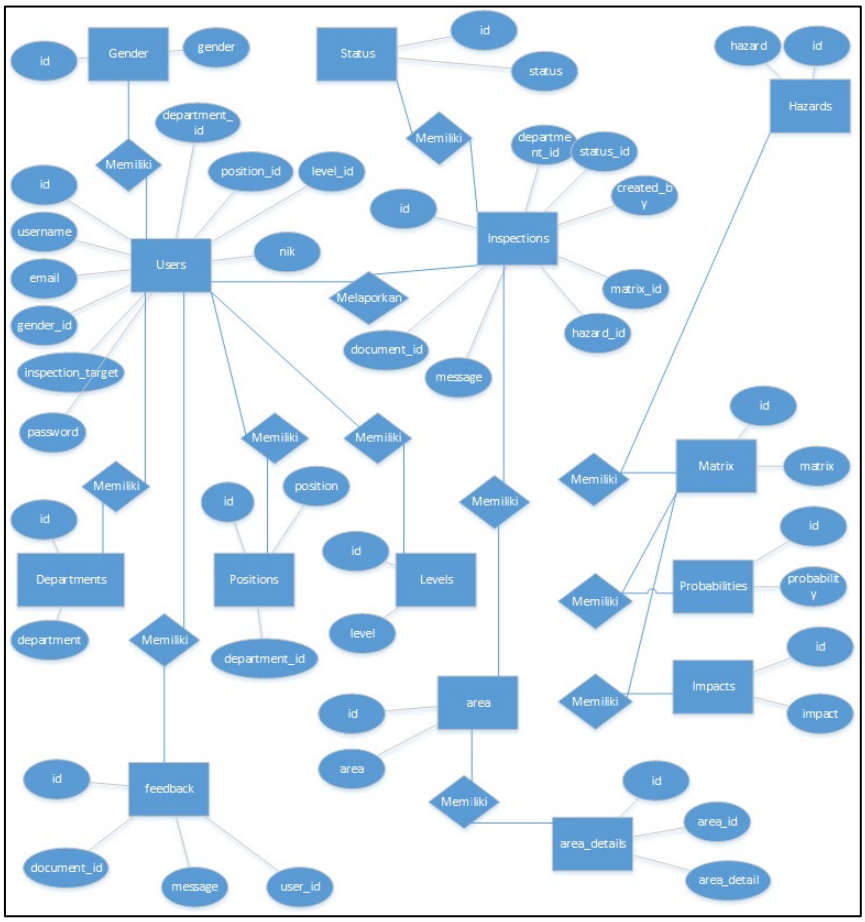
halaman menggunakan *Google Translate*.

BAB IV

PERANCANGAN SISTEM

4.1 Entity Relation Diagram

Entity Relationship Diagram atau ERD adalah sebuah diagram struktural yang digunakan untuk merancang sebuah database. Komponen dari ERD tediri dari entitas, atribut entitas, *primary key*, *foreign key*, dan relasi itu sendiri [12]. Tujuan utama ERD adalah membantu pengembang untuk merancang dan melihat struktur tampilan database sebelum pengembang benar-benar membuat sistem. *Entity Relationship Diagram* untuk aplikasi *Safety Management System* dapat dilihat pada Gambar 4.1.



4.2 Perancangan Basis Data

Perancangan basis data merupakan betuk gambaran dalam merancang entitas yang digunakan pada sistem. Rancangan basis data mendeskripsikan data apa saja yang disimpan ke dalam basis data. Aplikasi *Safety Management System* memanfaatkan layanan XAMPP untuk mengatur konfigurasi web server dan database, Apache sebagai *web server* dan MySQL sebagai *database server*. Rancangan tabel basis data aplikasi *Safety Management System* dapat dilihat di Tabel 4.1 sampai Tabel 4.6

Tabel 4.1 Rancangan tabel *Users*

No.	Atribut	Deskripsi
1	id	Atribut identitas tabel *Users.*
2	nik	Atribut nomor identitas karyawan.
2	username	Atribut username akun.
3	email	Atribut email akun.
4	first_name	Atribut nama depan akun.
5	last_name	Atribut nama belakang akun.
6	initial_name	Atribut inisial nama akun.
7	gender_id	Atribut jenis kelamin akun.
8	department_id	Atribut departemen akun.
9	Position_id	Atribut posisi jabatan akun.
10	level_id	Atribut level user akun.
11	inspections_target_id	Atribut inspeksi yang telah diacak oleh sistem.

12	profile_picture	Atribut gambar profil akun.
13	Join_date	Atribut informasi tanggal karyawan bergabung.
14	email_verified_at	Atribut waktu email telah diverifikasi.
15	password	Atribut password akun.
16	remember_token	Atribut token untuk fitur lupa password.
17	created_at	Atribut waktu data telah dibuat.
18	updated_at	Atribut waktu data terakhir kali diperbarui.

Tabel 4.2 Rancangan tabel *Genders*

No.	**Atribut**	**Deskripsi**
1	id	Atribut identitas tabel *Genders.*
2	gender	Atribut jenis kelamin.
3	created_at	Atribut waktu data telah dibuat.
4	updated_at	Atribut waktu data terakhir kali diperbarui.

Tabel 4.3 Rancangan tabel *Departments*

No.	**Atribut**	**Deskripsi**
1	id	Atribut identitas tabel *Departments.*
2	department	Atribut nama departemen.
3	head_id	Atribut penanggung jawab departemen.
4	properties	Atribut informasi tambahan.
5	created_at	Atribut waktu data telah dibuat.

6	updated_at	Atribut waktu data terakhir kali diperbarui.

Tabel 4.4 Rancangan tabel *Positions*

No.	Atribut	Deskripsi
1	id	Atribut identitas tabel *Positions*.
2	position	Atribut nama posisi jabatan.
3	department_id	Atribut nama departemen posisi jabatan.
4	level_id	Atribut level posisi jabatan.
5	created_at	Atribut waktu data telah dibuat.
6	updated_at	Atribut waktu data terakhir kali diperbarui.

Tabel 4.5 Rancangan tabel *Levels*

No.	Atribut	Deskripsi
1	**id**	Atribut identitas tabel *Levels*.
2	**level**	Atribut tingkatan pada user.
3	**created_at**	Atribut waktu data telah dibuat.
4	**updated_at**	Atribut waktu data terakhir kali diperbarui.

Tabel 4.6 Rancangan tabel *Inspections*

No.	Atribut	Deskripsi
1	**id**	Atribut identitas tabel *Inpections*.
2	**document_id**	Atribut nomor dokumen inspeksi.

3	**inspection_type**	Atribut jenis inspeksi.
4	**initial_picture1**	Atribut gambar inisiasi inspeksi 1.
5	**initial_picture2**	Atribut gambar inisiasi inspeksi 2.
6	**initial_picture3**	Atribut gambar inisiasi inspeksi 3.
7	**evaluation_picture**	Atribut gambar pengerjaan inspeksi.
8	**postpone_picture**	Atribut gambar penundaan inspeksi.
9	**finished_picture**	Atribut gambar hasil penyelesaian inspeksi.
10	**subject**	Atribut subyek inspeksi.
11	**message**	Atribut pesan inspeksi.
12	**area**	Atribut area/tempat inspeksi dilakukan.
13	**remark_open**	Atribut keterangan ketika inspeksi dalam status "*open*".
14	**remark_in_process**	Atribut keterangan ketika inspeksi dalam status "*in process*".
15	**remark_finished**	Atribut keterangan ketika inspeksi dalam status "*finished*".
16	**remark_deleted**	Atribut keterangan ketika inspeksi dalam status "*deleted*".
17	**department_id**	Atribut departemen yang ditunjuk atas inspeksi.
18	**pic**	Atribut user yang ditunjuk atas inspeksi.
19	**hazard_id**	Atribut *hazard category*/jenis bahaya dari inspeksi.
20	**warning_id**	Atribut *warning level*/tingkat peringatan dari inspeksi.

21	**matrix_id**	Atribut safety matrix dari inspeksi.
22	**evaluation_target**	Atribut waktu penyelesaian yang telah ditentukan untuk evaluasi atas inspeksi yang telah dilakukan.
23	**evaluation_date**	Atribut waktu penyelesaian atas evaluasi atas inspeksi yang telah dilakukan.
24	**Status_id**	Atribut status terkini dari inspeksi.
24	**readed**	Atribut informasi bahwa dokumen inspeksi telah dibaca oleh team safety.
25	**created_by**	Atribut informasi mengenai user yang membuat inspeksi tersebut.
26	**updated_by**	Atribut informasi mengenai user yang terakhir kali memperbarui data inspeksi tersebut.
27	**created_at**	Atribut waktu data telah dibuat.
28	**updated_at**	Atribut waktu data terakhir kali diperbarui.

Tabel 4.7 Rancangan tabel *Timelines*

No.	**Atribut**	**Deskripsi**
1	**id**	Atribut identitas tabel *Timelines.*
2	**document_id**	Atribut nomor dokumen inspeksi.
3	**activity**	Atribut jenis data.
5	**matrix_id**	Atribut gambar inisiasi inspeksi 2.
6	**Status**	Atribut gambar inisiasi inspeksi 3.
7	**created_by**	Atribut informasi mengenai user yang membuat

		inspeksi tersebut.
8	**updated_by**	Atribut informasi mengenai user yang terakhir kali memperbarui data inspeksi tersebut.
9	**created_at**	Atribut waktu data telah dibuat.
10	**updated_at**	Atribut waktu data terakhir kali diperbarui.

Tabel 4.8 Rancangan tabel *Hazards*

No.	Atribut	Deskripsi
1	**id**	Atribut identitas tabel *Hazards*.
2	**hazard**	Atribut jenis bahaya.
3	**color_code**	Atribut kode warna untuk setiap jenis bahaya.
4	**created_at**	Atribut waktu data telah dibuat.
5	**updated_at**	Atribut waktu data terakhir kali diperbarui.

Tabel 4.9 Rancangan tabel *Matrix*

No.	Atribut	Deskripsi
1	**id**	Atribut identitas tabel *Matrix*.
2	**probability_level**	Atribut informasi tingkatan probabilitas.
3	**impact_level**	Atribut informasi tingkatan impact.
4	**risk_level**	Atribut informasi tingkatan nilai resiko.
5	**level_name**	Atribut nama tingkat dari matrix.
6	**color_code**	Atribut informasi data untuk pengkodean warna.

7	**column**	Atribut informasi letak kolom pada matrix.
8	**row**	Atribut informasi letak baris pada matrix.
9	**properties**	Atribut informasi tambahan.
10	**created_at**	Atribut waktu data telah dibuat.
11	**updated_at**	Atribut waktu data terakhir kali diperbarui.

Tabel 4.10 Rancangan tabel *Probabilities*

No.	Atribut	Deskripsi
1	**id**	Atribut identitas tabel *Probabilites.*
2	**level**	Atribut tingkat probabilitas terhadap suatu potensi bahaya.
3	**probability**	Atribut detail dari masing-masing tingkat probabilitas terhadap suatu potensi bahaya.
4	**Created_by**	Atribut informasi mengenai user yang membuat data.
5	**updated_by**	Atribut informasi mengenai user terakhir yang memperbarui data.
6	**created_at**	Atribut waktu data telah dibuat.
7	**updated_at**	Atribut waktu data terakhir kali diperbarui.

Tabel 4.11 Rancangan tabel *Impacts*

No.	Atribut	Deskripsi

1	**id**	Atribut identitas tabel *Impacts*.
2	**level**	Atribut tingkat impact terhadap suatu potensi bahaya.
3	**impact**	Atribut detail dari masing-masing tingkat dampak terhadap suatu potensi bahaya.
4	**Created_by**	Atribut informasi mengenai user yang membuat data.
5	**updated_by**	Atribut informasi mengenai user terakhir yang memperbarui data.
6	**created_at**	Atribut waktu data telah dibuat.
7	**updated_at**	Atribut waktu data terakhir kali diperbarui.

Tabel 4.12 Rancangan tabel *Areas*

No.	Atribut	Deskripsi
1	**id**	Atribut identitas tabel *Areas*.
2	**area**	Atribut nama area.
3	**floor**	Atribut lantai gedung.
4	**head_id**	Atribut informasi user yang bertanggung jawab pada area.
5	**department_id**	Atribut informasi departemen yang bertanggung jawab pada area.
6	**created_by**	Atribut informasi user yang membuat data.
7	**updated_by**	Atribut informasi mengenai user terakhir yang

		memperbarui data.
8	**created_at**	Atribut waktu data telah dibuat.
9	**updated_at**	Atribut waktu data terakhir kali diperbarui.

Tabel 4.13 Rancangan tabel *Area_Details*

No.	Atribut	Deskripsi
1	**id**	Atribut identitas tabel *Area_details*.
2	**area_detail**	Atribut nama area detail.
3	**floor**	Atribut lantai gedung.
4	**area_id**	Atribut informasi data parent area.
5	**head_id**	Atribut informasi user yang bertanggung jawab pada area.
6	**department_id**	Atribut informasi departemen yang bertanggung jawab pada area.
7	**created_by**	Atribut informasi user yang membuat data.
8	**updated_by**	Atribut informasi mengenai user terakhir yang memperbarui data.
9	**created_at**	Atribut waktu data telah dibuat.
10	**updated_at**	Atribut waktu data terakhir kali diperbarui.

Tabel 4.14 Rancangan tabel *Status*

No.	Atribut	Deskripsi
1	**id**	Atribut identitas tabel *Status.*
2	**status**	Atribut informasi status.
3	**created_at**	Atribut waktu data telah dibuat.
4	**updated_at**	Atribut waktu data terakhir kali diperbarui.

Tabel 4.15 Rancangan tabel *Feedback*

No.	Atribut	Deskripsi
1	**id**	Atribut identitas tabel *Feedback.*
2	**document_id**	Atribut nomor dokumen feedback.
3	**activity**	Atribut jenis data.
4	**user_id**	Atribut informasi mengenai user yang membuat data.
5	**department_id**	Atribut informasi departemen yang membuat data.
6	**subject**	Atribut subyek feedback.
7	**message**	Atribut pesan feedback.
8	**remark**	Atribut keterangan feedback.
9	**status_id**	Atribut status feedback.
10	**answered**	Atribut informasi mengenai status feedback apakah sudah dijawab atau belum.
11	**feedback_picture1**	Atribut gambar feedback 1.
12	**feedback_picture2**	Atribut gambar feedback 2.

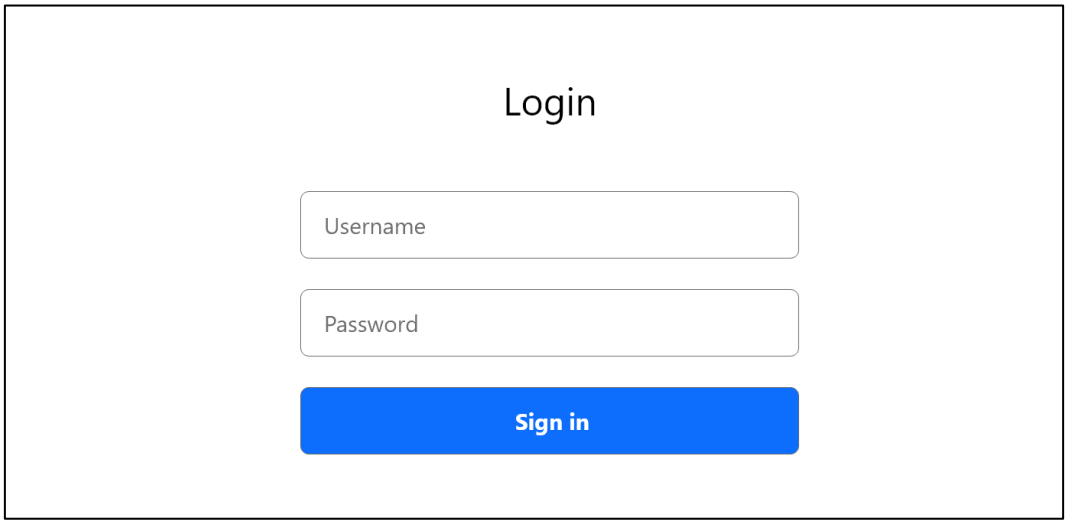
13	**feedback_picture3**	Atribut gambar feedback 3.
14	**created_at**	Atribut waktu data telah dibuat.
15	**updated_at**	Atribut waktu data terakhir kali diperbarui.

4.3 Perancangan Antarmuka

Perancangan antarmuka atau *User Interface Design* merupakan salah satu proses yang tidak kalah penting saat merancang sebuah sistem. Antarmuka berfungsi untuk menjembatani antara pengguna dengan sistem. Perancangan antarmuka bertujuan untuk membuat sistem tersebut lebih mudah digunakan oleh pengguna atau biasa disebut dengan istilah *user friendly*. Selain itu, beberapa pengaruh user interface terhadap kemudahan penggunaan sistem seperti tampilan grafik lebih berguna karena dapat memberikan trend secara visual [13].

