RANCANG BANGUN SISTEM PENCATATAN PENGUJIAN PERANGKAT LUNAK PADA DIVISI QUALITY ASSURANCE

KERJA PRAKTIK

Program Studi

S1 Sistem Informasi

Oleh:

SANDYKA DWI KURNIAWAN

22410100059

FAKULTAS TEKNOLOGI DAN INFORMATIKA

UNIVERSITAS DINAMIKA

2025

RANCANG BANGUN SISTEM PENCATATAN PENGUJIAN PERANGKAT LUNAK PADA DIVISI QUALITY ASSURANCE

Diajukan sebagai salah satu syarat untuk menyelesaikan

Program Sarjana

Disusun Oleh:

Nama : Sandyka Dwi Kurniawan

NIM : 22410100059

Program : S1 (Strata Satu)

Jurusan : Sistem Informasi

FAKULTAS TEKNOLOGI DAN INFORMATIKA

UNIVERSITAS DINAMIKA

2025

People come into your life for a reason, a season, or a lifetime. They're either a blessing or a lesson.

LEMBAR PENGESAHAN

Rancang Bangun Sistem Pencatatan Pengujian Perangkat Lunak Pada Divisi Quality Assurance

Laporan Kerja Praktik

oleh:

Sandyka Dwi Kurniawan

NIM. 22410100059

Telah diperiksa, diuji, dan disetujui

Surabaya, 10 Juli 2025

Disetujui

Dosen Pembimbing	Penyelia,
Tutut Wurijanto, M.Kom.	**Aminudin Nima**
NIDN. 0703056702	

Mengetahui,

Ketua Program Studi S1 Sistem Informasi

Digitally signed by Endra Rahmawati
Date: 2025.08.01 11:23:24 +07'00'

Endra Rahmawati, M.Kom.

NIDN. 0712108701

PERNYATAAN

PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH

Sebagai mahasiswa **Universitas Dinamika**, Saya:

Nama	: **Sandyka Dwi Kurniawan**
NIM	: **22410100059**
Program Studi	: **S1 Sistem Informasi**
Fakultas	: **Fakultas Teknologi dan Informatika**
Jenis Karya	: **Laporan Kerja Praktik**
Judul Karya	: **RANCANG BANGUN SISTEM PENCATATAN PENGUJIAN PERANGKAT LUNAK PADA DIVISI QUALITY ASSURANCE**

Menyatakan dengan sesungguhnya bahwa :

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, Saya menyetujui memberikan kepada **Universitas Dinamika** Hak Bebas Royalti Non-Eksklusif (***Non-Exclusive Royalty Free Right***) atas seluruh isi/sebagian karya ilmiah Saya tersebut diatas untuk disimpan, dialihmediakan, dan dikelola dalam bentuk pangkalan data (***database***) untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama Saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta.
2. Karya tersebut diatas adalah hasil karya asli Saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya, atau pendapat orang lain yang ada dalam karya ilmiah ini semata-mata hanya sebagai rujukan yang dicantumkan dalam Daftar Pustaka Saya.
3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiasi pada karya ilmiah ini, maka Saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada Saya.

Demikan surat pernyataan ini Saya buat dengan sebenar-benarnya.

Surabaya, 09 Juli 2025

Sandyka Dwi Kurniawan
NIM: 22410100059

ABSTRAK

Seiring meningkatnya kebutuhan akan perangkat lunak yang andal dan berkualitas, proses pengujian menjadi tahap penting dalam pengembangan sistem. PT. Sofco Graha sebagai perusahaan pengembang perangkat lunak menghadapi kendala karena belum memiliki sistem terpusat untuk pencatatan dan pemantauan proses pengujian. Hal ini menyulitkan pelacakan bug, monitoring progres, serta pelaporan kepada manajemen. Sebagai solusi, dirancang dan dibangun sebuah sistem informasi berbasis web bernama *Tester App*. Aplikasi ini ditujukan untuk mendukung tim Quality Assurance, Developer, dan Project Manager dalam mencatat aplikasi yang diuji, menyusun test case, mendokumentasikan bug, serta memantau progres pengujian secara sistematis. Sistem dikembangkan menggunakan metode Waterfall yang meliputi tahap analisis kebutuhan, perancangan, implementasi, pengujian, dan dokumentasi. Hasil akhir dari proyek ini adalah sistem pencatatan pengujian perangkat lunak yang dilengkapi fitur pengelolaan data master (menu, modul, aplikasi, developer), manajemen testing, serta laporan dan statistik untuk mendukung pengambilan keputusan. Sistem ini juga berfungsi sebagai dasar pengembangan lebih lanjut sebelum diimplementasikan secara penuh.

Kata Kunci: Pengujian perangkat lunak, Laravel, Quality Assurance, bug tracking, sistem informasi.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan laporan kerja praktik dengan judul *"Rancang Bangun Sistem Pencatatan Pengujian Perangkat Lunak pada Divisi Quality Assurance"* dengan baik dan tepat waktu.

Laporan ini disusun sebagai salah satu syarat dalam menyelesaikan program kerja praktik di Program Studi Sistem Informasi, Fakultas Teknologi dan Informatika, Universitas Dinamika. Dalam pelaksanaan kerja praktik ini, penulis banyak memperoleh pengalaman berharga, baik dari segi teknis maupun non-teknis, yang sangat berguna untuk pengembangan diri dan wawasan dalam dunia kerja, khususnya di bidang pengembangan perangkat lunak dan pengujian sistem informasi. Ucapan terima kasih penulis sampaikan kepada:

1. Bapak Tutut Wurijanto, M.Kom., selaku dosen pembimbing yang telah memberikan arahan dan bimbingan selama proses penyusunan laporan ini.
2. Ibu Endra Rahmawati, M.Kom., selaku Ketua Program Studi S1 Sistem Informasi Universitas Dinamika.
3. Ibu Pradita Maulidya Effendi, M.Kom., selaku dosen wali yang telah memberikan bimbingan selama proses Kerja Praktik
4. Seluruh staf dan karyawan PT. Sofco Graha, khususnya divisi Business Unit 4, yang telah memberikan kesempatan dan dukungan selama penulis menjalani kerja praktik.
5. Keluarga dan teman-teman yang selalu memberikan semangat, doa, dan dukungan moral selama proses kerja praktik berlangsung.

Penulis menyadari bahwa laporan ini masih memiliki kekurangan. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun demi perbaikan di masa mendatang. Semoga laporan ini dapat memberikan manfaat bagi pembaca serta menjadi referensi bagi mahasiswa lain yang akan melaksanakan kerja praktik di bidang yang serupa.

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB I

PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangan teknologi informasi yang semakin pesat, kebutuhan akan perangkat lunak *(software)* yang handal dan berkualitas menjadi sangat penting dalam mendukung operasional suatu perusahaan. Untuk mencapai hal tersebut, proses pengujian perangkat lunak menjadi salah satu tahapan penting dalam pengembangan sistem, agar produk yang dihasilkan dapat berjalan sesuai kebutuhan pengguna serta meminimalisir terjadinya kesalahan (*bug*) yang dapat mengganggu kinerja sistem.