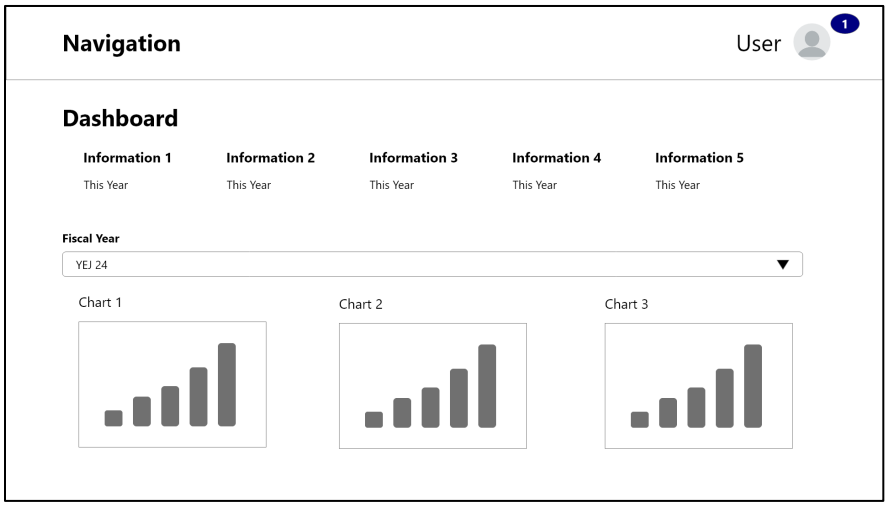
4.3.1 Halaman Login

Halaman *Login* adalah halaman awal yang ditampilkan ketika user belum melakukan login. Di halaman ini user harus menginput data username dan password yang akan divalidasi oleh sistem sebelum masuk ke halaman *Dashboard*.



4.3.2 Halaman Dashboard

Halaman *Dashboard* adalah halaman awal yang ditampilkan setelah user melakukan *Login*. Di halaman ini terdapat rangkuman data statistik terkait dengan inspeksi yang sedang berlangsung maupun inspeksi yang telah selesai pada periode waktu tertentu.



Gambar 4.3 Rancangan antarmuka Dashboard

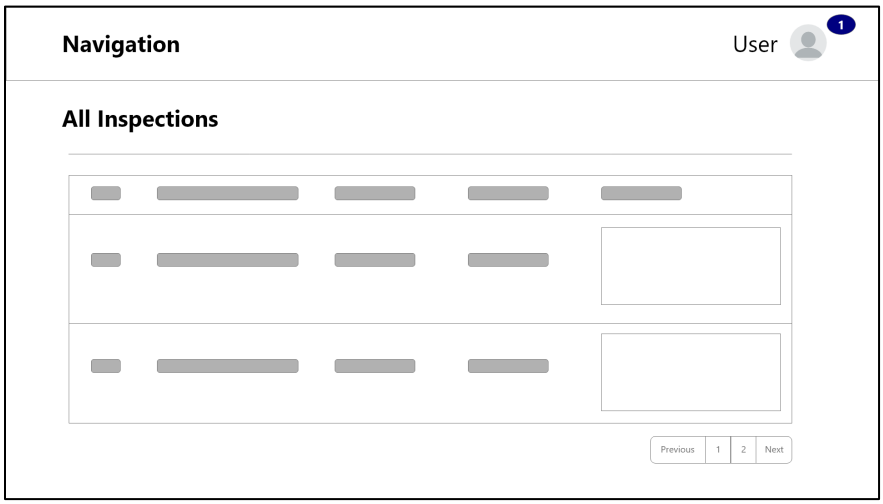
4.3.3 Halaman Notifications

Halaman *Notifications* adalah halaman yang menampilkan daftar notifikasi berupa data inspeksi yang ditujukan kepada user.

Gambar 4.4 Rancangan antarmuka Notifications

4.3.4 Halaman All Inspections

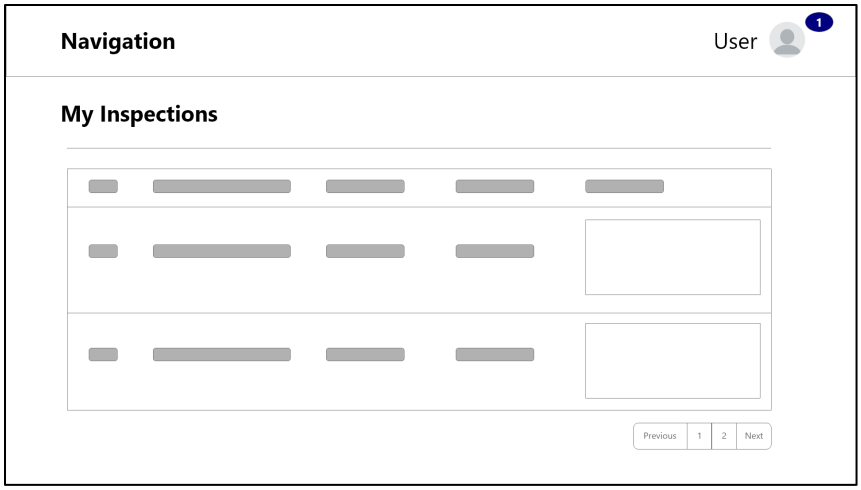
Halaman All Inspections adalah halaman yang menampilkan seluruh daftar inspeksi yang telah terdata kedalam sistem.



Gambar 4.5 Rancangan antarmuka All Inspections

4.3.5 Halaman My Inspections

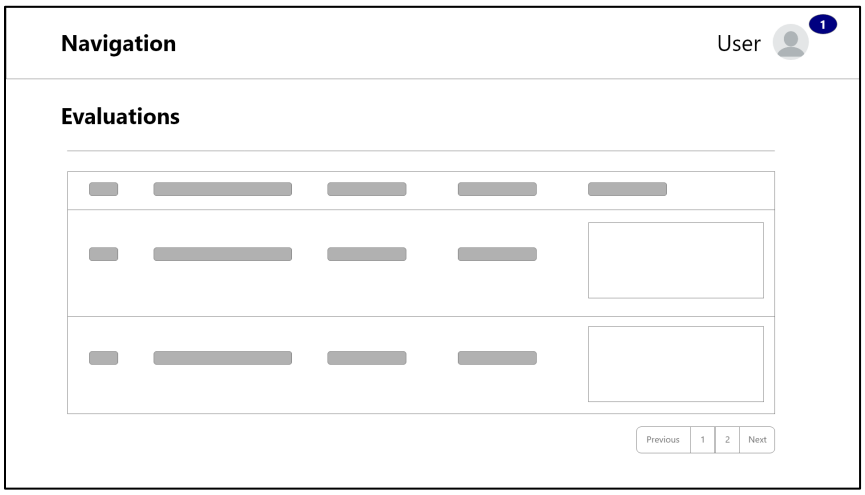
Halaman *My Inspections* adalah halaman yang menampilkan daftar inspeksi yang pernah dilaporkan oleh pengguna.



Gambar 4.6 Rancangan antarmuka My Inspections

4.3.6 Halaman Evaluations

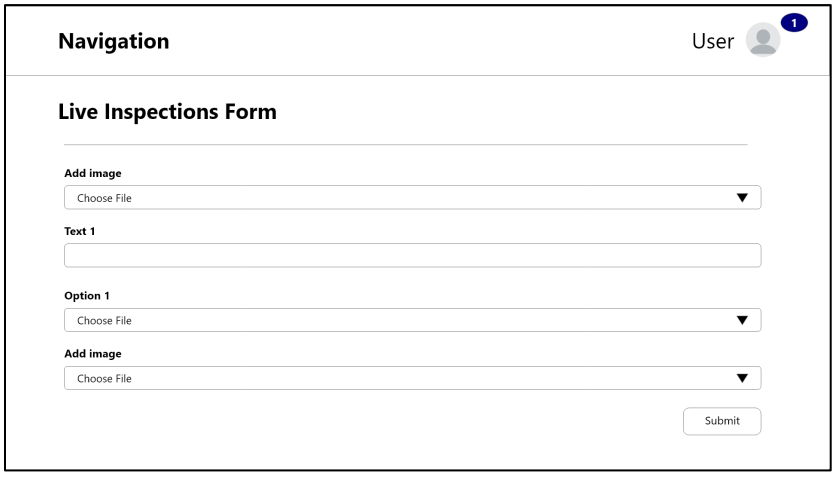
Halaman Evaluations adalah halaman yang menampilkan daftar inspeksi berdasarkan data departemen dari data pengguna.



Gambar 4.7 Rancangan antarmuka Evaluations

4.3.7 Halaman Live Inspections

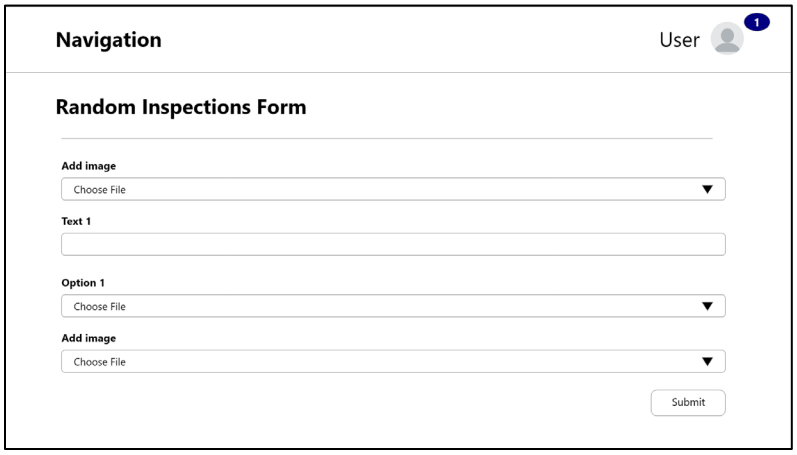
Halaman *Live Inspection* adalah halaman yang menampilkan form input untuk pengguna saat ingin melaporkan inspeksi.



Gambar 4.8 Rancangan antarmuka Live Inspections

4.3.8 Halaman Random Inspection

Halaman Random Inspection adalah halaman yang menampilkan form input untuk melaporkan inspeksi, tetapi pada halaman Random Inspection tidak memiliki opsi “pilih departemen” karena sistem sudah melakukan generasi secara acak untuk data target departemen pada data user.



Gambar 4.9 Rancangan antarmuka Random Inspections

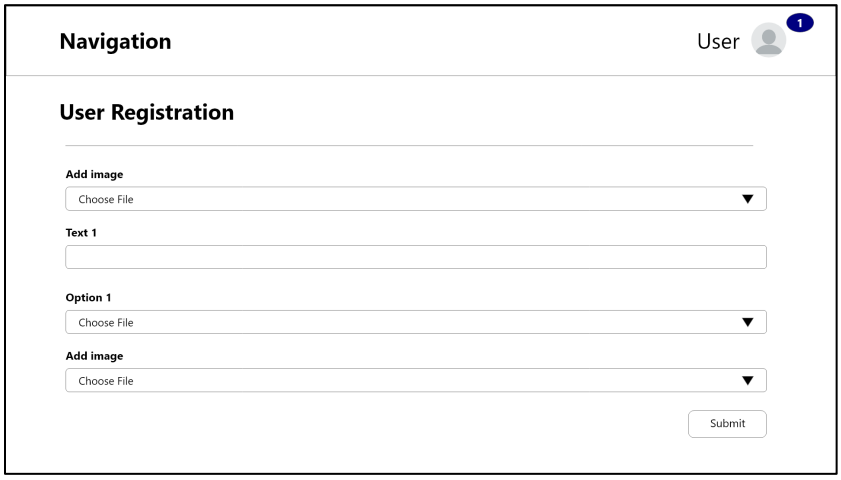
4.3.9 Halaman Inspection Details

Halaman *Inspection Details* digunakan untuk menampilkan rincian data inspeksi secara detail. Di halaman ini user yang berwenang dapat melakukan proses untuk inspeksi sesuai dengan status, departemen, dan PIC yang berkaitan.

Gambar 4.10 Rancangan antarmuka Inspection Details

4.3.10 Halaman User Registration

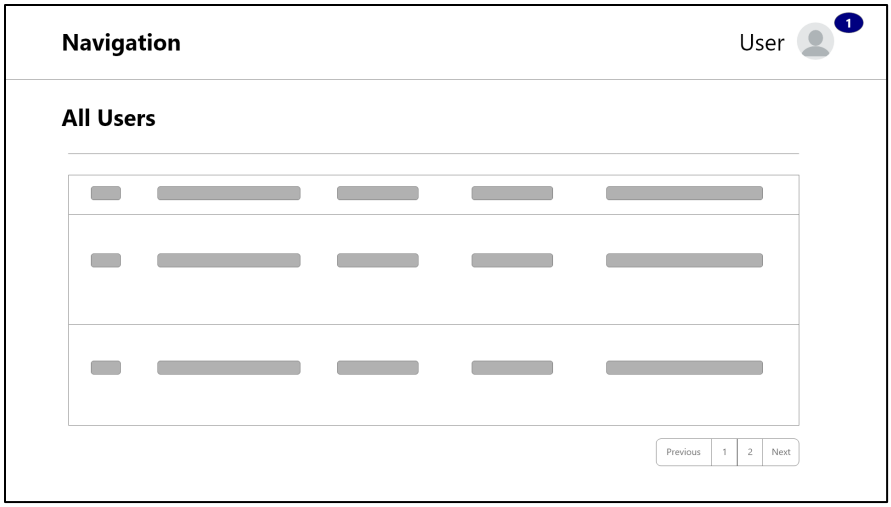
Halaman User Registration digunakan untuk menampilkan form untuk melakukan registrasi user. Halaman User Registration hanya dapat diakses oleh user dengan level Administrator.



Gambar 4.11 Rancangan antarmuka User Registration

4.3.11 Halaman All Users

Halaman *All Users* adalah halaman yang menampilkan seluruh daftar user yang telah terdata kedalam sistem.

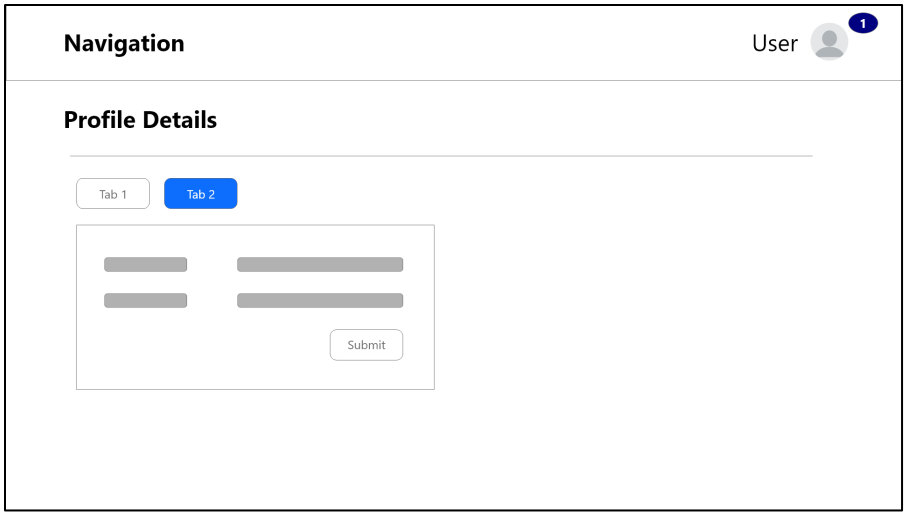


Gambar 4.12 Rancangan antarmuka All Users

4.3.12 Halaman Profiles

Halaman *Profiles* digunakan untuk menampilkan rincian data user secara detail. Di halaman ini user dengan level administrator atau user itu sendiri dapat melakukan perubahan password terhadap akun.

Gambar 4.13 Rancangan antarmuka Profile Details (tab 1)

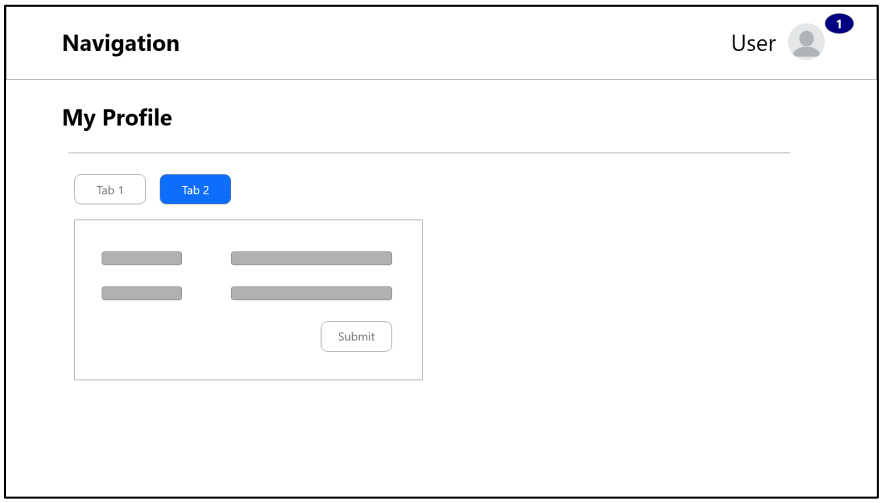


Gambar 4.14 Rancangan antarmuka Profile Details (tab 2)

4.3.13 Halaman My Profile

Halaman *My Profiles* digunakan untuk menampilkan rincian data user itu sendiri secara detail. Di halaman ini user dapat melakukan perubahan password terhadap akun.