PT. Sofco Graha merupakan perusahaan yang bergerak di bidang pengembangan perangkat lunak dan menyediakan berbagai layanan IT untuk mendukung kegiatan bisnis perusahaan-perusahaan besar di Indonesia. Dalam proses pengembangan aplikasi, tim *Quality Assurance (QA)* memiliki peranan penting dalam memastikan bahwa sistem yang dikembangkan telah diuji secara menyeluruh sebelum diserahkan kepada pengguna akhir. Namun, proses pencatatan pengujian dan pelaporan *bug* yang dilakukan secara manual atau melalui file terpisah masih kurang efektif dan menyulitkan dalam hal pemantauan serta dokumentasi progres pengujian.

Berdasarkan permasalahan tersebut, maka dibutuhkan sebuah sistem informasi yang dapat membantu tim *QA* dalam mencatat daftar aplikasi atau *website* yang sedang diuji, mendokumentasikan fitur atau menu beserta progres pengujiannya, serta mencatat *bug* yang ditemukan selama proses pengujian.

Dengan adanya sistem ini, proses pengujian dapat berjalan lebih efektif dan efisien, serta memudahkan dalam proses pelaporan dan evaluasi hasil pengujian perangkat lunak.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan maka dapat dirumuskan permasalahan adalah sebagai berikut:

1. Bagaimana merancang dan membangun sistem informasi yang dapat digunakan untuk mencatat daftar aplikasi atau *website* yang sedang diuji oleh tim *Quality Assurance*?
2. Bagaimana sistem dapat memfasilitasi pencatatan progres pengujian terhadap fitur atau menu dari aplikasi yang diuji?
3. Bagaimana sistem dapat mencatat dan mendokumentasikan temuan *bug* secara efektif selama proses pengujian berlangsung?
4. Bagaimana sistem dapat membantu meningkatkan efisiensi dan efektivitas proses pengujian serta mempermudah pelaporan hasil pengujian perangkat lunak?

1.3 Batasan Masalah

Agar pengembangan sistem lebih terfokus dan sesuai dengan kebutuhan, maka batasan masalah dalam proyek kerja praktik ini adalah sebagai berikut:

1. Sistem yang dibangun hanya digunakan oleh *Quality Assurance* (QA), *Project Manager*, dan *Developer* di PT. Sofco Graha untuk mencatat dan memantau aktivitas pengujian perangkat lunak.

2. Sistem hanya mencakup fitur pencatatan daftar aplikasi/website yang diuji, pencatatan fitur/menu yang diuji beserta status progresnya, serta pencatatan dan dokumentasi *bug* yang ditemukan.
3. Sistem tidak mencakup proses otomatisasi pengujian (*automated testing).*
4. Sistem dibangun menggunakan PHP dengan *framework* Laravel dan hanya diakses melalui *web browser*.
5. Pengguna sistem dibatasi hanya untuk *Admin*, *QA*, *Develope*r, dan *Project Manager*, tanpa melibatkan *end-user* dari luar tim internal.
6. Sistem ini tidak mencakup integrasi langsung dengan sistem pengembangan perangkat lunak lain.

1.4 Tujuan

1. Merancang dan membangun sistem informasi yang dapat digunakan untuk mencatat daftar aplikasi atau *website* yang sedang diuji oleh tim *Quality Assurance*.
2. Mengembangkan fitur pencatatan progres pengujian terhadap fitur atau menu dari masing-masing aplikasi yang diuji.
3. Menyediakan sarana pencatatan dan pemantauan proses pengujian perangkat lunak yang lebih terstruktur dan terdokumentasi, sehingga berpotensi mendukung proses pelaporan dan evaluasi hasil pengujian dengan lebih baik.

1.5 Manfaat

1. Memberikan kemudahan bagi tim *Quality Assurance* dalam mencatat dan mengelola daftar aplikasi atau *website* yang sedang diuji secara digital dan terpusat.

2. Membantu tim pengembang dan penguji dalam memantau *progress* pengujian fitur atau menu secara lebih sistematis.
3. Mendukung proses dokumentasi hasil pengujian yang rapi dan terdokumentasi, sehingga mempermudah proses evaluasi dan koordinasi antar tim yang terlibat.

BAB II

GAMBARAN UMUM PERUSAHAAN

2.1. Profil Perusahaan

PT Sofco Graha didirikan pada 1983 oleh Bapak Tan Kiong Hie. Seusai menyelesaikan studi S2 di Jerman, beliau bersama rekan-rekan alumnusnya menerapkan berbagai teknologi IT berbasis Komputerisasi Akuntansi dengan dukungan Bapak Tan Pei Ling (Pendiri Rodamas Group) di berbagai perusahaan Indonesia, seperti Rodamas Group, Asahimas Flat Glass, Asahimas Chemical, Konimex Group, Meratus Group, dan lainnya.

Pada 1986, PT Sofco Graha mulai melebarkan sayapnya dengan membuka kantor representatif di Surabaya, Jawa Timur. Dengan penggunaan sistem teknologi yang lebih modern & terbarukan, seperti *S/36, AS/400*, dan *Clipper*. Akhir 80-an hingga 2000-an mulai merambah menjadi berbasis *Windows, Apps Web*, hingga membuat berbagai program dengan *framework-framework* terbaru.

Pada 2012, PT Sofco Graha mulai menjadi bagian dari PT Konimex. Dari sinilah mulai berkembang teknologi yang nyaman digunakan oleh berbagai user & sesuai dengan kondisi di lapangan. Hingga pada 2016, Sofco mulai meluncurkan sebuah aplikasi HR berbasis online yang dapat dinikmati oleh seluruh kalangan dengan sistem berlangganan. Selain itu, kami juga bekerjasama dengan *Odoo* sebagai partner kolaborasi penyedia *software* yang handal & *implementor* berpengalaman, agar dapat selalu memberikan yang terbaik bagi para *customer* &

calon *customer*. Berikut ini adalah logo resmi dari PT Sofco Graha yang dapat dilihat di Gambar 2.1:

Gambar 2. 1 Logo PT. Sofco Graha

2.1. Identitas Perusahaan

Identitas perusahaan yang diinformasikan meliputi nama perusahaan, alamat, nomor telepon, serta email. Adapun rincian identitasnya adalah sebagai berikut.

Nama Instansi	:	PT. Sofco Graha
Alamat	:	Jl. Berbek Industri III No. 17, Berbek, Waru, Sidoarjo, Jawa Timur 61256
Nomor telepon	:	(031) 842 0861
Email	:	sofco_sb@sofcograha.co.id

2.2. Visi Perusahaan

Menjadi perusahaan IT terkemuka yang dapat memberikan solusi bagi konsumen di Indonesia dan *Regional*

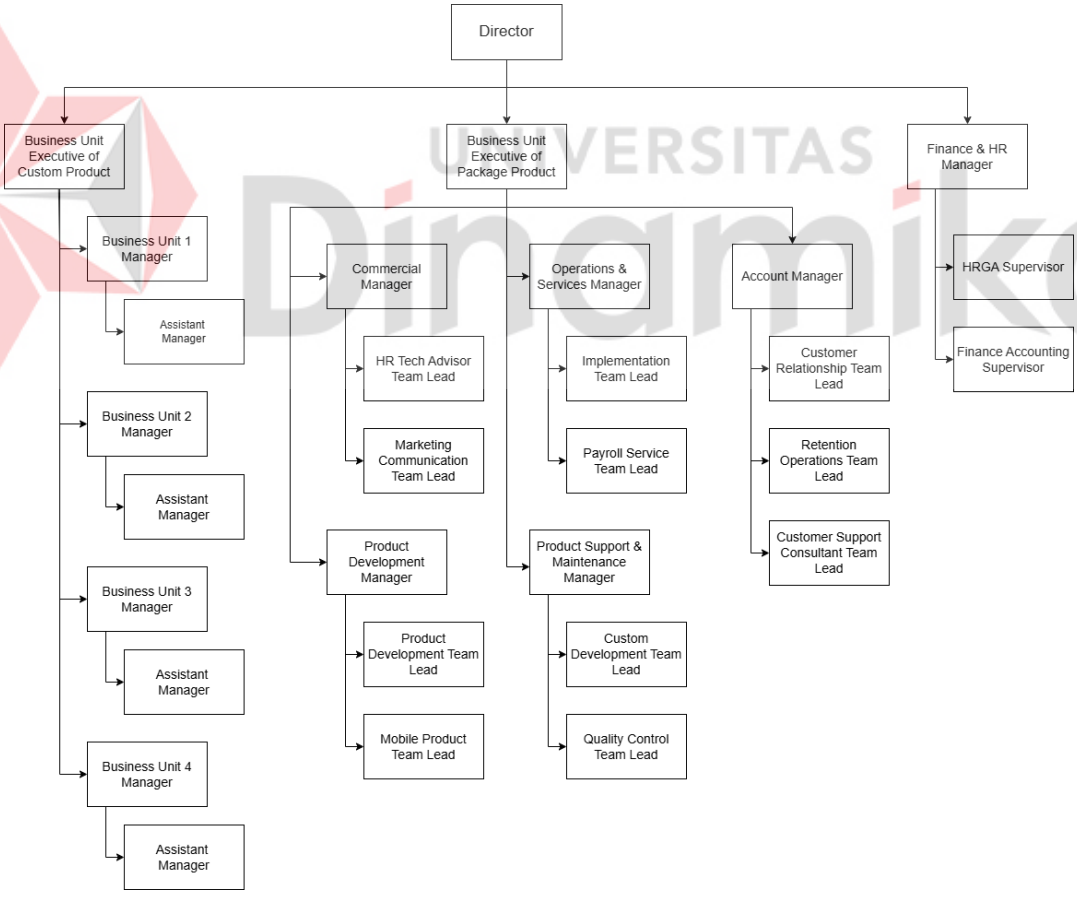
2.3. Misi Perusahaan

1. Membangun SDM yang bertanggung jawab, bermutu, dan dapat dipercaya

2. Mengikuti kondisi pasar dan lapangan untuk selali mengetahui kebutuhan konsumen terkini
3. Menghasilkan produk yang berkualitas dan selalu mengikuti teknologi terkini
4. Membangun relasi yang baik dengan pelanggan atas dasar kepercayaan
5. Membuka peluang bermitra dengan pihak lain untuk memperkuat penetrasi pasar

2.4. Struktur Organisasi

Adapun struktur organisasi di PT.SOFCO GRAHA dapat dilihat pada Gambar 2.1 :



Gambar 2. 2 Struktur Organisasi PT. Sofco Graha

BAB III

LANDASAN TEORI

3.1 Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah proses penting dalam siklus pengembangan sistem, yang bertujuan untuk mengevaluasi sejauh mana sistem memenuhi kebutuhan dan harapan pengguna (Dhandy et al., 2025). Pengujian perangkat lunak memiliki cakupan yang lebih luas dari sekadar mencari *bug*, yaitu bertujuan untuk meningkatkan kualitas *software* secara menyeluruh (Hidayat et al., 2025).

Terdapat beberapa jenis pengujian dalam application testing, salah satunya adalah *blackbox testing*. Metode *Black Box Testing* adalah teknik pengujian yang ditujukan untuk mengidentifikasi kesalahan dalam sistem aplikasi, seperti kekeliruan pada fungsi sistem dan hilangnya menu aplikasi (Hidayat et al., 2025). Pengujian ini umumnya dilakukan oleh tim *Quality Assurance* (QA) untuk memastikan bahwa sistem berjalan sesuai dengan kebutuhan pengguna dan spesifikasi yang telah ditentukan. *Blackbox testing* mencakup beberapa tahapan, seperti system testing dan *acceptance testing*, yang bertujuan untuk menguji keseluruhan sistem sebagai satu kesatuan serta mengevaluasi apakah aplikasi dapat diterima oleh pengguna akhir.

3.2 *Software Development Life Cycle*

Software Development Life Cycle (SDLC) adalah suatu kerangka kerja sistematis yang digunakan dalam pengembangan perangkat lunak untuk memastikan bahwa proses pembuatan sistem dilakukan secara terstruktur,

terencana, dan efisien. Metode *SDLC Waterfall* Merupakan salah satu metode yang mempunyai ciri khas bahwa pengerjaan setiap fase harus dikerjakan terlebih dahulu sebelum melanjutkan ke fase berikutnya (Nugraha et al., 2018).

Tujuan utama dari penerapan SDLC adalah untuk meminimalisasi risiko kesalahan dalam pengembangan sistem serta menghindari pemborosan waktu dan sumber daya. Dengan mengikuti alur SDLC, tim pengembang dapat memahami dan merancang sistem berdasarkan kebutuhan yang jelas, membangun solusi yang terukur, serta mengevaluasi hasilnya secara menyeluruh sebelum digunakan oleh pengguna akhir. Model SDLC memiliki berbagai pendekatan, seperti *Waterfall, Agile, Iterative, dan Spiral*, yang dapat dipilih sesuai dengan karakteristik proyek dan kebijakan organisasi.

3.3 Laravel

Laravel adalah salah satu *framework* PHP berbasis *Model-View-Controller (MVC)* yang bersifat open-source dan dirancang untuk mempermudah pengembangan aplikasi web dengan sintaks yang elegan dan ekspresif. Laravel diciptakan untuk membuat pengalaman bekerja dengan aplikasi menjadi lebih baik bagi pengembang melalui sintaks yang ekspresif, gamblang, dan efisien (Anggarah et al., 2025).

3.4 PHP

PHP adalah bahasa pemrograman yang digunakan untuk membangun aplikasi web dinamis. PHP berjalan di sisi *server* dan dapat digunakan untuk memproses data, berinteraksi dengan database, serta menghasilkan halaman web

secara otomatis. PHP dikenal fleksibel dan mudah dipelajari, sehingga banyak digunakan dalam pengembangan aplikasi web.

Framework Laravel adalah salah satu *framework* PHP modern yang memudahkan pengembangan aplikasi dengan struktur yang rapi, aman, dan mudah dirawat. Laravel mendukung konsep *MVC (Model-View-Controller)* yang membantu pengelolaan kode menjadi lebih terstruktur dan efisien. MVC memisahkan tanggung jawab antara logika aplikasi, tampilan, dan kontrol, sehingga tim pengembang dapat bekerja secara paralel tanpa mengganggu satu sama lain (Anggarah et al., 2025).