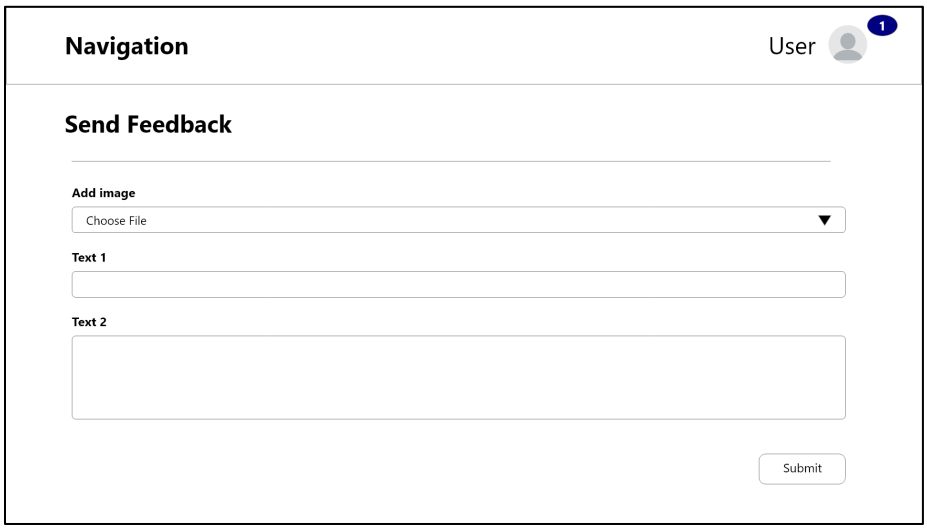
Gambar 4.15 Rancangan antarmuka My Profile (tab 1)



Gambar 4.16 Rancangan antarmuka My Profile (tab 2)

4.3.14 Halaman Send Feedback

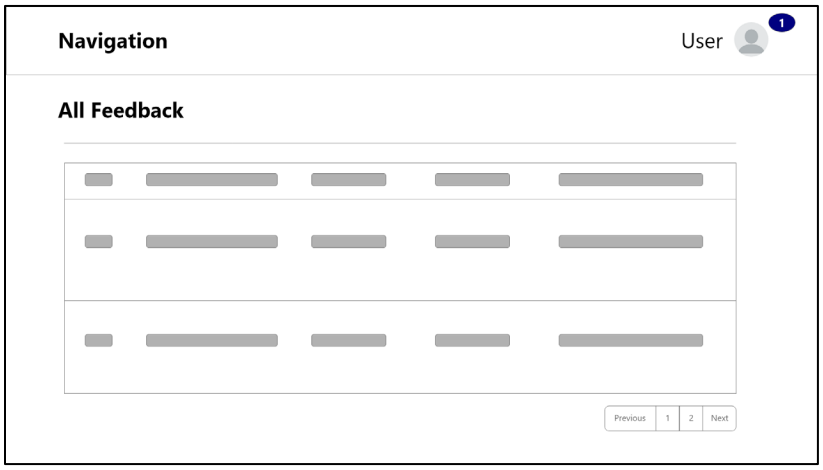
Halaman *Send Feedback* adalah halaman yang menampilkan form input untuk pengguna saat ingin mengirim feedback.



Gambar 4.17 Rancangan antarmuka Send Feedback

4.3.15 Halaman All Feedback

Halaman *All Feedback* adalah halaman yang menampilkan seluruh daftar feedback yang telah terdata kedalam sistem.



Gambar 4.18 Rancangan antarmuka All Feedback

4.3.16 Halaman Feedback Details

Halaman *Feedback Details* digunakan untuk menampilkan rincian data feedback secara detail.

BAB V

IMPLEMENTASI SISTEM

5.1 Implementasi Basis Data

Bagian ini menunjukan hasil implementasi dari rancangan basis data yang sudah di deskripsikan sebelumnya. Implementasi basis data akan dipaparkan dalam bentuk kode program dengan bahasa pemrograman PHP menggunakan *framework* Laravel.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('nik')->nullable();
            $table->string('username') ;
            $table->string('email')->unique();
            $table->string('first_name')->nullable();
            $table->string('last_name')->nullable();
            $table->string('initial_name')->nullable();
            $table->foreignId('gender_id')->nullable()->constrained();
            $table->foreignId('department_id')->nullable()->constrained();
            $table->foreignId('position_id')->nullable()->constrained();
            $table->foreignId('level_id')->nullable()->constrained();
            $table->unsignedBigInteger('inspection_target_id')->default('404');
            $table->string('profile_picture')->nullable();
            $table->date('join_date')->nullable();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
};
```

5.1.1 Implementasi Entitas *Users*

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('genders', function (Blueprint $table) {
            $table->id();
            $table->string('gender')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('genders');
    }
};
```

5.1.2 Implementasi Entitas *Genders*

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('departments', function (Blueprint $table) {
            $table->id();
            $table->string('department');
            $table->unsignedBigInteger('head_id')->nullable();
            $table->json('properties')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('departments');
    }
};
```

5.1.3 Implementasi Entitas *Departments*

5.1.4 Implementasi Entitas Positions

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('positions', function (Blueprint $table) {
            $table->id();
            $table->string('position');
            $table->foreignId('department_id')->nullable()->constrained();
            $table->foreignId('level_id')->nullable()->constrained();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('positions');
    }
};
```

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('levels', function (Blueprint $table) {
            $table->id();
            $table->string('level');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('levels');
    }
};
```

5.1.5 Implementasi Entitas Levels

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('inspections', function (Blueprint $table) {
            $table->id();
            $table->string('document_id')->nullable();
            $table->string('inspection_type')->nullable();
            $table->string('initial_picture1')->nullable();
            $table->string('initial_picture2')->nullable();
            $table->string('initial_picture3')->nullable();
            $table->string('evaluation_picture')->nullable();
            $table->string('postpone_picture')->nullable();
            $table->string('finished_picture')->nullable();
            $table->string('subject')->nullable();
            $table->string('message')->nullable();
            $table->foreignId('area_id')->nullable()->constrained();
            $table->foreignId('area_detail_id')->nullable()->constrained();
            $table->string('remark_open')->nullable();
            $table->string('remark_in_process')->nullable();
            $table->string('remark_finished')->nullable();
            $table->string('remark_deleted')->nullable();
            $table->foreignId('department_id')->nullable()->constrained();
            $table->unsignedBigInteger('pic')->nullable();
            $table->foreign('pic')->references('id')->on('users');
            $table->foreignId('hazard_id')->nullable()->constrained();
            $table->foreignId('warning_id')->nullable()->constrained();
            $table->unsignedBigInteger('matrix_id')->nullable();
            $table->foreign('matrix_id')->references('id')->on('matrix');
```

5.1.6 Implementasi Entitas *Inspections*

```
            $table->timestamp('evaluation_target')->nullable();
            $table->date('evaluation_date')->nullable();
            $table->unsignedBigInteger('status_id')->nullable();
            $table->foreign('status_id')->references('id')->on('status');
            $table->boolean('readed')->nullable();
            $table->unsignedBigInteger('created_by')->nullable();
            $table->foreign('created_by')->references('id')->on('users');
            $table->unsignedBigInteger('updated_by')->nullable();
            $table->foreign('updated_by')->references('id')->on('users');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('inspections');
    }
};

```

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('timelines', function (Blueprint $table) {
            $table->id();
            $table->string('document_id')->nullable();
            $table->string('activity')->nullable();
            $table->string('remark')->nullable();
            $table->unsignedBigInteger('matrix_id')->nullable();
            $table->foreign('matrix_id')->references('id')->on('matrix');
            $table->unsignedBigInteger('status_id')->nullable();
            $table->foreign('status_id')->references('id')->on('status');
            $table->unsignedBigInteger('created_by')->nullable();
            $table->unsignedBigInteger('updated_by')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('timelines');
    }
};
```

5.1.7 Implementasi Entitas *Timelines*

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('hazards', function (Blueprint $table) {
            $table->id();
            $table->string('hazard');
            $table->string('color_code')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('hazards');
    }
};
```

5.1.8 Impelementasi Entitas *Hazards*

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('matrix', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('probability_level')->nullable();
            $table->foreign('probability_level')->references('id')->on('probabilities');
            $table->unsignedBigInteger('impact_level')->nullable();
            $table->foreign('impact_level')->references('id')->on('impacts');
            $table->integer('risk_level')->nullable();
            $table->string('level_name')->nullable();
            $table->string('color_code')->nullable();
            $table->integer('column')->nullable();
            $table->integer('row')->nullable();
            $table->json('properties')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('matrix');
    }
};
```

5.1.9 Implementasi Entitas *Matrix*

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('probabilities', function (Blueprint $table) {
            $table->id();
            $table->string('level')->nullable();
            $table->string('probability')->nullable();
            $table->unsignedBigInteger('created_by')->nullable();
            $table->unsignedBigInteger('updated_by')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('probabilities');
    }
};
```

5.1.10 Implementasi Entitas *Probabilities*

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('impacts', function (Blueprint $table) {
            $table->id();
            $table->string('level')->nullable();
            $table->string('impact')->nullable();
            $table->unsignedBigInteger('created_by')->nullable();
            $table->unsignedBigInteger('updated_by')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('impacts');
    }
};
```

5.1.11 Implementasi Entitas *Impacts*

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('areas', function (Blueprint $table) {
            $table->id();
            $table->string('area');
            $table->unsignedBigInteger('floor')->nullable();
            $table->unsignedBigInteger('head_id')->nullable();
            $table->unsignedBigInteger('department_id')->nullable();
            $table->unsignedBigInteger('created_by')->nullable();
            $table->unsignedBigInteger('updated_by')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('areas');
    }
};
```

5.1.12 Implementasi Entitas *Areas*

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('area_details', function (Blueprint $table) {
            $table->id();
            $table->string('area_detail');
            $table->integer('floor')->nullable();
            $table->foreignId('area_id')->nullable()->constrained();
            $table->unsignedBigInteger('head_id')->nullable();
            $table->unsignedBigInteger('department_id')->nullable();
            $table->unsignedBigInteger('created_by')->nullable();
            $table->unsignedBigInteger('updated_by')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('area_details');
    }
};
```

5.1.13 Implementasi Entitas *Area_Details*

s

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('status', function (Blueprint $table) {
            $table->id();
            $table->string('status');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('status');
    }
};
```

5.1.14 Implementasi Entitas *Status*

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('feedback', function (Blueprint $table) {
            $table->id();
            $table->string('document_id')->nullable();
            $table->string('activity')->nullable();
            $table->foreignId('user_id')->nullable()->constrained();
            $table->foreignId('department_id')->nullable()->constrained();
            $table->string('subject')->nullable();
            $table->string('message')->nullable();
            $table->string('remark')->nullable();
            $table->unsignedBigInteger('status_id')->nullable();
            $table->foreign('status_id')->references('id')->on('status');
            $table->boolean('answered');
            $table->string('feedback_picture1')->nullable();
            $table->string('feedback_picture2')->nullable();
            $table->string('feedback_picture3')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('feedback');
    }
};
```

5.1.15 Implementasi Entitas Feedback

5.2 Implementasi Safety Management System

Bagian ini membahas hasil implementasi dari perancangan yang telah di deskripsikan sebelumnya dalam bentuk kode program.

5.2.1 Implementasi *Login*

Kode program ini merupakan fungsi authenticate() milik controller LoginController. Kode program ini akan berjalan saat user menekan tombol “Sign in” pada halaman *Login*. Kode program untuk fungsi ini dapat dilihat pada Gambar 5.16 dan dijelaskan pada Tabel 5.1.

```php
// LOGIN PROCESS
public function authenticate(Request $request)
{
    $credentials = $request->validate([
        'username' => ['required'],
        'password' => ['required'],
    ]);

    if (Auth::attempt($credentials)) {
        $request->session()->regenerate();
        return redirect()->intended('/dashboard');
    }

    return back()->withErrors([
        'username' => 'username or password is incorrect!',
    ])->onlyInput('username');
}
```

Tabel 5.1 Penjelasan kode program fungsi *Authenticate*

Baris	**Penjelasan**
24 - 32	Kode ini bertugas untuk melakukan validasi terhadap input username dan password harus terisi terlebih dahulu sebelum proses validasi login akun dengan data yang sudah tersimpan di database. Jika login berhasil, maka tampilan akan dialihkan ke halaman dashboard.
34 - 36	Kode ini bertugas untuk mengalihkan tampilan ke halaman login kembali dan menyimpan serta menampilkan pesan error dan mengembalikan input sebelumnya pada bagian username.

5.2.2 Implementasi *Dashboard*

Kode program ini merupakan fungsi index() milik controller DashboardController. Kode program ini akan berjalan saat user berada di halaman *Dashboard*. Kode program untuk fungsi ini dapat dilihat pada Gambar 5.17 dan d

```
26  class DashboardController extends Controller
27  {
28      //
29      public function index()
30      {
31          $active = 'dashboard';
32          $inspections = count(Inspection::all());
33          $pendingInspections = count(Inspection::where('status_id', 1)->get());
34          $openInspections = count(Inspection::where('status_id', 2)->get());
35          $inProgressInspections = count(Inspection::where('status_id', 3)->get());
36          $finishedInspections = count(Inspection::where('status_id', 4)->get());
37          $date = Carbon::now();
38          $employees = count(User::where('nik', '!=', NULL)->get());
39
40          return view('home', compact(
41              'active',
42              'inspections',
43              'pendingInspections',
44              'openInspections',
45              'inProgressInspections',
46              'finishedInspections',
47              'employees',
48              'date'
49          ));
50      }
51
52  }
```

da Tabel 5.2.

Tabel 5.2 Penjelasan kode program fungsi *index*

Baris	**Penjelasan**
31 - 38	Kode ini bertugas untuk mengambil seluruh data yang diperlukan untuk ditampilkan pada halaman *Dashboard*.
40 - 50	Kode ini bertugas untuk menyimpan variabel yang telah dibuat

	dan membuka view pada file "*home*".

5.2.3 Implementasi *Notifications*

Kode program ini merupakan fungsi notifications() milik controller InspectionController. Kode program ini akan berjalan saat user berada di halaman *Notifications*. Kode program untuk fungsi ini dapat dilihat pada Gambar 5.18 dan

```
24    // NOTIFICATIONS
25    public function notifications()
26    {
27        $active = 'notifications';
28        // SHOW DATA BASED ON USER
29        if (Auth::user()->level_id == 5) {
30            $data = Inspection::with('user', 'status', 'department', 'hazard')
31            ->where('status_id', 1)
32            ->orWhere('status_id', 4)->get();
33        }
34        elseif (Auth::user()->level_id == 99) {
35            $data = Inspection::with('user', 'status', 'department', 'hazard')->get();
36        }
37        else {
38            $data = Inspection::with('user', 'status', 'department', 'hazard')
39            ->where('department_id', Auth::user()->department_id)
40            ->orWhere('created_by', Auth::id())->get();
41        }
42        $date = Carbon::now();
43
44        return view('inspections.showInspections', compact('active', 'data', 'date'));
45    }
```

dijelaskan pada Tabel 5.3.

Tabel 5.3 Penjelasan kode program fungsi *notifications*

Baris	**Penjelasan**
27 - 42	Kode ini bertugas untuk mengambil data *inspections* yang diperlukan sesuai dengan kondisi berikut: Jika level user adalah

	safety team, maka variabel *$data* akan menyimpan data inspections dengan status "*Pending*" dan "*Finished*". Jika level user adalah *administrator*, maka variable *$data* akan meyimpan seluruh data inspections. Jika level user bukan *safety team* atau *administrator*, maka variabel *$data* menyimpan data *inspections* berdasarkan data departemen yang berkaitan dengan user dan data *inspections* yang telah dibuat oleh user itu sendiri. Proses pengambilan data menggunakan *Eager Loading* sehingga proses pegambilan data akan terhindar dari masalah query n+1, total query yang digunakan juga akan semakin sedikit dibandingkan dengan *Lazy Loading*.
40 - 50	Kode ini bertugas untuk menyimpan variabel yang telah dibuat dan membuka view pada file "*showInspections*".

5.2.4 Implementasi get *All Inspections*

Kode program ini merupakan fungsi inspectionsAll() milik controller InspectionController. Kode program ini akan berjalan saat user berada di halaman *All Inspections*. Kode program untuk fungsi ini dapat dilihat pada Gambar 5.19 dan

```
43      // ALL INSPECTIONS
44      public function inspectionsAll()
45      {
46          $active = 'all inspections';
47          $data = Inspection::with('user', 'status', 'department', 'hazard')->get();
48          $date = Carbon::now();
49
50          return view('inspections.showInspections', compact('active', 'data', 'date'));
51      }
```

dijelaskan pada Tabel 5.4.

Tabel 5.4 Penjelasan kode program fungsi *inspectionsAll*

Baris	**Penjelasan**
46 - 48	Kode ini bertugas untuk mengambil seluruh data *inspection*s yang tersimpan pada database. Proses pengambilan data menggunakan *Eager Loading*
50	Kode ini bertugas untuk menyimpan variabel yang telah dibuat dan membuka view pada file "*showInspections*".

5.2.5 Implementasi *get My Inspections*

Kode program ini merupakan fungsi showMyInspections() milik controller InspectionController. Kode program ini akan berjalan saat user berada di halaman *My Inspections*. Kode program untuk fungsi ini dapat dilihat pada

```
53      // MY INSPECTIONS
54      public function myInspections()
55      {
56          $active = 'my inspections';
57          $data = Inspection::with('user', 'status', 'department', 'hazard')
58                  ->where('created_by', Auth::user()->id)
59                  ->orWhere('department_id', Auth::user()->department_id)->get();
60          $date = Carbon::now();
61
62          return view('inspections.showInspections', compact('active', 'data', 'date'));
63      }
```

Gambar 5.20 dan dijelaskan pada Tabel 5.5.

Tabel 5.5 Penjelasan kode program fungsi *myInspections*

Baris	**Penjelasan**
46 - 48	Kode ini bertugas untuk mengambil data *inspection*s yang telah dibuat oleh user yang tersimpan pada database. Proses pengambilan data menggunakan *Eager Loading*.
50	Kode ini bertugas untuk menyimpan variabel yang telah dibuat dan membuka view pada file "*showInspections*".

5.2.6 Implementasi *get Evaluations*

Kode program ini merupakan fungsi showEvaluations() milik controller InspectionController. Kode program ini akan berjalan saat user berada di halaman *Evaluations*. Kode program untuk fungsi ini dapat dilihat pada Gambar 5.21 dan

```
65      // EVALUATIONS
66      public function showEvaluations()
67      {
68          $active = 'evaluations';
69          $data = Inspection::with('user', 'status', 'department', 'hazard')
70                  ->where('department_id', Auth::user()->department_id)->get();
71          $date = Carbon::now();
72
73          return view('inspections.showInspections', compact('active', 'data', 'date'));
74      }
```

dijelaskan pada Tabel 5.6.

Tabel 5.6 Penjelasan kode program fungsi *showEvaluations*

Baris	**Penjelasan**
68 - 71	Kode ini bertugas untuk mengambil data *inspection*s yang sedang dikerjakan oleh user sebagai pic dari inspeksi tersebut yang tersimpan pada database. Proses pengambilan data menggunakan *Eager Loading*.
50	Kode ini bertugas untuk menyimpan variabel yang telah dibuat dan membuka view pada file "*showInspections*".

5.1.1 Implementasi *add Inspection*

Kode program ini merupakan fungsi storeInspection() milik controller InspectionController. Kode program ini akan berjalan saat user menekan tombol "*Submit*" pada halaman *Live Inspections*. Kode program untuk fungsi ini dapat dilihat pada Gambar 5.22 dan Gambar 5.23 dan dijelaskan pada Tabel 5.7.

```php
public function storeInspection(Request $request){
    $lastRow = Inspection::all()->last();
    $check = Inspection::count();

    if ($check == 0)
    {
        $serial = SerialNumber::where('active', 1)->first();
        $number = $serial->serial_number;

    }

    else if (isset($check))
    {
        $order = 1;
        $number = $lastRow->document_id+$order;
    }

    if ($request->file('initial_picture1') && $request->file('initial_picture2') && $request->file('initial_picture3'))
    {
        $file1 = $request->file('initial_picture1');
        $image1 = date('YmdHi').'-'.$file1->getClientOriginalName();

        $file2 = $request->file('initial_picture2');
        $image2 = date('YmdHi').'-'.$file2->getClientOriginalName();

        $file3 = $request->file('initial_picture3');
        $image3 = date('YmdHi').'-'.$file3->getClientOriginalName();
    }

    else
    {
        $image1 = NULL;
        $image2 = NULL;
        $image3 = NULL;
    }
```

Gambar 5.23 Kode program fungsi *storeInspection*

```
147         Inspection::create([
148             'document_id' => $number,
149             'inspection_type' => 'Live Inspections',
150             'initial_picture1' => $image1,
151             'initial_picture2' => $image2,
152             'initial_picture3' => $image3,
153             'subject' => $request->subject,
154             'message' => $request->message,
155             'area_id' => $request->area_id,
156             'department_id' => $request->department_id,
157             'hazard_id' => $request->hazard_id,
158             'status_id' => 1,
159             'created_by' => Auth::id(),
160             'updated_by' => Auth::id()
161         ]);
162 
163         Timeline::create([
164             'document_id' => $number,
165             'activity' => 'Live Inspections',
166             'status' => '1',
167             'created_by' => Auth::id(),
168             'updated_by' => Auth::id()
169         ]);
170 
171         if ($request->file('initial_picture1') && $request->file('initial_picture2') && $request->file('initial_picture3'))
172         {
173             $file1->move('public/image', $image1);
174             $file2->move('public/image', $image2);
175             $file3->move('public/image', $image3);
176         }
177 
178         return redirect('/myInspections');
179 
180     }
```

Tabel 5.7 Penjelasan kode program fungsi *storeInspection*

Baris	**Penjelasan**
76 - 90	Kode ini bertugas untuk menentukan nomor dokumen yang digunakan untuk data inspeksi yang akan ditambahkan. kode ini akan melakukan validasi apakah sudah terdapat data inspeksi pada database, jika belum maka nomor yang digunakan adalah nomor dokumen sesuai dengan data pada tabel serial number dengan status aktif. Jika data inspeksi sebelumnya sudah ada pada database, maka nomor dokumen selanjutnya akan ditambahkan 1 sesuai dari nilai variabel $order.
92 - 109	Kode ini bertugas melakukan validasi apakah user telah menginput

	sebanyak 3 gambar yang disimpan ke variabel $request. Jika benar, maka file gambar tersebut akan diberi nama yang berisi tanggal dan nama file melalui function *getClientOriginalName()*, lalu nama file tersebut akan disimpan ke variabel *$image1*, *$image2*, dan *$image3*. Jika kondisi tersebut tidak terpenuhi, maka variabel *$image1*, *$image2*, dan *$image3* akan memiliki nilai "*NULL*".
147 - 161	Kode ini bertugas untuk melakukan proses insert data terhadap model *Inspection* menggunakan method eloquent *create*.
163 - 169	Kode ini bertugas untuk melakukan proses insert data terhadap model *Timeline* menggunakan method eloquent *create*.
173 - 176	Kode ini bertugas untuk memindahkan file gambar yang telah diinput ke penyimpanan public pada server. Kode ini hanya akan berjalan ketika user telah menginput gambar.

5.1.2 Implementasi *Inspection Details*

Kode program ini merupakan fungsi processInspections() milik controller InspectionController. Kode program ini akan berjalan saat user berada di halaman *Inspection Details*. Kode program untuk fungsi ini dapat dilihat pada Gambar 5.24 dan Gambar 5.25 dan dijelaskan pada Tabel 5.8.

```
270     // INSPECTION DETAILS
271     public function processInspections(Inspection $document_id)
272     {
312         $matrix = Matrix::all();
313         for ($i=1;$i<=5;$i++) {
314             ${"arr$i"} = Matrix::where('row', $i)->get();
315         }
316
317         return view('inspections.process', compact(
318             'active',
319             'areas',
320             'areaDetails',
321             'data',
322             'date',
323             'time',
324             'hazards',
325             'timeline',
326             'probabilities',
327             'impacts',
328             'action_target',
329             'matrix',
330             'departments',
331             'allAreas',
332             'matrix',
333             'arr1',
334             'arr2',
335             'arr3',
336             'arr4',
337             'arr5'
338         ));
339     }
305         {
306             $daysToAdd = 3;
307             $date = $data->created_at;
308             $date = $date->addDays($daysToAdd)->format('Y-m-d');
309             $time = '08:00';
310         }
```

Tabel 5.8 Penjelasan kode program fungsi *processInspections*

Baris	**Penjelasan**
279 – 284	Kode ini bertugas untuk menentukan opsi yang akan muncul pada bagian *area* di halaman *Inspection Details*. Kode ini memiliki kondisi sebagai berikut: Jika data *inspections* sudah memiliki data *area_id*, maka opsi yang ditampilkan adalah *area_detail* berdasarkan area sesuai dengan data *inspections* tersebut saja. Jika tidak maka, opsi *area* yang ditampilkan adalah seluruh *area* dan

	area_detail secara keseluruhan.
290 - 295	Kode ini bertugas untuk menentukan route yang digunakan untuk form pada halaman Inspection Details yang disimpan ke variabel $action_target. Kode ini memiliki kondisi sebagai berikut: Jika data *inspections* yang diakses memiliki status "Finished" maka route akan mengarah ke "inspections/close". Jika tidak, maka route akan mengarah ke "inspections/edit".

5.1.3 Implementasi *Update, Approve, Proceed Inspections*

Kode program ini merupakan fungsi actionInspections() milik controller InspectionController. Kode program ini akan berjalan saat user menekan tombol *Update* atau *Submit* di halaman *Inspection Details*. Kode program untuk fungsi ini dapat dilihat pada Gambar 5.26 sampai dengan Gambar 5.29 dan dijelaskan pada Tabel 5.9.

```
341     // INSPECTION APPROVAL/ PROCEED/ UPDATE
342     public function actionInspections(Request $request, $document_id)
343     {
344         $data = Inspection::where('document_id', $document_id)->first();
345         $areaDetail = AreaDetail::where('id', $request->area_detail)->first();
346 
347         // Area & Area Details
348         if ($request->area_detail == NULL) {
349             $area = $data->area_id;
350             $area_detail = $data->area_detail_id;
351         }
352         else {
353             $area = $areaDetail->area_id;
354             $area_detail = $request->area_detail;
355         }
356 
357         // Condition by Status
358         if ($data->status_id == 1) {
359             $status = 2;
360             $picture = 'initial_picture1';
361             $image = $data->initial_picture1;
362             $remark = 'remark_open';
363         }
364         elseif ($data->status_id == 2) {
365             $status = 3;
366             $picture = 'evaluation_picture';
367             $remark = 'remark_in_process';
368         }
369         elseif ($data->status_id == 3) {
370             $status = 4;
371             $picture = 'finished_picture';
372             $remark = 'remark_finished';
373         }
```

```php
// Matrix
if ($request->probability_level == NULL || $request->impact_level == NULL) {
    $matrix_id = $data->matrix_id;
}
else {
    $matrixes_id = Matrix::where([
                    ['probability_level', $request->probability_level],
                    ['impact_level', $request->impact_level]])
                    ->first();
    $matrix_id = $matrixes_id->id;
}

// Responsible Department
if ($request->department == NULL) {
    $department = $data->department_id;
}
else {
    $department = $request->department;
}

// Post picture validation
if ($request->file('evaluation_picture'))
{
    $file = $request->file('evaluation_picture');
    $image = date('YmdHi').'-'.$file->getClientOriginalName();
    $file-> move('public/Image', $image);
}
elseif ($request->file('finished_picture'))
{
```

```php
// FORM POST DECISION
if ($request->input('submit')) {
    // APPROVE INSPECTIONS
    if ($data->status_id <= 2) {
        $data->update([
            $picture => $image,
            'subject' => $request->subject,
            'message' => $request->message,
            'area_id' => $area,
            'area_detail_id' => $area_detail,
            'readed' => 1,
            'matrix_id' => $matrix_id,
            $remark => $request->remark,
            'department_id' => $department,
            'status_id' => $status,
            'evaluation_target' => $request->evaluation_target,
            'updated_by' => Auth::id()
        ]);
        Timeline::create([
            'document_id' => $document_id,
            'activity' => 'Live Inspections',
            'remark' => $request->remark,
            'matrix_id' => $data->matrix_id,
            'status_id' => $status,
            'created_by' => $data->created_by,
            'updated_by' => Auth::id()
        ]);
        return redirect('/inspections/all');
    }
    // PROCEED INSPECTIONS
    if($data->status_id > 2) {
        Inspection::where('document_id', $document_id)
        ->update([
            $picture => $image,
            'matrix_id' => $matrix_id,
            'status_id' => $status,
            $remark => $request->remark,
        ]);
        return redirect('/inspections/all');
    }
}
```

```
450         // UPDATE INSPECTIONS
451         elseif ($request->input('update')) {
452             $data->update([
453                 'subject' => $request->subject,
454                 'message' => $request->message,
455                 'area_id' => $area,
456                 'area_detail_id' => $area_detail,
457                 'readed' => 1,
458                 'matrix_id' => $matrix_id,
459                 $remark => $request->remark,
460                 'department_id' => $department,
461                 'evaluation_target' => $request->evaluation_target,
462                 'updated_by' => Auth::id()
463             ]);
464             Timeline::create([
465                 'document_id' => $document_id,
466                 'activity' => 'Update - Live Inspections',
467                 'remark' => $request->remark,
468                 'matrix_id' => $data->matrix_id,
469                 'status_id' => $status-1,
470                 'created_by' => $data->created_by,
471                 'updated_by' => Auth::id()
472             ]);
473                 return back();
474         }
475     }
```

Tabel 5.9 Penjelasan kode program fungsi *actionInspections*

Baris	**Penjelasan**
396 - 407	Kode ini bertugas untuk memindahkan file gambar yang telah diinput ke penyimpanan public pada server. Kode ini hanya akan berjalan ketika user telah menginput gambar.
412 - 437	Kode ini berjalan ketika user melakukan approval terhadap data *inspections* dengan menekan tombol update pada halaman "Inspection Details". Kode ini bertugas untuk melakukan update data inspections dengan memasukkan data lanjutan dan merubah status menjadi *"Open"* lalu melakukan insert terhadap data *timelines*.

439 - 448	Kode ini berjalan ketika user melakukan evaluasi atau penyelesaian terhadap data *inspections* dengan menekan tombol *"submit"* pada halaman *"Inspection Details"*.
451 - 474	Kode ini berjalan ketika user menekan tombol *"update"* pada halaman *"Inspection Details"*. Kode ini bertugas untuk melakukan proses *update* data terhadap model *Inspection* menggunakan method eloquent update dan melakukan proses *insert* data terhadap model *Timeline* menggunakan method eloquent *create*.

5.1.4 Implementasi *Add User*

Kode program ini merupakan fungsi storeUser() milik controller UserController. Kode program ini akan berjalan saat user menekan tombol "*Submit"* di halaman *User Registration*. Kode program untuk fungsi dapat dilihat pada

```php
public function storeUser(Request $request)
{
    $dept = Position::where('id', $request->position_department)->first();
    $department = $dept->department_id;

    if ($request->file('profile_picture'))
    {
        $file = $request->file('profile_picture');
        $image = date('YmdHi').'-'.$file->getClientOriginalName();
    }
    else {
        $image = NULL;
    }

    $request->validate([
        'username' => ['unique:users'],
        'email' => ['unique:users'],
        'password' => ['confirmed'],
    ]);
        User::create([
            'nik' => $request->identification_number,
            'username' => $request->username,
            'email' => $request->email,
            'first_name' => ucwords($request->first_name),
            'last_name' => ucwords($request->last_name),
            'initial_name' => $request->initial_name,
            'gender_id' => $request->gender,
            'department_id' => $department,
            'position_id' => $request->position_department,
            'level_id' => $request->level,
            'inspection_target_id' => 404,
            'profile_picture' => $image,
            'join_date' => $request->join_date,
            'password' => bcrypt($request->password),
        ]);

    if ($request->file('profile_picture'))
    {
        $file->move('public/image', $image);
    }

    return redirect()->intended('/');
}
```

Gambar 5.30 dan dijelaskan pada Tabel 5.10.

Tabel 5.10 Penjelasan kode program fungsi *storeUser*

Baris	**Penjelasan**
83 - 90	Kode ini bertugas untuk melakukan validasi jika user telah menginput gambar, maka akan disimpan pada variabel $image. Jika tidak, maka variabel $image akan berisi nilai *"NULL"*.
92 - 96	Kode ini bertugas untuk melakukan validasi terhadap input diantaranya: username harus unik dan belum ada pada database, email harus unik dan belum ada pada database, dan password harus sama dengan password konfirmasi.
97 - 112	Kode ini bertugas untuk melakukan proses insert data terhadap model *Inspection* menggunakan method eloquent *create*.
114 - 117	Kode ini bertugas untuk memindahkan file gambar yang telah diinput ke penyimpanan public pada server. Kode ini hanya akan berjalan ketika user telah menginput gambar.

5.1.5 Implementasi get All Users

Kode program ini merupakan fungsi userAll() milik controller UserController. Kode program ini akan berjalan saat user berada di halaman *All Users*. Kode program untuk fungsi dapat dilihat pada Gambar 5.31 dan dijelaskan

```
15      public function userAll()
16      {
17          //
18          $active = 'users';
19          $data = User::with('department', 'position', 'level', 'gender', 'inspection_target')->get();
20          return view('users.index', compact('data', 'active'));
21      }
```

pada Tabel 5.11.

Tabel 5.11 Penjelasan kode program fungsi *userAll*

Baris	**Penjelasan**
19	Kode ini bertugas untuk mengambil seluruh data *users* yang tersimpan pada database. Proses pengambilan data menggunakan *Eager Loading*
20	Kode ini bertugas untuk menyimpan variabel yang telah dibuat dan membuka view pada file "*showInspections*".

5.1.6 Implementasi User Details

Kode program ini merupakan fungsi showUser() milik controller UserController. Kode program ini akan berjalan saat user berada di halaman *User Details*. Kode program untuk fungsi dapat dilihat pada Gambar 5.32 dan dijelaskan pada Tabel 5.12.