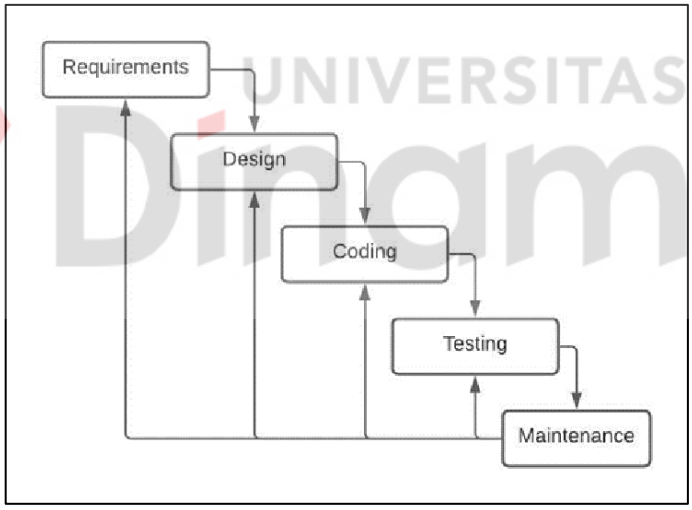
3.5 MySQL

MySQL adalah sistem manajemen basis data relasional yang digunakan untuk menyimpan dan mengelola data dalam aplikasi. PHP dan MySQL digunakan sebagai bahasa pemrograman dan sistem manajemen basis data, yang merupakan pilihan populer dalam pengembangan aplikasi berbasis web karena fleksibilitas dan kemampuannya dalam menangani volume data yang besar serta menyediakan antarmuka yang *user-friendly* (Supriadi et al., 2025).

Dalam sistem pencatatan pengujian perangkat lunak, MySQL digunakan untuk menyimpan data seperti daftar aplikasi yang diuji, fitur-fitur yang telah melalui pengujian, status *progress*, dan dokumentasi bug. MySQL dipilih karena stabil, cepat, dan cocok digunakan dalam proyek pengembangan aplikasi web skala kecil hingga menengah.

3.6 Metode Waterfall

Metode *waterfall* adalah sebuah metode pengembangan sistem dimana antar satu fase ke fase yang lain dilakukan secara berurutan (Fachri & Wahyu Surbakti, 2021). Kelebihan metode ini terletak pada dokumentasi yang lengkap dan alur kerja yang jelas, sehingga cocok diterapkan pada proyek-proyek dengan kebutuhan yang stabil dan tidak banyak berubah. Model *waterfall* ini menyediakan pendekatan alur hidup perangkat lunak secara sekuensial terurut dimulai dari analisis, desain, pengkodean, pengujian dan tahap pendukung (*support*) (Supiyandi et al., 2022). Gambar metode Waterfall dapat dilihat pada Gambar 3.1:



Gambar 3. 1 Gambar Metode Waterfall

Pada kerja praktik ini, perusahaan tempat pelaksanaan menggunakan metode *Waterfall* dalam proses pengembangan sistem pencatatan pengujian perangkat lunak. Pemilihan metode ini dilakukan agar pengembangan berjalan lebih terkontrol, sesuai dengan alur kerja formal yang telah diterapkan di

perusahaan, serta memudahkan dalam proses evaluasi dan pelaporan hasil kerja setiap tahap.

BAB IV

DESKRIPSI PEKERJAAN

4.1 Analisis Permasalahan

Analisis permasalahan pada proyek ini dilakukan sebagai tahap awal untuk merancang sistem pencatatan pengujian perangkat lunak. Peneliti bertemu langsung dengan pemangku kepentingan yakni divisi *Quality Assurance*, *Project Manager*, dan tim *Developer* di PT. Sofco Graha untuk mengidentifikasi kendala yang selama ini dialami dalam proses pencatatan dan pelaporan hasil pengujian. Saat ini, seluruh aktivitas pengujian masih bergantung pada catatan manual maupun dokumen terpisah di komputer masing-masing, sehingga sulit untuk mendapatkan gambaran menyeluruh tentang progres pengujian fitur atau menu aplikasi. Akibatnya, *Project Manager* dan *Developer* sering kali terlambat memperoleh informasi terbaru, yang berdampak pada lambatnya penanganan *bug* dan koordinasi antar tim menjadi kurang efektif. Selain itu, belum adanya sistem terpusat berbasis web membuat akses terhadap data pengujian menjadi terbatas pada perangkat dan lokasi tertentu, padahal fleksibilitas akses melalui *browser* akan sangat mendukung kolaborasi lintas divisi.

Berdasarkan identifikasi masalah tersebut mulai dari pencatatan manual dan tersebar, kurangnya visibilitas progres, dokumentasi *bug* yang tidak terstruktur, hingga risiko kehilangan data dan ketiadaan aplikasi terpusat maka diperlukan pengembangan Sistem Pencatatan Pengujian Perangkat Lunak Berbasis Laravel. Sistem ini diharapkan mampu menyediakan antarmuka tunggal untuk mencatat setiap aplikasi atau *website* yang diuji, memonitor progres

pengujian fitur/menu, serta mendokumentasikan temuan *bug* dengan format baku dan mekanisme update status secara *real-time*, sehingga proses pengujian menjadi lebih efektif, efisien, dan terstruktur.

4.2 Analisis Sistem yang Sedang Berjalan

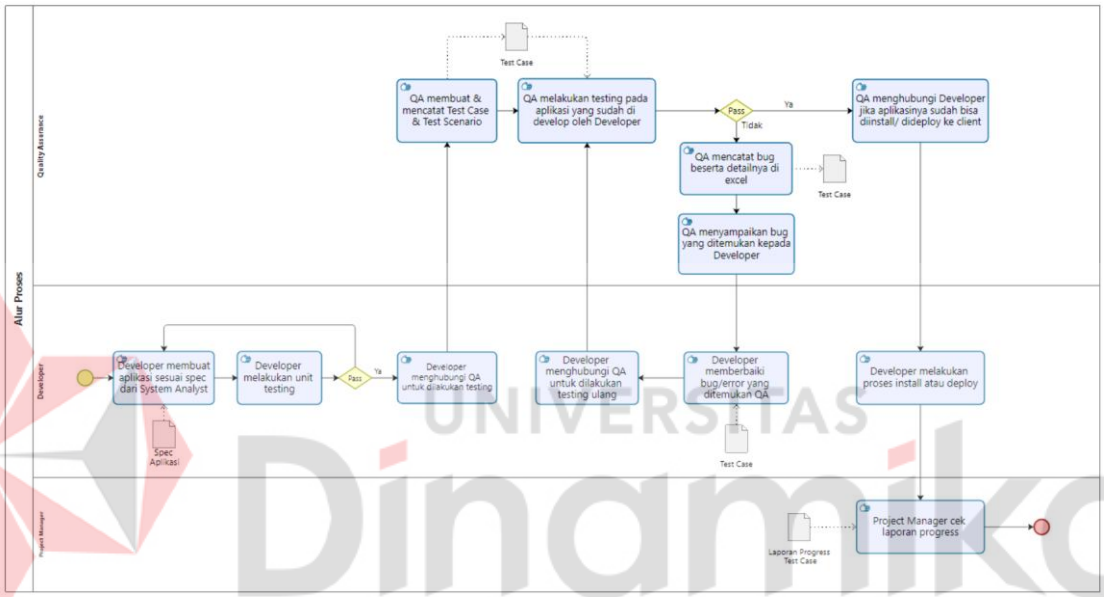
Sistem pencatatan pengujian perangkat lunak yang saat ini berjalan di PT. Sofco Graha masih dilakukan secara manual dan belum terpusat. Berdasarkan pengamatan dan analisis terhadap proses yang berlangsung, alur kerja pengujian melibatkan tiga peran utama, yaitu *Developer, Quality Assurance (QA)*, dan *Project Manager. Developer* bertanggung jawab untuk membuat aplikasi dan melakukan *unit testing* secara mandiri. Jika hasil *unit testing* dinyatakan lolos, *Developer* akan menghubungi *QA* untuk melanjutkan ke tahap pengujian sistem.