```
62    public function showUser(User $id)
63    {
64        //
65        $active = 'users';
66        $data = $id;
67        return view('users.profile', compact('data', 'active'));
68    }
```

Tabel 5.12 Penjelasan kode program fungsi *showUser*

Baris	**Penjelasan**
62 - 66	Kode ini bertugas untuk mengambil data *users* berdasarkan id dari nilai yang telah dimasukkan di url. Data diproses menggunakan method *route model binding*.
67	Kode ini bertugas untuk menyimpan variabel yang telah dibuat dan membuka view pada file "*profile*".

5.1.7 Implementasi My Profile

Kode program ini merupakan fungsi myProfile() milik controller UserController. Kode program ini akan berjalan saat user berada di halaman *myProfile*. Kode program untuk fungsi dapat dilihat pada Gambar 5.33 dan dijelaskan pada Tabel 5.13

```
70      public function myProfile()
71      {
72          // MY PROFILE
73          $active = 'users';
74          $data = User::where('id', Auth::id())->first();
75          return view('users.profile', compact('data', 'active'));
76      }
```

Tabel 5.13 Penjelasan kode program fungsi *myProfile*

Baris	**Penjelasan**
74	Kode ini bertugas untuk mengambil data User berdasarkan id sama dengan id user yang sedang login.
67	Kode ini bertugas untuk menyimpan variabel yang telah dibuat dan membuka view pada file "*profile*".

5.1.8 Implementasi add Feedback

Kode program ini merupakan fungsi storeFeedback() milik controller FeedbackController. Kode program ini akan berjalan saat user menekan tombol "*Submit"* di halaman *Send Feedback*. Kode program untuk fungsi dapat dilihat pada Gambar 5.34 dan dijelaskan pada Tabel 5.14.

```php
public function storeFeedback(Request $request)
{
    $lastRow = Feedback::all()->last();
    $check = Feedback::count();

    if ($check == 0)
    {
        $serial = SerialNumber::where('active', 1)->first();
        $number = $serial->serial_number;

    }

    else if (isset($check))
    {
        $order = 1;
        $number = $lastRow->document_id+$order;
    }

    if ($request->file('feedback_picture1'))
    {
        $file1 = $request->file('feedback_picture1');
        $image1 = date('YmdHi').'-'.$file1->getClientOriginalName();
    }

    else
    {
        $image1 = NULL;
    }

    Feedback::create([
        'document_id' => $number,
        'activity' => 'Feedback',
        'user_id' => Auth::id(),
        'department_id' => Auth::user()->department_id,
        'feedback_picture1' => $image1,
        'subject' => $request->subject,
        'message' => $request->message,
        'status_id' => 1,
        'answered' => 0,
        'remark' => NULL,
    ]);

    if ($image1 != NULL) {
        $file1->move('public/image', $image1);
    }

    return redirect('/feedback');
}
```

Tabel 5.14 Penjelasan kode program fungsi *storeFeedback*

Baris	**Penjelasan**
76 - 90	Kode ini bertugas untuk menentukan nomor dokumen yang digunakan untuk data inspeksi yang akan ditambahkan. kode ini akan melakukan validasi apakah sudah terdapat data inspeksi pada database, jika belum maka nomor yang digunakan adalah nomor dokumen sesuai dengan data pada tabel serial number dengan status aktif. Jika data inspeksi sebelumnya sudah ada pada database, maka nomor dokumen selanjutnya akan ditambahkan 1 sesuai dari nilai variabel $order.
92 - 109	Kode ini bertugas melakukan validasi apakah user telah menginput gambar yang disimpan ke variabel $request. Jika benar, maka file gambar tersebut akan diberi nama yang berisi tanggal dan nama file melalui function *getClientOriginalName()*, lalu nama file tersebut akan disimpan ke variabel *$image1*, Jika kondisi tersebut tidak terpenuhi, maka variabel *$image1* akan memiliki nilai "*NULL*".
147 - 161	Kode ini bertugas untuk melakukan proses insert data terhadap model *Inspection* menggunakan method eloquent *create*.
163 - 169	Kode ini bertugas untuk melakukan proses insert data terhadap model *Timeline* menggunakan method eloquent *create*.

173 - 176	Kode ini bertugas untuk memindahkan file gambar yang telah diinput ke penyimpanan public pada server. Kode ini hanya akan berjalan ketika user telah menginput gambar.

5.1.9 Implementasi get All Feedback

Kode program ini merupakan fungsi feedbackAll() milik controller FeedbackController. Kode program ini akan berjalan saat user berada di halaman *All Feedback*. Kode program untuk fungsi dapat dilihat pada Gambar 5.35 dan dijelaskan pada Tabel 5.15.

```
14      public function feedbackAll()
15      {
16          if (Auth::user()->level_id = 99) {
17              $active = 'notifications';
18          }
19          else {
20              $active = 'feedback';
21          }
22
23          $data = Feedback::with('user', 'department')->get();
24          // $date = Carbon::now();
25
26          return view('feedbacks.index', compact(
27              'active',
28              'data',
29              // 'date'
30          ));
31      }
```

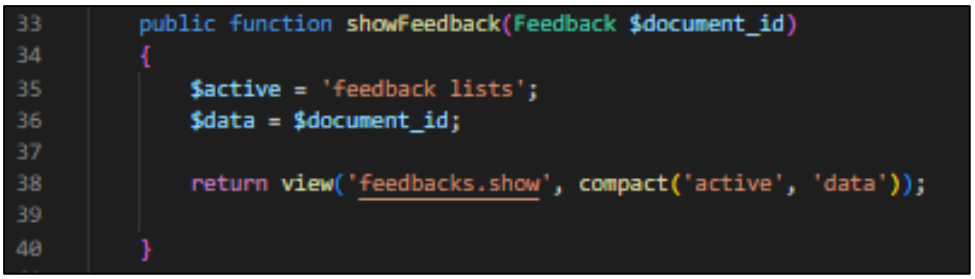
Tabel 5.15 Penjelasan kode program fungsi *feedbackAll*

Baris	**Penjelasan**
23	Kode ini bertugas untuk mengambil seluruh data *feedback* yang

	tersimpan pada database. Proses pengambilan data menggunakan *Eager Loading*.
26 - 30	Kode ini bertugas untuk menyimpan variabel yang telah dibuat dan membuka view pada file "feedbacks/*index*".

5.1.10 Implementasi Feedback Details

Kode program ini merupakan fungsi showFeedback() milik controller FeedbackController. Kode program ini akan berjalan saat user berada di halaman *Feedback Details*. Kode program untuk fungsi dapat dilihat pada Gambar 5.36 dan



dijelaskan pada Tabel 5.16.

Tabel 5.16 Penjelasan kode program fungsi *showFeedback*

Baris	**Penjelasan**
33 - 36	Kode ini bertugas untuk mengambil data *feedback* berdasarkan id dari nilai yang telah dimasukkan di url. Data diproses menggunakan method *route model binding*.
38	Kode ini bertugas untuk menyimpan variabel yang telah dibuat dan membuka view pada file "*feedbacks/show*".

5.1.11 Implementasi *Generate Random Inspections*

Kode program ini merupakan fungsi showEvaluations() milik controller InspectionController. Kode program ini akan berjalan saat user menekan tombol *Generate Random Inspections* di halaman *Sidebar Menu*. Kode program untuk

```
503     // GENERATE RANDOM INSPECTIONS
504     public function generateTarget(Request $request){
505         $deptsId = array(1, 2, 3, 4, 5, 6);
506             $randomsId = $deptsId[array_rand($deptsId)];
507             DB::table('users')->where('id', '=', Auth::id())
508                 ->update(['inspection_target_id' => $randomsId]);
509         return redirect('/randInspect/form');
510     }
```

fungsi dapat dilihat pada Gambar 5.11 dan dijelaskan pada Tabel 5.4.

Tabel 5.17 Penjelasan kode program fungsi *generateTarget*

Baris	**Penjelasan**
505	Kode ini bertugas untuk mengumpulkan nilai array dari angka 1 sampai dengan 6 sebagai identifier dari data departments yang disimpan ke dalam variabel *$deptsId*.
506	Kode ini bertugas untuk melakukan pengacakan nilai terhadap array pada variabel $deptsId yang disimpan ke dalam variabel

	$randomsId.
507 - 508	Kode ini bertugas untuk melakukan update data *users* berdasarkan id sama dengan user yang sedang login pada bagian inspection_target_id berisi nilai hasil dari pengacakan array pada variabel *$randomsId*.
38	Kode ini bertugas untuk mengalihkan tampilan ke halaman *Random Inspections*.

5.1.12 Implementasi Google Translate API

Kode program ini merupakan fungsi storeInspection() milik controller InspectionController. Kode program ini akan berjalan saat user menekan tombol "*Submit*" pada halaman *Live Inspections*. Kode program untuk fungsi ini dapat