Pada sisi *QA*, proses dimulai dari pembuatan dan pencatatan *test case* serta *test scenario*. *QA* kemudian melakukan pengujian terhadap aplikasi yang telah diserahkan oleh *Developer*. Jika dalam proses ini ditemukan *bug*, *QA* akan mencatat detailnya ke dalam file Excel dan menyampaikannya kembali kepada *Developer* untuk diperbaiki. Siklus ini akan terus berulang hingga aplikasi dianggap bebas dari bug. Setelah itu, *QA* akan menginformasikan kepada *Developer* bahwa aplikasi sudah layak untuk dilakukan instalasi atau *deploy* ke *client*.

Developer kemudian melapor kepada *Project Manager* untuk meminta konfirmasi sebelum proses *deploy* dilakukan. Setelah mendapat persetujuan, *Developer* akan melakukan proses instalasi aplikasi ke lingkungan *client*.

Dari analisis ini, dapat disimpulkan bahwa proses yang berjalan masih memiliki beberapa kelemahan, seperti ketergantungan pada pencatatan manual

(Excel), kurangnya pelacakan progress secara *real-time*, tidak adanya sistem terpusat untuk dokumentasi *bug* dan status pengujian, serta tidak adanya visualisasi dalam bentuk laporan maupun statistik, sehingga pihak *project manager* kesulitan dalam memantau progres pengujian secara menyeluruh. Pada Gambar 4.1 dibawah ini adalah proses yang saat ini sedang berjalan di PT. Sofco Graha.



Gambar 4. 1 Proses Bisnis Sofco Graha (Manual)

4.3 Perancangan Sistem

4.3.1 Tujuan Perancangan Sistem

Tujuan dari perancangan sistem ini adalah untuk mengatasi permasalahan yang ada pada sistem manual sebelumnya. Sistem yang dirancang bertujuan untuk mempermudah pencatatan aktivitas pengujian perangkat lunak, meningkatkan transparansi progres fitur atau menu yang diuji, serta mempercepat proses pelaporan dan perbaikan *bug*. Dengan adanya sistem ini, diharapkan kolaborasi

antara *QA*, *Developer*, dan *Project Manager* menjadi lebih terstruktur, efisien, dan terdokumentasi dengan baik.

4.3.2 Gambaran Umum Sistem Yang Diusulkan

Sistem yang diusulkan adalah sebuah aplikasi berbasis web bernama *Tester App* yang dirancang untuk mempermudah dan mengefisienkan proses pengujian perangkat lunak di PT. Sofco Graha. Aplikasi ini mengintegrasikan peran *Developer, Quality Assurance (QA),* dan *Project Manager* dalam satu sistem terpusat, mulai dari proses *unit testing*, pencatatan *test case*, pelaporan *bug*, hingga *monitoring* status pengujian.

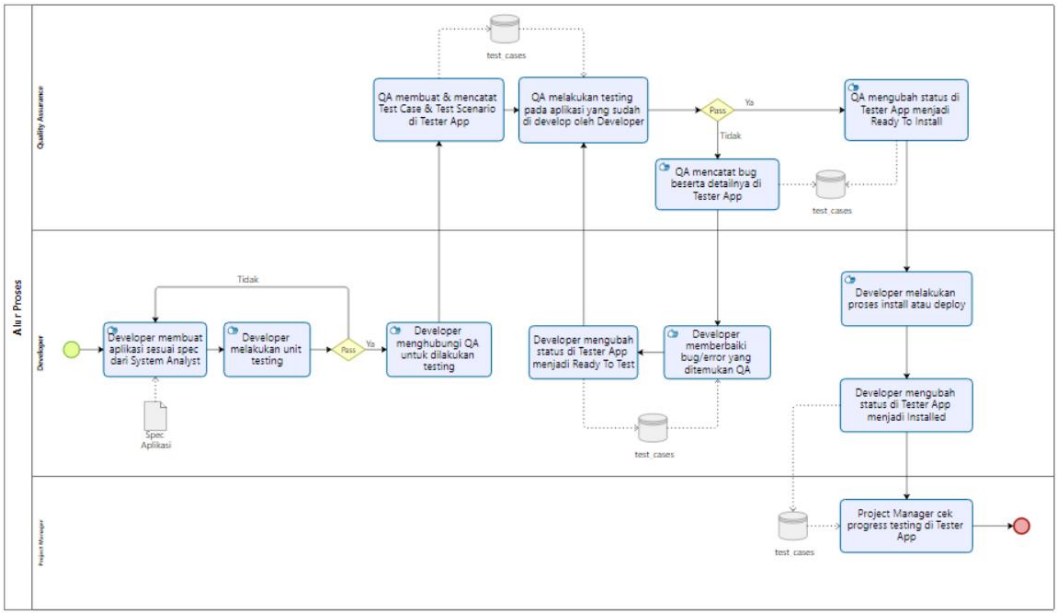
4.3.2.1 Perancangan Prosedur Sistem Yang Diusulkan

Sistem yang diusulkan dirancang untuk mengatasi permasalahan yang ada pada proses pengujian perangkat lunak di PT. Sofco Graha, yang sebelumnya masih dilakukan secara manual dan belum terintegrasi. Sistem ini melibatkan tiga peran utama, yaitu *Developer*, *Quality Assurance (QA)*, dan *Project Manager*, yang masing-masing memiliki akses dan tugas yang spesifik dalam proses pengujian. Proses dimulai ketika *Developer* membuat atau memperbaiki aplikasi berdasarkan spesifikasi dari *System Analyst*, kemudian melakukan *unit testing* secara mandiri. Jika *unit testing* berhasil, *Developer* menghubungi *QA* untuk melanjutkan pengujian sistem, serta mengubah status aplikasi di sistem menjadi "*Ready to Test*".

Pada sisi *QA*, proses dimulai dengan pembuatan dan pencatatan *test case* dan test scenario langsung di dalam aplikasi Tester App. QA kemudian melakukan pengujian terhadap aplikasi yang telah dikembangkan. Jika ditemukan

bug atau error, *QA* mencatat detailnya langsung di sistem, sehingga informasi dapat segera diterima dan ditindaklanjuti oleh *Developer*. Developer kemudian melakukan perbaikan terhadap *bug* yang ditemukan, lalu siklus pengujian berlanjut kembali hingga aplikasi dinyatakan lolos uji. Jika aplikasi telah melewati semua tahap pengujian tanpa error, *QA* akan mengubah status aplikasi menjadi *"Ready to Install"*.