```
1   <div id="google_translate_element"></div>
2
3   <script type="text/javascript">
4   function googleTranslateElementInit()
5   {
6       new google.translate.TranslateElement({pageLanguage: 'en'}, 'google_translate_element');
7   }
8   </script>
9
10  <script type="text/javascript"
11      src="//translate.google.com/translate_a/element.js?cb=googleTranslateElementInit"></script>
```

dilihat pada Gambar 5.8 dan dijelaskan pada Tabel 5.1.

Tabel 5.18 Penjelasan kode program fungsi *googleTranslateElementInit*

Baris	**Penjelasan**
1	Kode ini bertugas untuk menyediakan elemen *div* yang akan

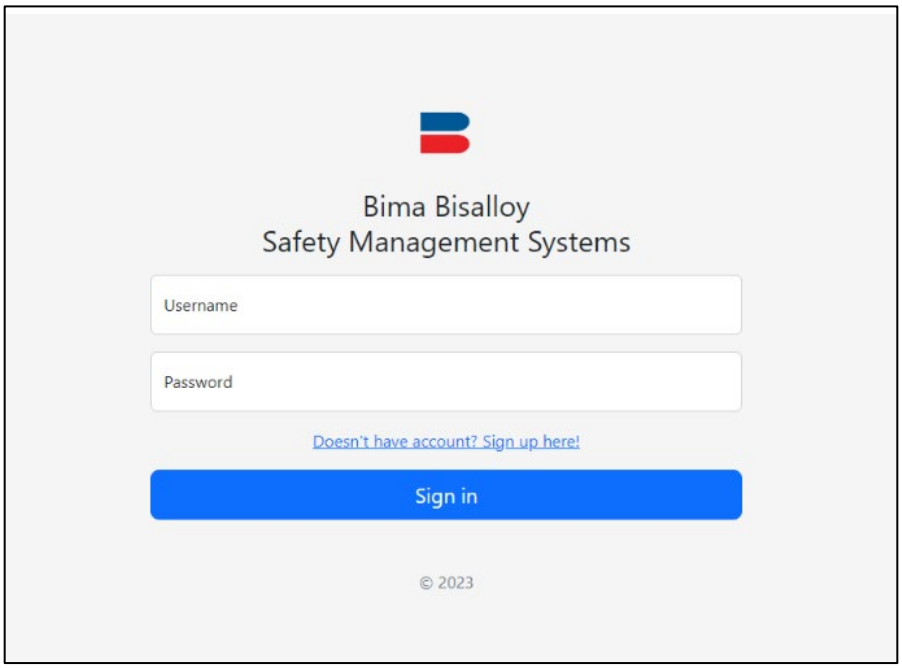
	digunakan untuk tampilan *google translate*.
3 - 8	Kode ini bertugas untuk menampilkan elemen *google translate* dan mengirim ke elemen *div* dengan id=”google_translate_element”.
10 - 11	Kode ini bertugas untuk mengambil *library javascript* dari direktori library sesuai dengan link yang tertera.

4.2 Implementasi Antarmuka

Bagian ini membahas hasil implementasi dari perancangan antarmuka yang telah di deskripsikan sebelumnya ke dalam kode program.

4.2.1 Halaman Login

Halaman ini menampilkan form input untuk melakukan login ke dalam sistem. Hasil implementasi halaman *Login* dapat dilihat pada Gambar 5.40.

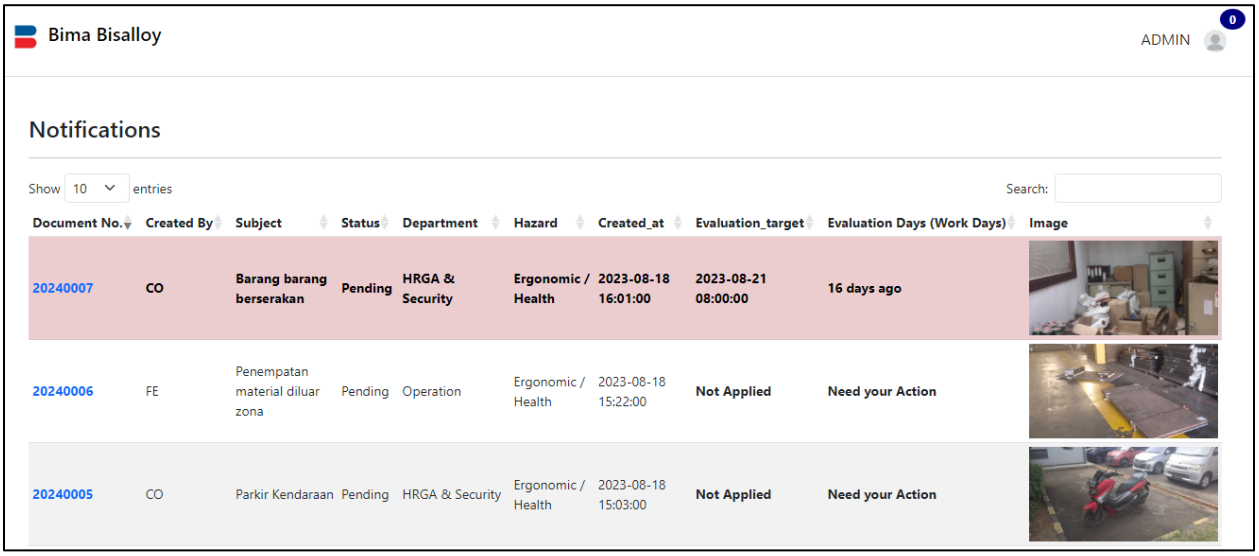

Bima Bisalloy
Safety Management Systems
Username
Password
Doesn't have account? Sign up here!
Sign in
© 2023

4.2.2 Halaman Dashboard

Halaman ini menampilkan rangkuman singkat tentang data inspeksi dalam bentuk grafik pada periode waktu tertentu. Hasil implementasi halaman *Dashboard* dapat dilihat pada Gambar 5.41.

4.2.3 Halaman Notifications

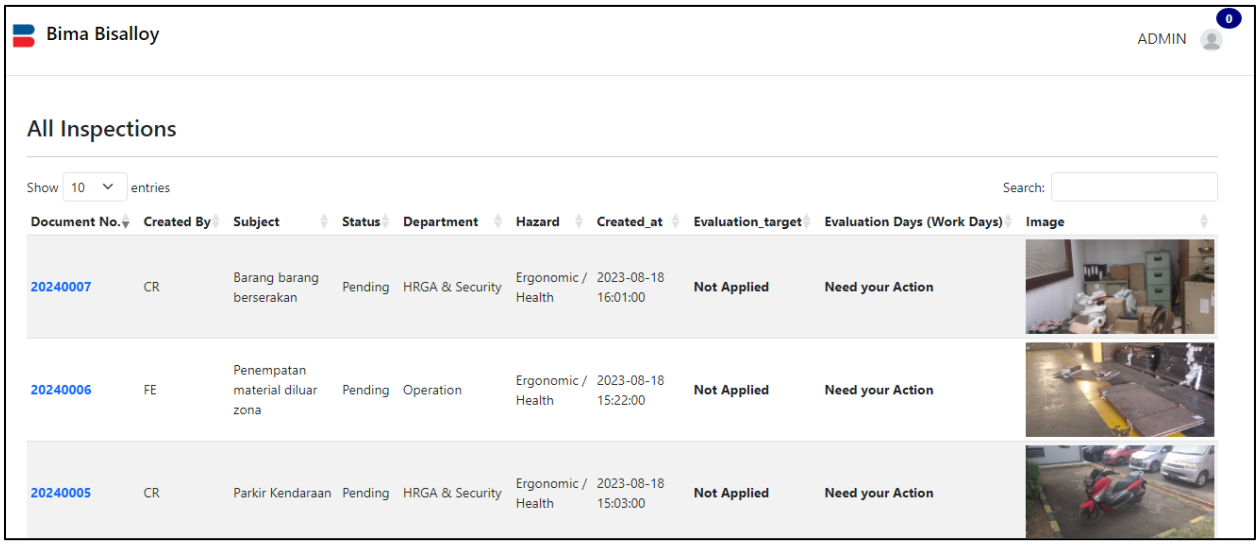
Halaman ini menampilkan data inspeksi yang terkait dengan user. Jika user yang saat ini login adalah user normal, maka halaman ini akan menampilkan data inspeksi yang telah dibuat oleh user itu sendiri. Sedangkan jika user yang saat ini login adalah user safety team, maka halaman ini akan menampilkan data inspeksi yang berstatus "*pending*" atau "*finished*". Hasil implementasi halaman



Notifications dapat dilihat pada Gambar 5.42.

4.2.4 Halaman All Inspections

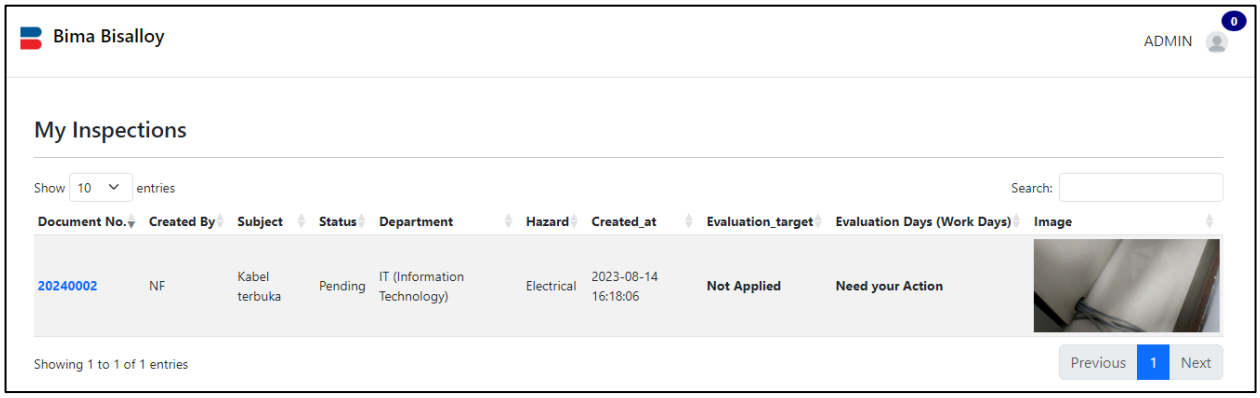
Halaman ini menampilkan daftar sejarah seluruh inspeksi yang telah masuk ke dalam sistem. Hasil implementasi halaman *All Inspections* dapat dilihat



pada Gambar 5.43.

4.2.5 Halaman My Inspections

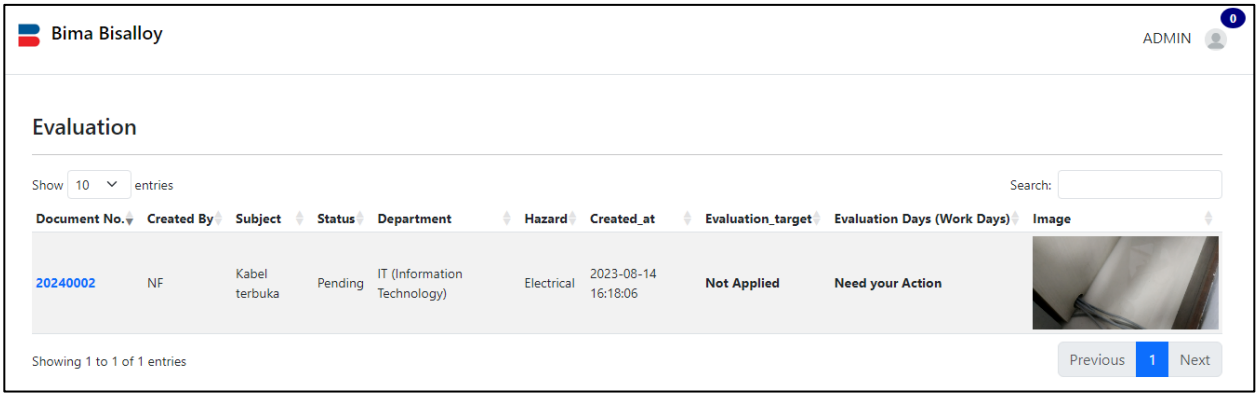
Halaman ini menampilkan daftar sejarah inspeksi yang pernah dilaporkan oleh pengguna. Hasil implementasi halaman *My Inspections* dapat dilihat pada



Gambar 5.44.

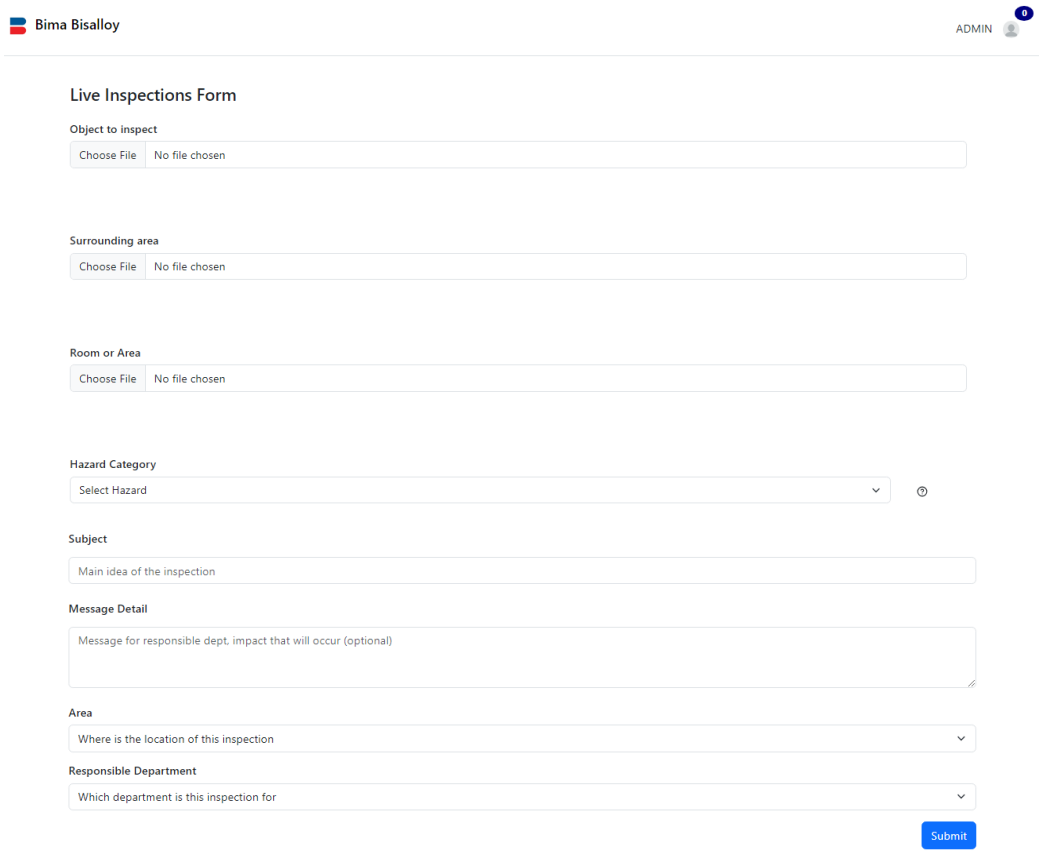
4.2.6 Halaman Evaluations

Halaman ini menampilkan data inspeksi berdasarkan data departemen dari data pengguna. Hasil implementasi halaman *Evaluations* dapat dilhat pada Gambar 5.45.



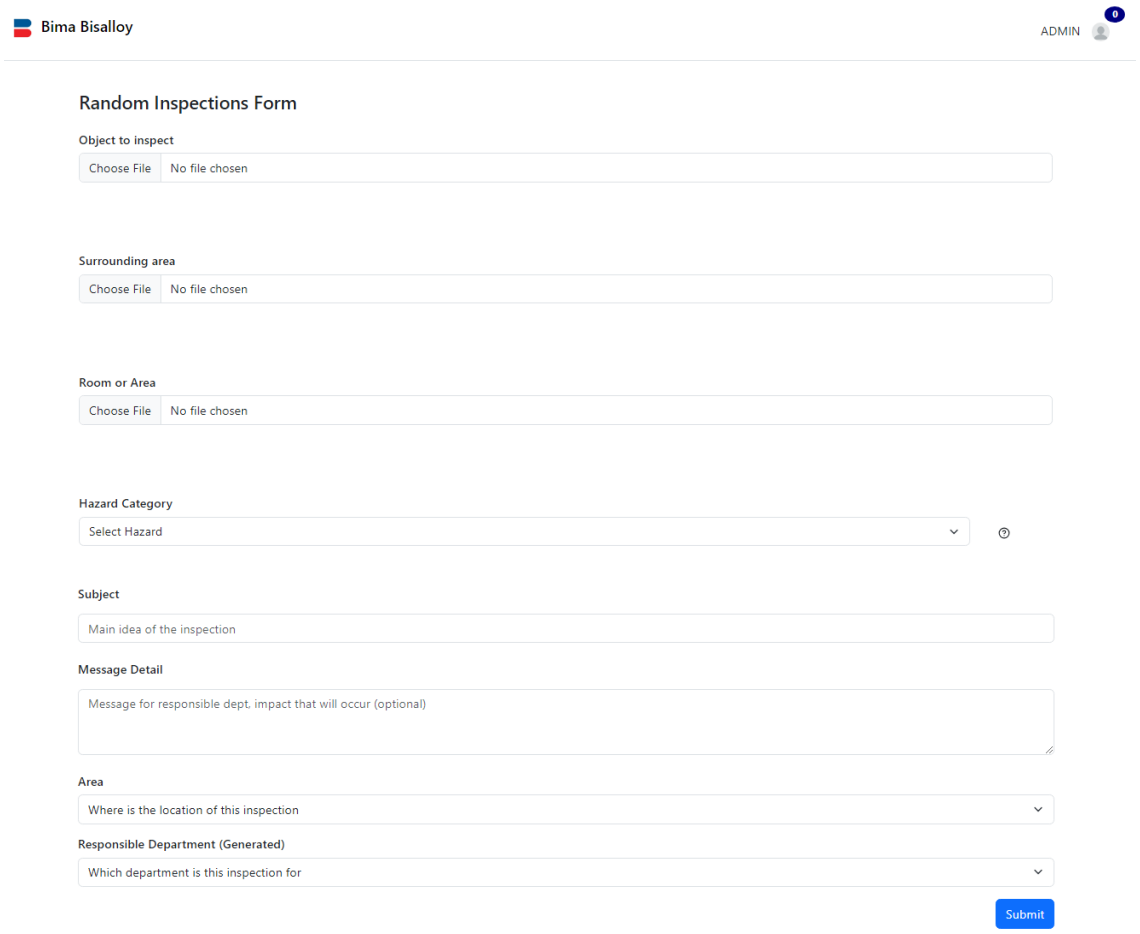
4.2.7 Halaman Live Inspections

Halaman ini menampilkan form yang digunakan untuk menambahkan inspeksi baru. Hasil implementasi halaman *Live Inspections* dapat dilihat pada Gambar 5.46.



4.2.8 Halaman Random Inspections

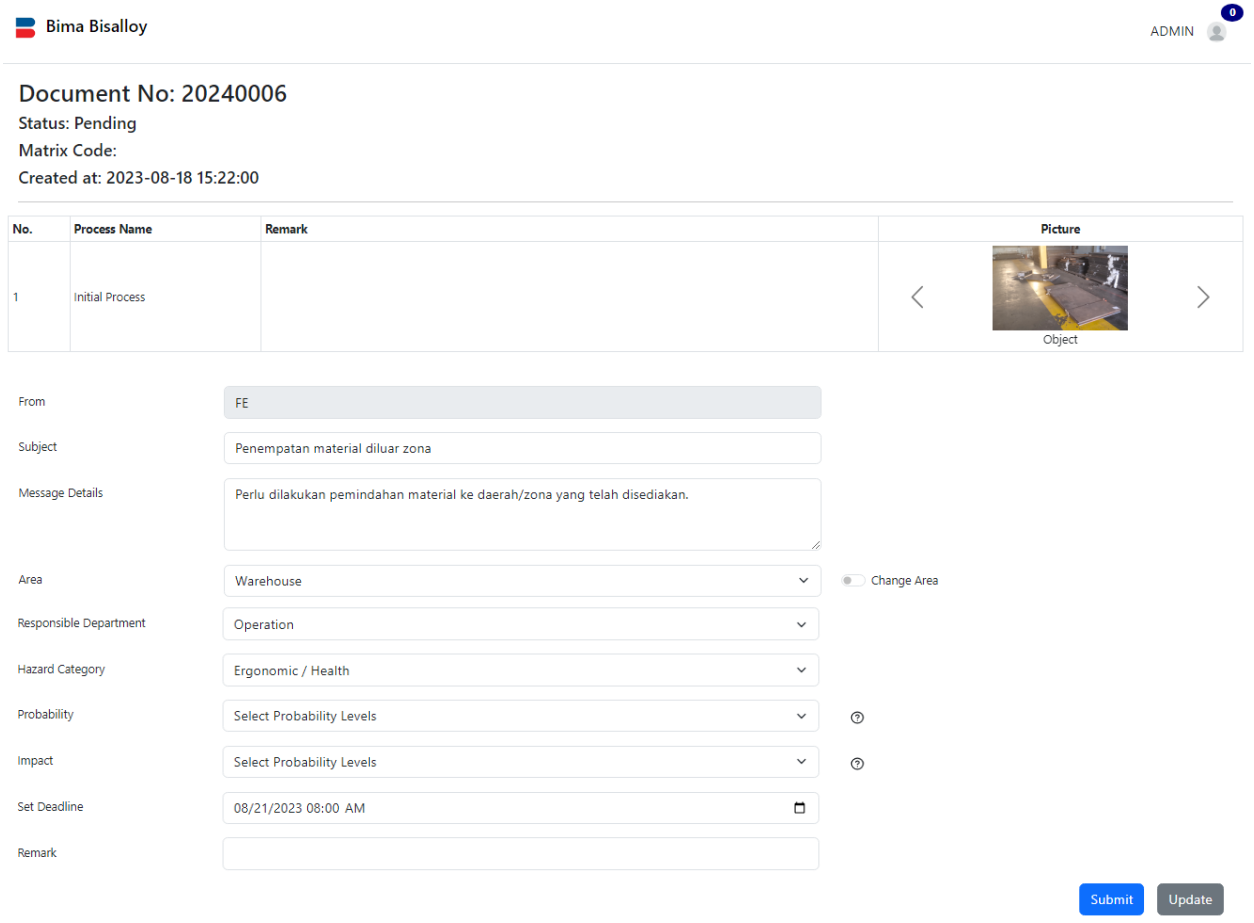
Halaman ini menampilkan form yang digunakan untuk menambahkan inspeksi baru. Selain itu halaman ini hanya menampilkan pilihan departemen sesuai dengan data *target_inspections* pada tabel *users* sesuai dengan user yang saat ini sedang login. Hasil implementasi halaman *Random Inspections* dapat



dilihat pada Gambar 5.47.

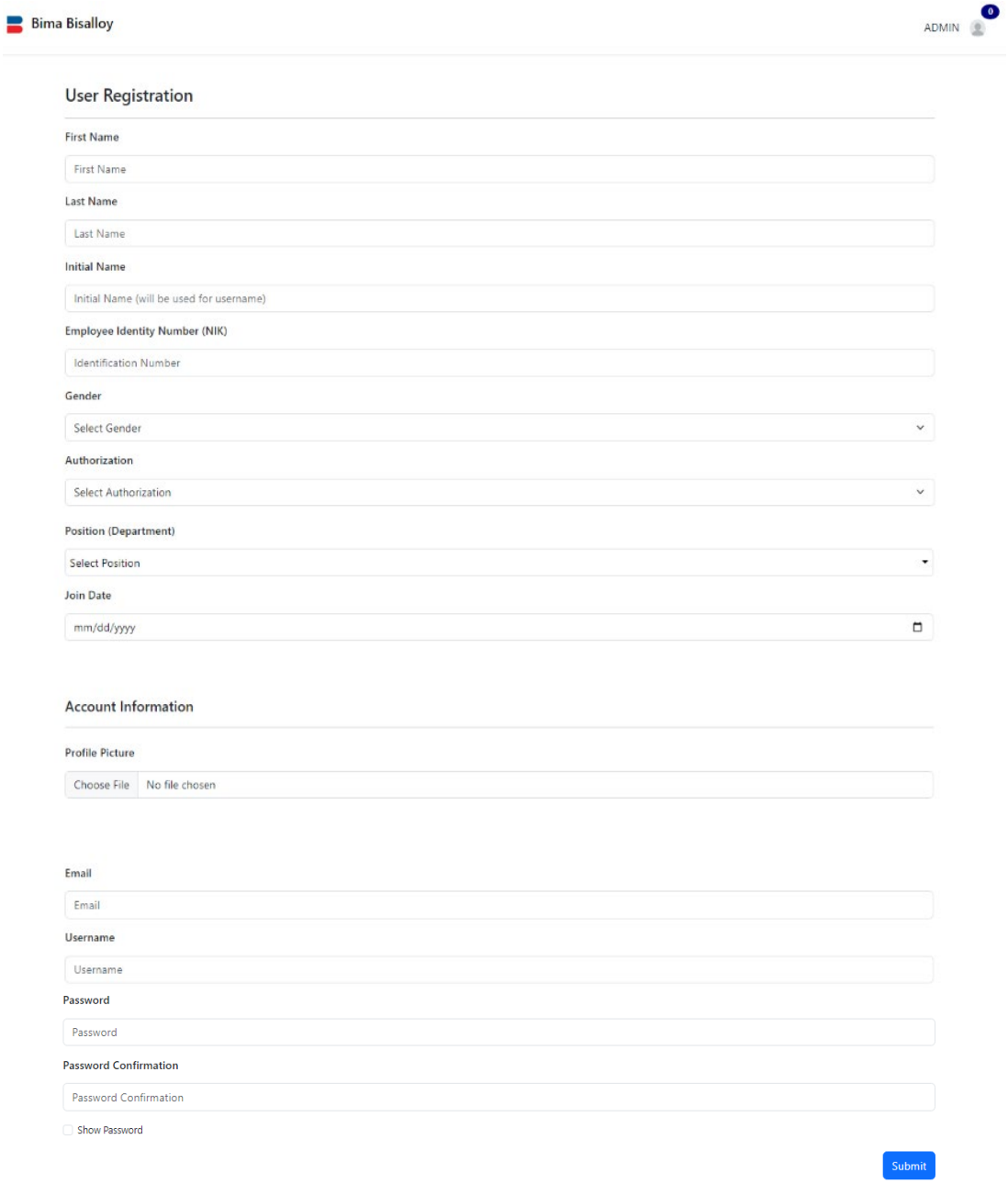
4.2.9 Halaman Inspection Details

Halaman ini menampilkan rincian data inspeksi yang sudah ditambahkan ke dalam sistem. Hasil implementasi halaman *Inspection Details* dapat dilihat pada Gambar 5.48.



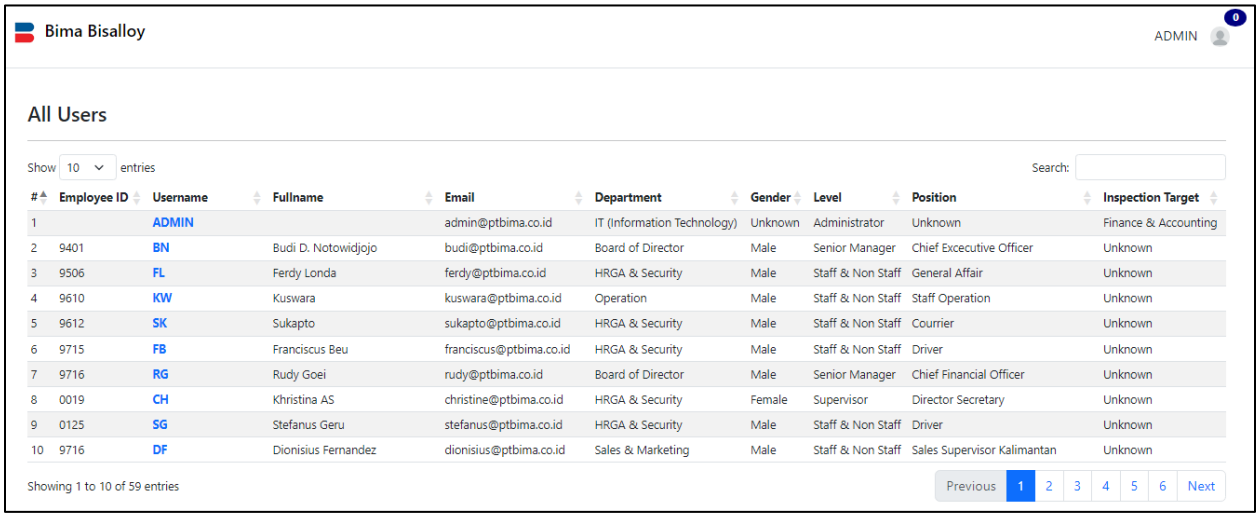
4.2.10 Halaman User Registration

Halaman ini menampilkan form untuk melakukan registrasi user. Hasil implementasi halaman *User Registration* dapat dilihat pada Gambar 5.49.



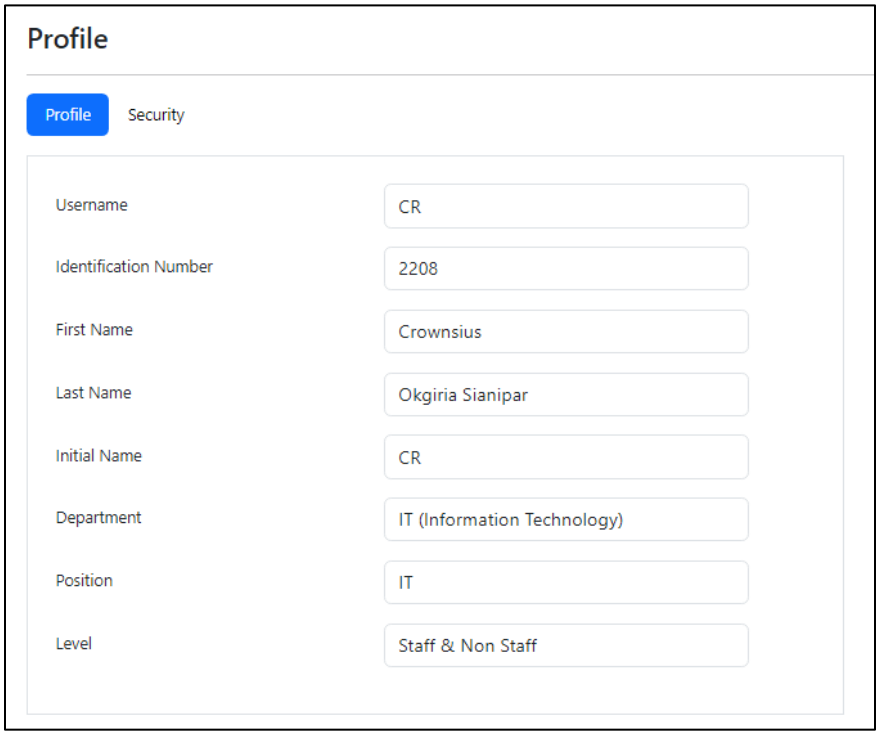
4.2.11 Halaman All Users

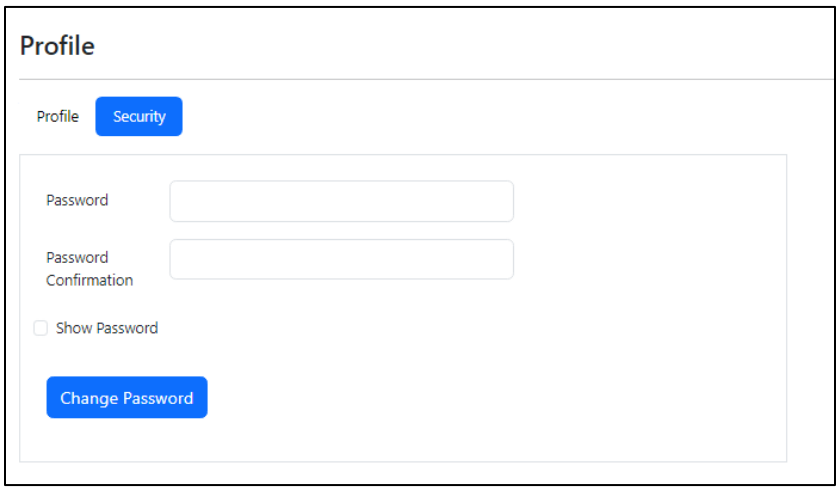
Halaman ini menampilkan rincian data user secara detail. Di halaman ini user yang berwenang dapat mengubah data user. Hasil implementasi halaman *Profiles* dapat dilihat pada Gambar 5.50.



4.2.12 Halaman Profile

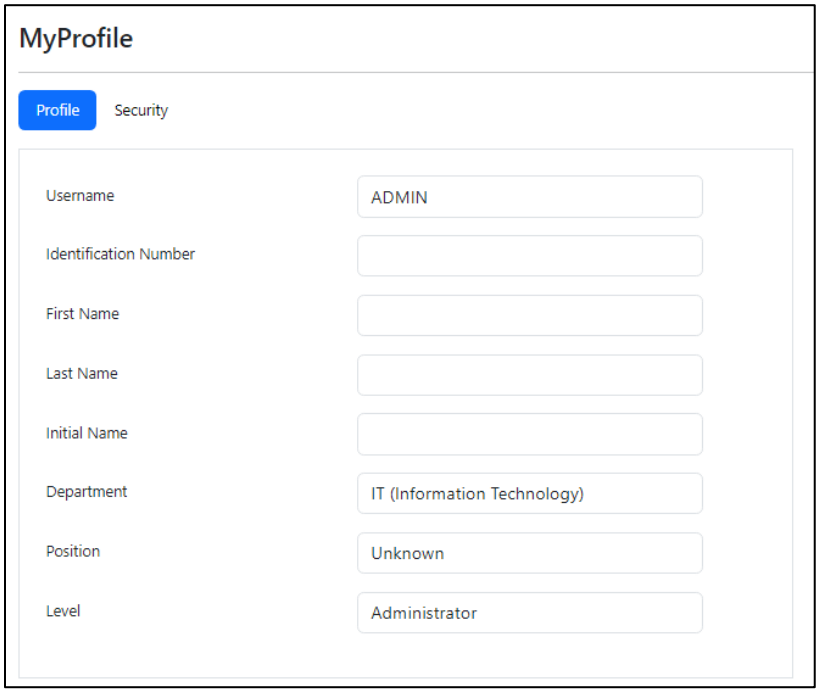
Kode menampilkan rincian data user secara detail. Di halaman ini user yang berwenang dapat mengubah data user. Hasil implementasi halaman *Profile* dapat dilihat pada Gambar 5.51 dan 5.52.

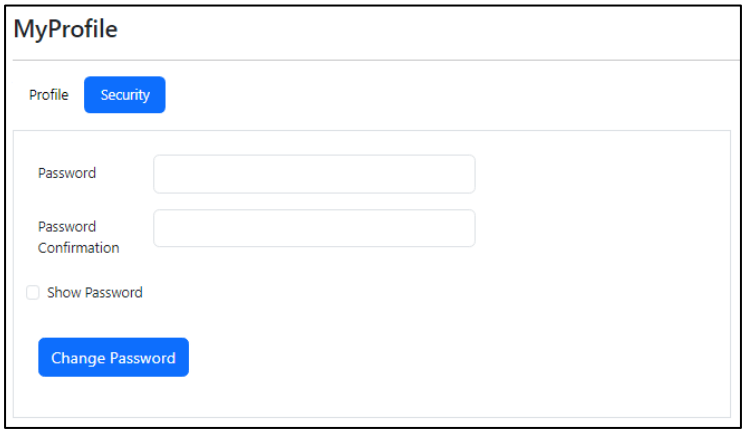




4.2.13 Halaman My Profile

Halaman ini menampilkan rincian data user secara detail. Di halaman ini user yang berwenang dapat mengubah data user. Hasil implementasi halaman *My Profile* dapat dilihat pada Gambar 5.53 dan 5.54.





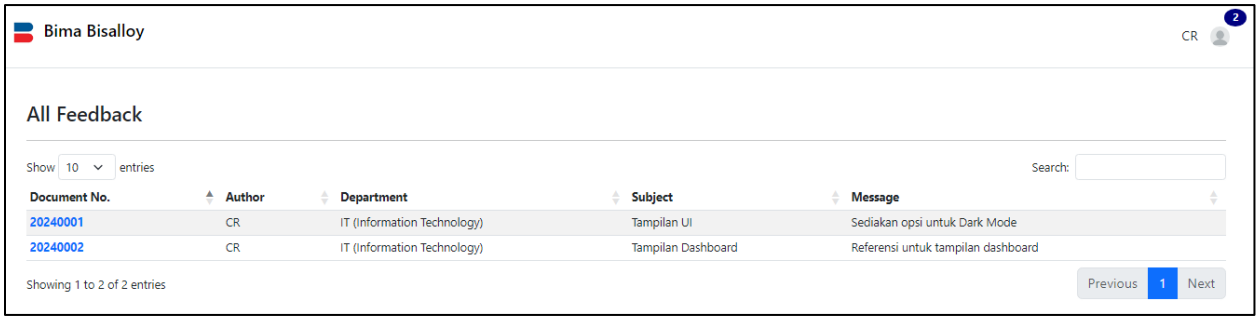
4.2.14 Halaman Send Feedback

Halaman ini menampilkan form yang digunakan untuk menambahkan feedback

baru. Hasil implementasi halaman Send Feedback dapat dilihat pada Gambar 5.55.

4.2.15 Halaman All Feedback

Halaman ini menampilkan rincian data user secara detail. Di halaman ini user yang berwenang dapat mengubah data user. Hasil implementasi halaman *All Feedback* dapat dilihat pada Gambar 5.56.



4.2.16 Halaman Feedback Details

Halaman ini menampilkan rincian data inspeksi yang sudah ditambahkan ke dalam sistem. Hasil implementasi halaman *Feedback Details* dapat dilihat pada Gambar 5.57.

BAB VI

PENGUJIAN DAN EVALUASI

5.1 Pengujian Fungsional

Pengujian fungsional ditujukan untuk memvalidasi kebutuhan fungsional yang dibuat dapat berjalan dengan semestinya dan telah sesuai dengan kebutuhan yang telah ditetapkan pada analisis kebutuhan fungsional dan *use case narrative*. Pengujian fungsional menggunakan metode *black box testing* dengan menguji apakah hasil input dan output yang diberikan sesuai atau tidak. Jika input dan output yang didapatkan sesuai dengan kebutuhan maka dianggap valid, jika tidak sesuai maka akan dianggap tidak valid. Pengujian fungsional dapat dilihat pada Tabel 6.1 sampai dengan Tabel 6.17.

Tabel 6.1 Kasus uji halaman *Login*

Kode Kebutuhan		**SRS_SMS_F_01**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Masuk ke halaman login	Menampilkan form username & password, tombol untuk opsi lupa password.	Sukses	Valid
2.	Validasi data akun	Melakukan validasi jika username atau password yang di masukkan benar maka akan dialihkan ke halaman *Dasboard*. Sedangkan jika hasil input salah maka akan muncul pesan error "username atau password salah".	Sukses	Valid

Tabel 6.2 Kasus uji halaman *Dashboard*

Kode Kebutuhan		**SRS_SMS_F_02**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Masuk ke halaman *Dashboard*	Menampilkan data Dashboard (Rangkuman Inspeksi dan data karyawan).	Sukses	Valid
2.	Menyaring berdasarkan periode *YEJ* tertentu	Menampilkan daftar data berdasarkan periode *YEJ* yang telah dipilih tanpa melakukan refresh.	Sukses	Valid
3.	Melakukan hover pada bagian poin di grafik	Menampilkan informasi detail pada tiap poin yang di hover.	Sukses	Valid
4.	Mengklik bagian poin di grafik	Mengalihkan pada menu sesuai dengan subjek dari grafik yang dipilih.	Sukses	Valid
5.	Melakukan seleksi data (jika tersedia check box)	Menampilkan data sesuai dengan check box yang dipilih.	Sukses	Valid

Tabel 6.3 Kasus uji fitur *Live Inspections*

Kode Kebutuhan		**SRS_SMS_F_03**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Masuk ke halaman *Live Inspections*	Menampilkan halaman *Live Inspection.*	Sukses	Valid
2.	Menekan tombol “Choose File” pada bagian: *Object to inspect, Surrounding area,* dan *Room or area*.	Masuk ke form pengambilan gambar melalui penyimpanan lokal, lalu melakukan preview gambar yang telah dipilih sesuai dengan urutan. Note: tersedia fitur foto secara live jika menggunakan hp/smartphone	Sukses	Valid