Setelah status dinyatakan siap untuk instalasi, *Developer* akan melanjutkan proses deploy aplikasi ke *server client* dan mengubah status aplikasi menjadi "Installed" di dalam sistem. Seluruh proses ini terekam secara otomatis di sistem, termasuk pencatatan *test case*, status bug, dan progres pengujian secara keseluruhan. *Project Manager* dapat secara langsung memantau hasil pengujian melalui fitur Laporan & Statistik yang tersedia di *Tester App*, tanpa perlu meminta laporan manual dari *QA* maupun *Developer*. Fitur ini menampilkan data visual seperti grafik jumlah *bug*, status pengujian per fitur, serta riwayat aktivitas pengujian dalam bentuk yang mudah dipahami. Gambar 4.2 dibawah ini adalah Diagram proses bisnisyang diusulkan pada PT.Sofco Graha.

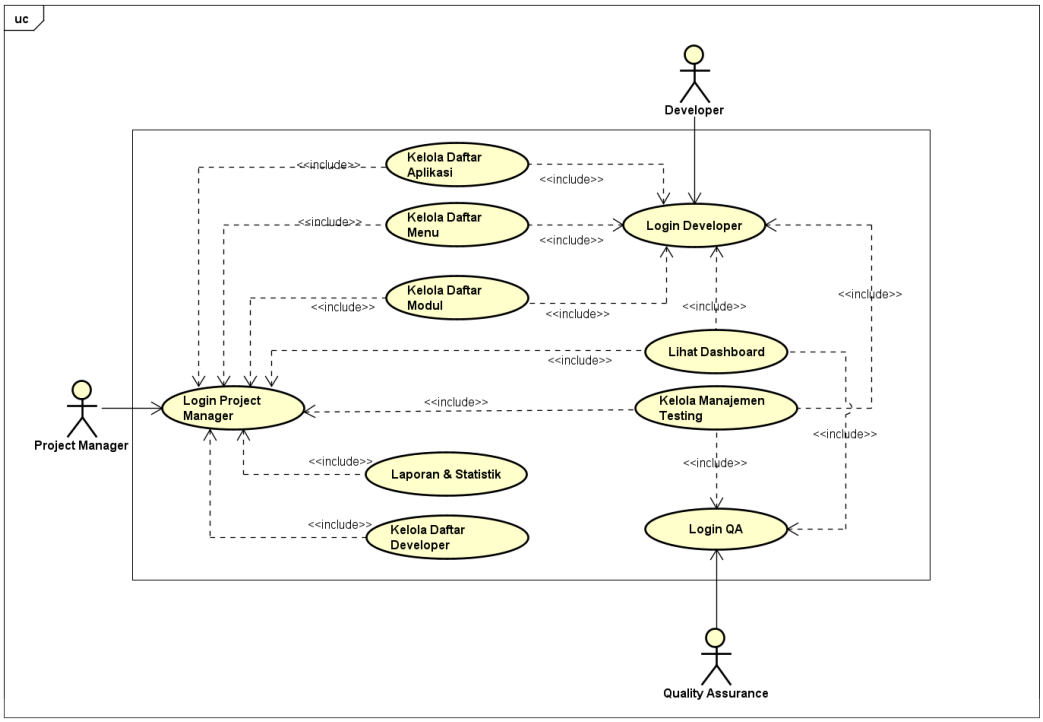


Gambar 4. 2 Proses Bisnis Sofco Graha (Sistem)

Dengan sistem yang terintegrasi dan berbasis digital ini, proses pengujian menjadi lebih efisien, transparan, dan mudah dipantau. Ketergantungan terhadap pencatatan manual seperti Excel dapat dihilangkan, progres pengujian dapat dilacak secara *real-time*, dan manajemen proyek dapat melakukan evaluasi terhadap performa pengujian dengan lebih akurat dan efektif.

A. Use Case Diagram

Berikut adalah use case diagram dari *Tester App*:



Gambar 4. 3 Use Case Tester App

Pada Gambar 4.3 diatas menggambarkan interaksi antara tiga aktor utama (*Developer, Quality Assurance, dan Project Manager*) dengan sistem *Tester App*. Setiap aktor memiliki peran dan tanggung jawab spesifik dalam proses pengujian aplikasi, yang dijelaskan sebagai berikut:

A. Developer

1. Login Developer

Developer masuk ke sistem menggunakan akun yang telah diberikan untuk mengakses fitur-fitur yang berkaitan dengan pengelolaan aplikasi dan pengujian.

2. Kelola Daftar Aplikasi

Developer dapat menambahkan, memperbarui, atau menghapus daftar aplikasi yang akan diuji. Ini penting agar *QA* dapat mengetahui aplikasi mana yang akan diuji.

3. Kelola Daftar Modul

Developer mencatat pembagian modul dari masing-masing aplikasi untuk mempermudah identifikasi struktur sistem saat pengujian dilakukan.

4. Kelola Daftar Menu

Developer mengelola fitur atau menu yang terdapat dalam setiap modul aplikasi, yang nantinya menjadi acuan bagi *QA* dalam membuat *test case* dan *test scenario.*

5. Lihat Dashboard

Developer dapat melihat ringkasan *progress* pengujian, termasuk status *bug, test case* yang sudah dieksekusi, dan informasi penting lainnya secara *real-time.*

B. Quality Assurance (QA)

1. Login QA

QA melakukan login ke sistem untuk mulai bekerja dalam proses pengujian perangkat lunak yang dikembangkan oleh *Developer*.

2. Kelola Manajemen Testing

QA mengelola data pengujian seperti pencatatan *test case*, *test scenario*, serta hasil pengujian (*pass*/*fail*). Jika ditemukan *bug*, *QA* dapat mencatat dan memantau status perbaikannya.

3. Lihat Dashboard

Sama seperti *Developer, QA* juga dapat melihat *dashboard* untuk mengetahui *progress* pengujian aplikasi, *bug* yang sedang ditangani, serta dokumentasi pengujian lainnya.

C. Project Manager

1. Login Project Manager

Project Manager mengakses sistem dengan akun khusus untuk memantau seluruh proses pengujian yang sedang berjalan.

2. Kelola Daftar Developer

Project Manager memiliki hak akses untuk mengelola informasi pengguna *Developer* dalam sistem, termasuk menambah atau mengatur hak aksesnya.