3.	Menambahkan rincian data inspeksi dengan lengkap	Menyimpan data ke database. Mengalihkan ke halaman *My Inspections*.	Sukses	Valid
4.	Menambahkan rincian	Memberikan pesan Error sesuai	Sukses	Valid

	data inspeksi tidak lengkap	dengan bagian yang tidak lengkap.		

Tabel 6.4 Kasus uji halaman *Random Inspections*

Kode Kebutuhan		**SRS_SMS_F_04**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Masuk ke halaman *Random Inspections*	Melakukan validasi jika target random inspections sudah ditetapkan, maka dialihkan ke halaman Random Inspection. Jika tidak, maka akan muncul pesan error "Random Inspections Target is not yet generated".	Sukses	Valid
2.	Menekan tombol "Choose File" pada bagian: *Object to inspect, Surrounding area,* dan *Room or area*.	Masuk ke form pengambilan gambar melalui penyimpanan lokal, lalu melakukan preview gambar yang telah dipilih sesuai dengan urutan. Note: tersedia fitur foto secara live jika menggunakan hp/smartphone	Sukses	Valid
3.	Menambahkan rincian data inspeksi dengan lengkap	Menyimpan data ke database. Mengalihkan ke halaman *My Inspections*.	Sukses	Valid
4.	Menambahkan rincian data inspeksi tidak lengkap	Memberikan pesan Error sesuai dengan bagian yang tidak lengkap.	Sukses	Valid

Tabel 6.5 Kasus uji fitur *Update Inspection Details*

Kode Kebutuhan		**SRS_SMS_F_05**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Masuk ke halaman *Inspection Details*	Menampilkan detail rincian data inspeksi.	Sukses	Valid
2.	Menekan tombol update	Melakukan update data dan mengalihkan kembali ke halaman Inspection Details	Sukses	Valid

Tabel 6.6 Kasus uji fitur *Approve Inspections*

Kode Kebutuhan		**SRS_SMS_F_06**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Menambahkan rincian data inspeksi dengan lengkap	Menyimpan data ke database. Mengalihkan ke halaman *My Inspections*.	Sukses	Valid
2.	Menambahkan rincian data inspeksi tidak lengkap	Memberikan pesan Error sesuai dengan bagian yang tidak lengkap.	Sukses	Valid

Tabel 6.7 Kasus uji fitur *Proceed Inspections*

Kode Kebutuhan		**SRS_SMS_F_07**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Menambahkan rincian data inspeksi dengan lengkap	Menyimpan data ke database. Mengalihkan ke halaman *My Inspections*.	Sukses	Valid
2.	Menambahkan rincian data inspeksi tidak lengkap	Memberikan pesan Error sesuai dengan bagian yang tidak lengkap.	Sukses	Valid

Tabel 6.8 Kasus uji fitur *Finish Inspections*

Kode Kebutuhan		**SRS_SMS_F_08**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Menambahkan rincian data inspeksi dengan lengkap	Menyimpan data ke database. Mengalihkan ke halaman *My Inspections*.	Sukses	Valid
2.	Menambahkan rincian data inspeksi tidak lengkap	Memberikan pesan Error sesuai dengan bagian yang tidak lengkap.	Sukses	Valid

Tabel 6.9 Kasus uji fitur *Close Inspections*

Kode Kebutuhan		**SRS_SMS_F_09**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Menambahkan rincian data inspeksi dengan lengkap	Menyimpan data ke database. Mengalihkan ke halaman *My Inspections*.	Sukses	Valid
2.	Menambahkan rincian data inspeksi tidak lengkap	Memberikan pesan Error sesuai dengan bagian yang tidak lengkap.	Sukses	Valid

Tabel 6.10 Kasus uji fitur *User Registration*

Kode Kebutuhan		**SRS_SMS_F_10**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Masuk ke halaman User Registration	Menampilkan halaman *User Registration.*	Sukses	Valid
2.	Menekan tombol “Choose File” pada bagian: *Profile Picture.*	Masuk ke form pengambilan gambar melalui penyimpanan lokal, lalu melakukan preview gambar yang telah dipilih sesuai dengan urutan.	Sukses	Valid

		Note: tersedia fitur foto secara live jika menggunakan hp/smartphone		
3.	Menambahkan rincian data inspeksi dengan lengkap	Menyimpan data ke database. Mengalihkan ke halaman *All Users*.	Sukses	Valid
4.	Menambahkan rincian data inspeksi tidak lengkap	Memberikan pesan Error sesuai dengan bagian yang tidak lengkap.	Sukses	Valid

Tabel 6.11 Kasus uji mengubah *User Details*

Kode Kebutuhan		**SRS_SMS_F_11**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Masuk ke halaman *User Profile.*	Menampilkan detail rincian data user.	Sukses	Valid
2.	Menekan tombol update	Melakukan update data dan mengalihkan kembali ke halaman *User Profile.*	Sukses	Valid

Tabel 6.12 Kasus uji menonaktifkan *User*

Kode Kebutuhan		**SRS_SMS_F_12**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Masuk ke halaman *User Profile.*	Menampilkan detail rincian data user.	Sukses	Valid
2.	Menekan tombol disable	Jika user tidak memiliki inspeksi berstatus selain “closed”, maka Sistem akan melakukan update data user menjadi disabled dan mengalihkan kembali ke halaman *All Users.* Jika sebaliknya, sistem akan memberikan pesan error	Sukses	Valid

		dan mengalihkan ke halaman *User Profile*.		

Tabel 6.13 Kasus uji menghapus *User*

Kode Kebutuhan		**SRS_SMS_F_13**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Masuk ke halaman *User Profile.*	Menampilkan detail rincian data user.	Sukses	Valid
2.	Menekan tombol delete	Jika user tidak memiliki inspeksi, maka Sistem akan menghapus data user dan mengalihkan kembali ke halaman *All Users.* Jika sebaliknya, sistem akan memberikan pesan error dan mengalihkan ke halaman *User Profile*.	Sukses	Valid

Tabel 6.14 Kasus uji halaman *Send Feedback*

Kode Kebutuhan		**SRS_SMS_F_14**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Masuk ke halaman Send Feedback	Menampilkan halaman Send Feedback.	Sukses	Valid
2.	Menekan tombol “Choose File” pada bagian: *Add Picture.*	Masuk ke form pengambilan gambar melalui penyimpanan lokal, lalu melakukan preview gambar yang telah dipilih sesuai dengan urutan. Note: tersedia fitur foto secara live jika menggunakan hp/smartphone	Sukses	Valid
3	Menekan tombol submit	Melakukan validasi jika data sudah lengkap maka hasil input akan disimpan ke Database dan	Sukses	Valid

		mengalihkan ke halaman *All Feedback*. Sedangkan jika data tidak lengkap maka akan muncul pesan error sesuai bagian mana yang belum lengkap.		

Tabel 6.15 Kasus uji *Feedback Details*

Kode Kebutuhan		**SRS_SMS_F_15**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Menampilkan detail feedback yang telah masuk ke dalam sistem.	Menampilkan detail rincian feedback.	Sukses	Valid

Tabel 6.16 Kasus uji fitur *Generate Random Inspections*

Kode Kebutuhan		**SRS_SMS_F_16**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Menekan tombol *Generate Random Inspections.*	Melakukan pengacakan target inspeksi terhadap user agar dapat melakukan *Random Inspections.*	Sukses	Valid

Tabel 6.17 Kasus uji fitur *Google Translate API*

Kode Kebutuhan		**SRS_SMS_F_17**		
No.	**Skenario**	**Hasil yang di Harapkan**	**Hasil**	**Status**
1.	Memilih bahasa yang ingin digunakan untuk bahasa terjemahan.	Menerjemahkan seluruh teks pada halaman ke bahasa terjemahan.	Sukses	Valid
2.	Menekan tombol show original	Mengembalikan ke bahasa awal sebelum	Sukses	Valid

		diterjemahkan tanpa menutup dialog Google Translate API.		
3.	Menekan tombol cancel atau “x”.	Mengembalikan ke bahasa awal dan menutup dialog Google Translate API.	Sukses	Valid

5.2 Pengujian Kompatibilitas

Pengujian kompatibilitas ditujukan untuk mengetahui apakah sistem *Safety Management System* dapat berjalan dengan baik pada lingkungan, perangkat lunak maupun perangkat keras yang berbeda-beda. Pengujian dilakukan dengan menjalankan aplikasi *Safety Management System* pada beberapa perangkat yane berbeda diantaranya, smartphone dan pc yang memiliki resolusi layar yang berbeda-beda. Perangkat uji harus dapat menyelesaikan kasus uji berjumlah 17 kasus yang mangacu pada *use case*. Jika kasus uji dapat dijalankan dengan baik dan tidak terjadi *crash* pada aplikasi maka dianggap valid, namun jika gagal dalam menjalankan kasus uji maka dianggap tidak valid. Perangkat uji yang digunakan berjumlah 3 buah yang memiliki resolusi yang berbeda-beda. Berikut merupakan hasil pengujian kompatibilitas yang dijabarkan pada Tabel 6.18.