3. Laporan & Statistik

Project Manager dapat melihat visualisasi data berupa grafik dan laporan hasil pengujian yang mencakup jumlah *bug*, status pengujian setiap aplikasi, serta progres kinerja *QA* dan *Developer.*

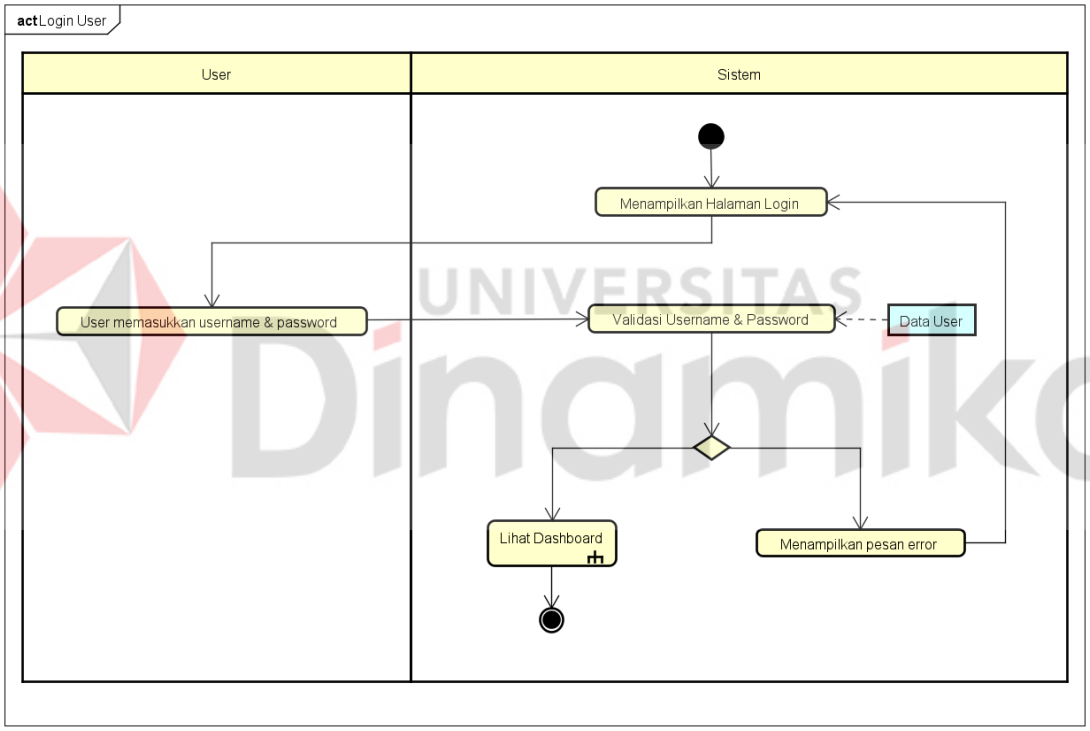
4. Lihat Dashboard

Selain laporan statistik, *Project Manager* juga bisa memantau aktivitas sistem secara *real-time* melalui *dashboard*, sehingga memudahkan pengambilan keputusan dan pengawasan.

B. Activity Diagram

Activity Diagram menggambarkan alur aktivitas dalam sistem pengujian perangkat lunak (*Tester App*) yang dilakukan oleh tiga aktor utama: *Developer*, *Quality Assurance (QA)*, dan *Project Manager*. Diagram ini menvisualisasikan proses bisnis dari awal pengujian hingga aplikasi dinyatakan siap untuk di-deploy. Berikut ini adalah beberapa *Activity Diagram* yang ada pada *Tester App*:

B.1. Login User

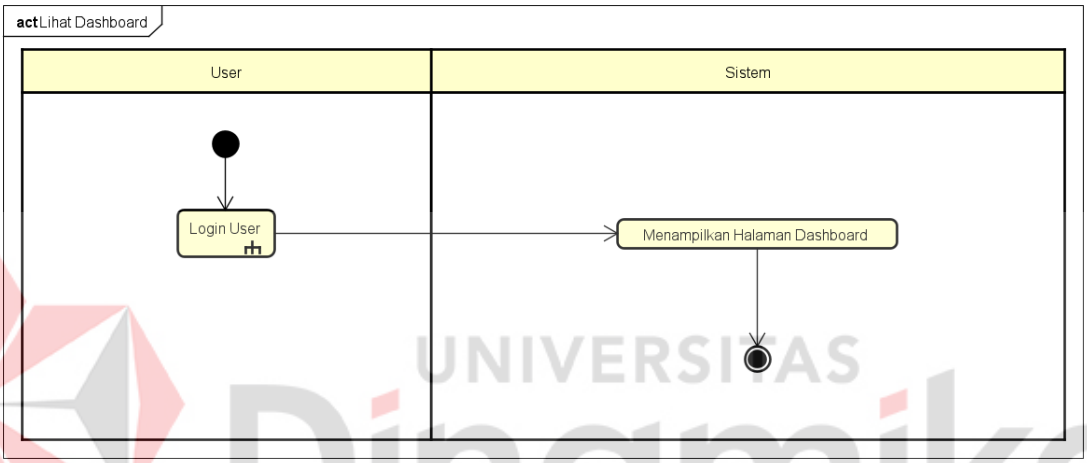


Gambar 4. 4 Activity Diagram Login User

Pada Gambar 4.4 diatas menggambarkan alur proses ketika user melakukan login ke dalam sistem. Proses diawali ketika sistem menampilkan halaman login, lalu user memasukkan *username* dan *password* yang dimilikinya. Data tersebut kemudian dikirim ke sistem untuk dilakukan proses validasi terhadap data pengguna (user) yang tersimpan di basis data. Setelah proses

validasi dilakukan, terdapat dua kemungkinan hasil. Jika data yang dimasukkan sesuai (valid), maka sistem akan mengarahkan user menuju halaman *dashboard* sebagai tampilan awal setelah *login* berhasil. Namun, jika data tidak valid (*username* atau *password* salah), maka sistem akan menampilkan pesan error yang memberi tahu bahwa *login* gagal.

B.2. Lihat Dashboard



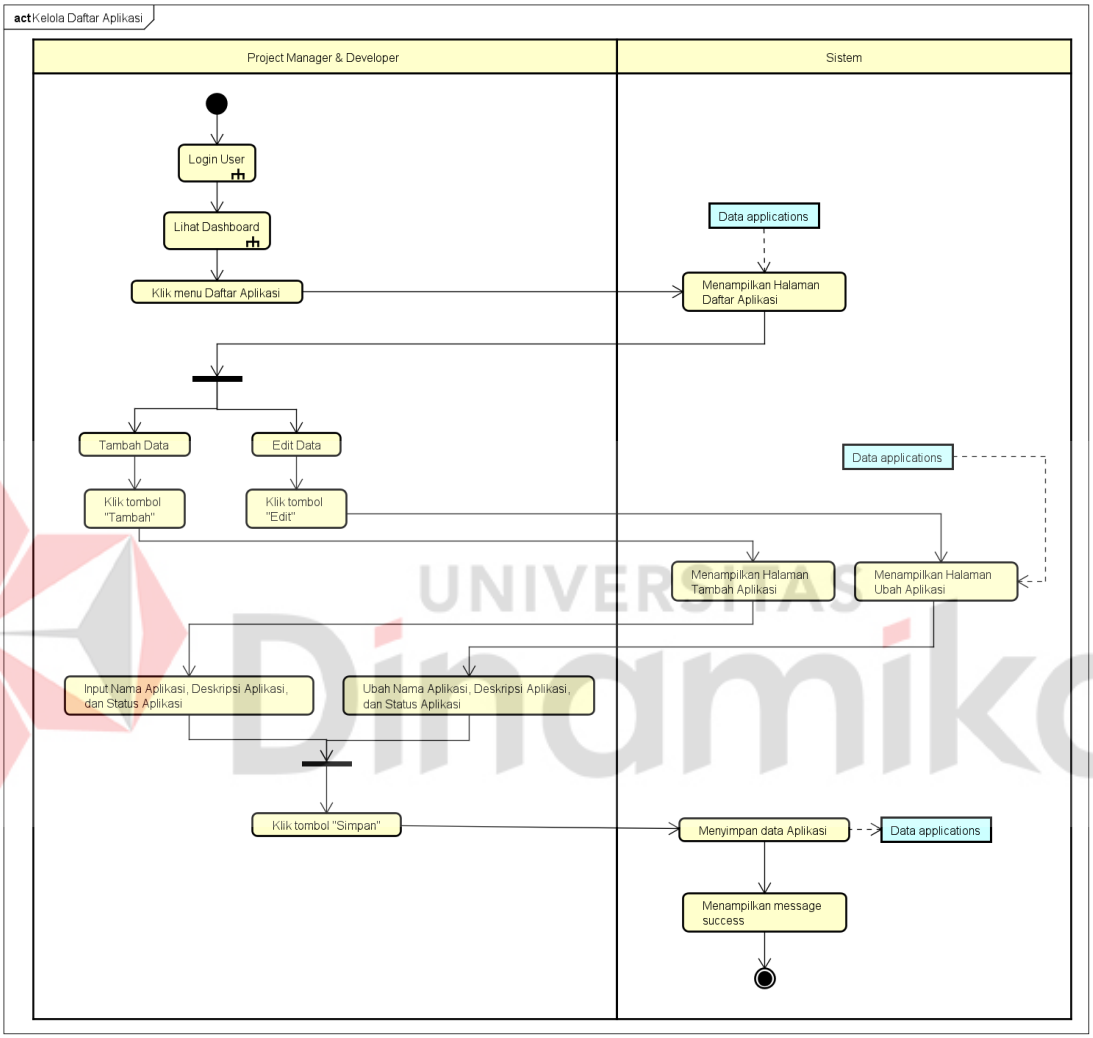
Gambar 4. 5 Activity Diagram Lihat Dashboard