Tabel 6.18 Hasil Pengujian Kompabilitas

No.	Use Case	Perangkat Uji		
		PC1	PC2	Smartphone
		Resolusi: 1280x720	Resolusi: 1920x1080	Resolusi: Smartphone
1.	Halaman Login	B	B	B
2.	Halaman Dashboard	B	B	B
3.	Melakukan Live Inspections	B	B	B
4.	Melakukan Random Inspections	B	B	B

5.	Mengubah detail inspeksi	B	B	B
6.	Melakukan Approve Inspections	B	B	B
7.	Melakukan Proceed Inspections	B	B	B
8.	Melakukan Finish Inspections	B	B	B
9.	Melakukan Close Inspections	B	B	B
10.	Menambah user	B	B	B
11.	Mengubah detail user	B	B	B
12.	Menonaktifkan user	B	B	B
13.	Menghapus user	B	B	B
14.	Menambah feedback	B	B	B
15.	Halaman Feedback Details	B	B	B
16.	Melakukan Generate Random Inspections	B	B	B
17.	Menerjemahkan halaman (Google Translate API)	B	B	B
Catatan: Baik; K: Kurang Baik; TB: Tidak Baik.				

Berdasarkan hasil pengujian kompatibilitas yang telah dilakukan, dapat disimpulkan bahwa semua use case dapat dijalankan 100% tanpa terjadinya *error* atau *crash* pada perangkat yang menggunakan resolusi 1280x720 ke atas pada beberapa perangkat tersebut. Pada perangkat atau resolusi tertentu kemungkinan dapat terjadi perbedaan tampilan dikarenakan tampilan yang sudah dibuat responsive pada saat pengembangan.

BAB VII

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dari hasil yang telah didapatkan berdasarkan tahap analisis kebutuhan, perancangan sistem, implementasi sistem, dan pengujian serta evaluasi yang telah selesai dilakukan sebelumnya, maka dapat disimpulkan sebagai berikut:

1. Analisis kebutuhan pada aplikasi Safety didasarkan pada studi Health Safety and Environment dan Sistem Informasi menghasilkan 17 kebutuhan fungsional yang di bagi menjadi 7 grup yaitu *Add Inspections* dimana pengguna dapat melakukan upload laporan inspeksi secara langsung *(live inspections)* atau menggunakan metode pengacakan dari sistem *(random inspections)* ke dalam sistem. *Inspection Details* dimana pengguna dapat melihat perkembangan informasi mengenai proses evaluasi terhadap inspeksi dan mengubah rincian laporan inspeksi. *Open / Close Inspections* dimana pengguna dapat melakukan approval terhadap inspeksi, dan melakukan closing terhadap inspeksi. *Evaluations* dimana pengguna dapat melaporkan proses evaluasi terhadap inspeksi atau melaporkan bahwa proses evaluasi telah selesai. serta *Manage Users* dimana pengguna dapat menambahkan, mengubah, menonaktifkan, menghapus data user secara individual. *All Users Authority* dimana pengguna dapat melakukan *login* ke dalam sistem, melihat data statistik pada halaman *Dashboard*, menambahkan data feedback, melihat rincian data feedback, mentrigger sistem agar melakukan pengacakan target inspeksi menggunakan fitur

generate *Random Inspections*, menerjemahkan halaman menggunakan *Google Translate API*.

2. Hasil rancangan yang didasarkan pada 17 kebutuhan fungsional menghasilkan perancangan berupa Entity Realtionship Diagram, Perancangan Basis Data dan Perancangan Antarmuka yang digambarkan dalam bentuk low-fidelity.
3. Sistem diimplementasikan pada platform aplikasi berbasis web dengan menggunakan sistem operasi Windows, arsitektur MVC *(Model-View-Controller)*, bahasa pemrograman PHP dan kerangka kerja *Laravel Framework*.
4. Pengujian yang dilakukan dengan pengujian fungsionalitas menunjukkan bahwa fitu-fitur kebutuhan fungsionalitas memiliki tingkat keberhasilan 100% sesuai dengan yang diharapkan. Fitur-fitur pada *Add Inspections* dapat menambahkan data inspeksi baru secara langsung *(live inspections),* atau menggunakan metode pengacakan dari sistem *(random inspections)*. Fitur-fitur pada *Inspection Details* melihat inspeksi terbaru, melihat detail inspeksi, mengubah inspeksi. Fitur-fitur *Open / Close Inspections* dapat melakukan approval terhadap inspeksi, dan melakukan closing terhadap inspeksi. Fitur-fitur pada *Evaluations* dapat melaporkan proses dan penyelesaian evaluasi terhadap inspeksi. Fitur- fitur pada *Manage Users* dapat menambahkan, mengubah, menonaktifkan, menghapus data user secara individual. Sedangkan dalam pengujian kompabilitas sistem, aplikasi safety management system dapat dijalankan pada hampir semua jenis perangkat elektronik seperti PC *(Personal Computer)* dan Laptop yang memiliki resolusi setidaknya diatas 1280 x 720, smartphone yang setidaknya memiliki sistem operasi android atau iOS.

6.2 Saran

Pengembangan lebih lanjut dari sistem ini membuka ruang untuk perbaikan, baik dengan menyempurnakan fitur yang sudah ada maupun menambahkan yang baru. Berikut adalah beberapa kemungkinan tersebut.

1. Menambahkan fitur Manage Hazard untuk memungkinkan user dapat menambahkan jenis kategori bahaya baru.
2. Memperbaiki *user experience* dan desain *user interface* untuk meningkatkan kepuasan pengguna dalam menggunakan aplikasi.
3. Menambahkan fitur two factor authentication agar proses login dapat dilakukan dengan lebih aman, terhindar dari potensi pencurian data/hacking.

REFERENCES

[1] Jerusalem, M.A & Khayati, E.Z. 2010. Keselamatan dan Kesehatan Kerja. Yogyakarta: Fakultas Teknik Universitas Negeri Yogyakarta.

[2] Caspio, "4 Reason Why You Should Turn Excel Into Web Apps Now," [Online]. Available: https://www.caspio.com/blog/4-reasons-why-you-should-turn-excel-into-web-apps-now/. [Accessed 8 September 2023].

[3] Dorota Klimecka-Tatar, Robert Ulewicz, Manuela Ingaldi "Minimizing occupational risk by automation of the special processes-based on occupational risk assessment," Czestochowa University of Technology, al. Armii Krajowej 19b, 42-201 Czetochowa, Poland: Elsevier, 2023.

[4] C. V. Geambasu, I Jianu, I Jianu, A Gavrila "Influence Factors for the Choice of a Software Development Methodology," Journal of Accounting and Management Information Systems, vol. 10, no. 4, pp. 479-494, 2011].

[5] Lucid Content Team, "4 Phases of Rapid Application Development Methodology," [Online]. Available: https://www.lucidchart.com/blog/rapid- application-development-methodology. [Accessed 7 September 2023].

[6] Kiscerti.co.id, "Tugas dan Tanggung Jawab HSE," [Online]. Available: https://kiscerti.co.id/artikel/tugas-dan-tanggung-jawab-hse. [Accessed 7 September 2023].

[7] TechTarget, "What is information systems (IS)," [Online]. Available: https://www.techtarget.com/whatis/definition/IS-information-system-or-information-services [Accessed 7 September 2023].

[8] PHP.NET, "What is PHP," [Online]. Available: https://www.php.net/manual/en/intro-whatis.php [Accessed 7 September 2023].

[9] Laravel, "Laravel - The PHP Framework for Web Artisans," [Online]. Available: https://laravel.com/ [Accessed 7 September 2023].

[10] Wikipedia, "Model-View-Controller - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Model-view-controller [Accessed 7 September 2023].

[11] J. Sauro and J. Lewis, Quantifying the User Experience, Cambridge: Elsevier, 2016.

[12] M. A. Mohammed, D. A. Muhammed and J. M. Abdullah, "Practical Approaches of Transforming ER Diagram into Tables," International Journal of Multidisciplinary and Scientific Emerging Research, vol. 4, no. 2, pp. 1106-1110, 2015.

[13] D. T. Bauer, S. Guerlain and P. J. Brows, "The design and evaluation of a graphical display for laboratory data," Journal of the American Medical Informatics Association, vol. 17, no. 4, pp. 416-424, 2010

[14] Freecodecamp, “The Model View Controller Pattern – MVC Architecture and Frameworks Explained,” [Online]. Available: https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/. [Accessed 12 September 2023].