Pada Gambar 4.5 diatas menggambarkan proses ketika user melihat halaman *dashboard* setelah berhasil *login* ke dalam sistem. Proses dimulai dari aksi *Login User*, yang menjadi prasyarat agar user dapat mengakses menu dashboard. Setelah *login* berhasil, sistem secara otomatis akan menampilkan halaman *dashboard* sesuai dengan hak akses pengguna.

Dashboard ini berfungsi sebagai halaman utama yang memberikan gambaran umum mengenai status dan aktivitas dalam sistem, seperti ringkasan jumlah *bug*, progres pengujian, dan informasi lainnya yang relevan dengan peran user. Aktivitas ini sangat penting karena memungkinkan pengguna baik itu

Developer, QA, maupun *Project Manager* untuk langsung mendapatkan informasi terbaru secara cepat dan efisien begitu mereka masuk ke sistem.

B.3. Kelola Daftar Aplikasi

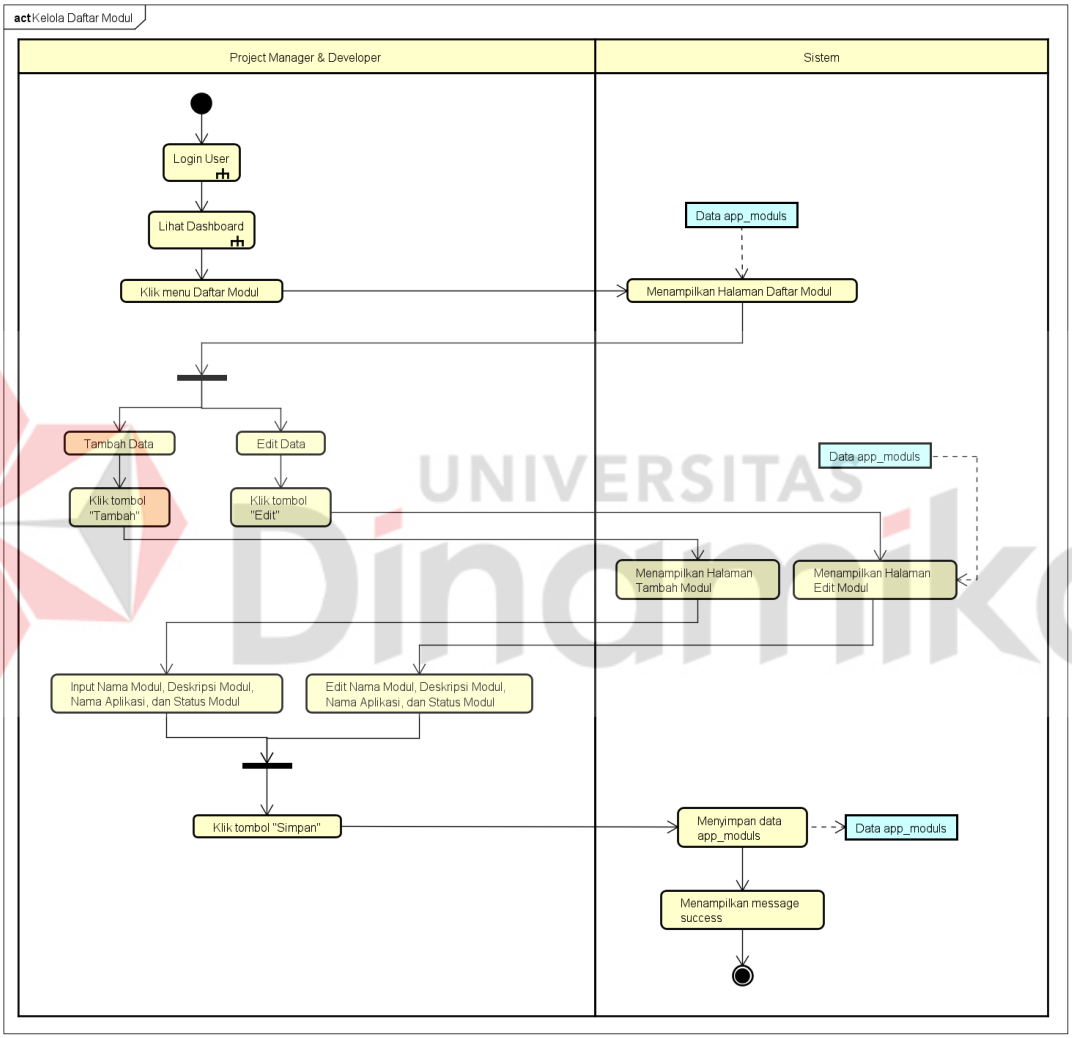


Gambar 4. 6 Activity Diagram Kelola Daftar Aplikasi

Pada Gambar 4.6 diatas menggambarkan proses yang dilakukan oleh *Project Manager* atau *Developer* untuk menambah atau mengubah data aplikasi di sistem. Proses dimulai dari login, masuk ke *dashboard*, lalu memilih menu Daftar Aplikasi. Sistem akan menampilkan halaman yang berisi daftar aplikasi. Pengguna dapat memilih untuk menambahkan aplikasi baru dengan mengisi

nama, deskripsi, dan status aplikasi, atau melakukan pengeditan terhadap aplikasi yang sudah ada. Setelah data diisi atau diubah, pengguna menekan tombol Simpan, dan sistem akan menyimpan data ke database serta menampilkan pesan sukses.

B.4. Kelola Daftar Modul

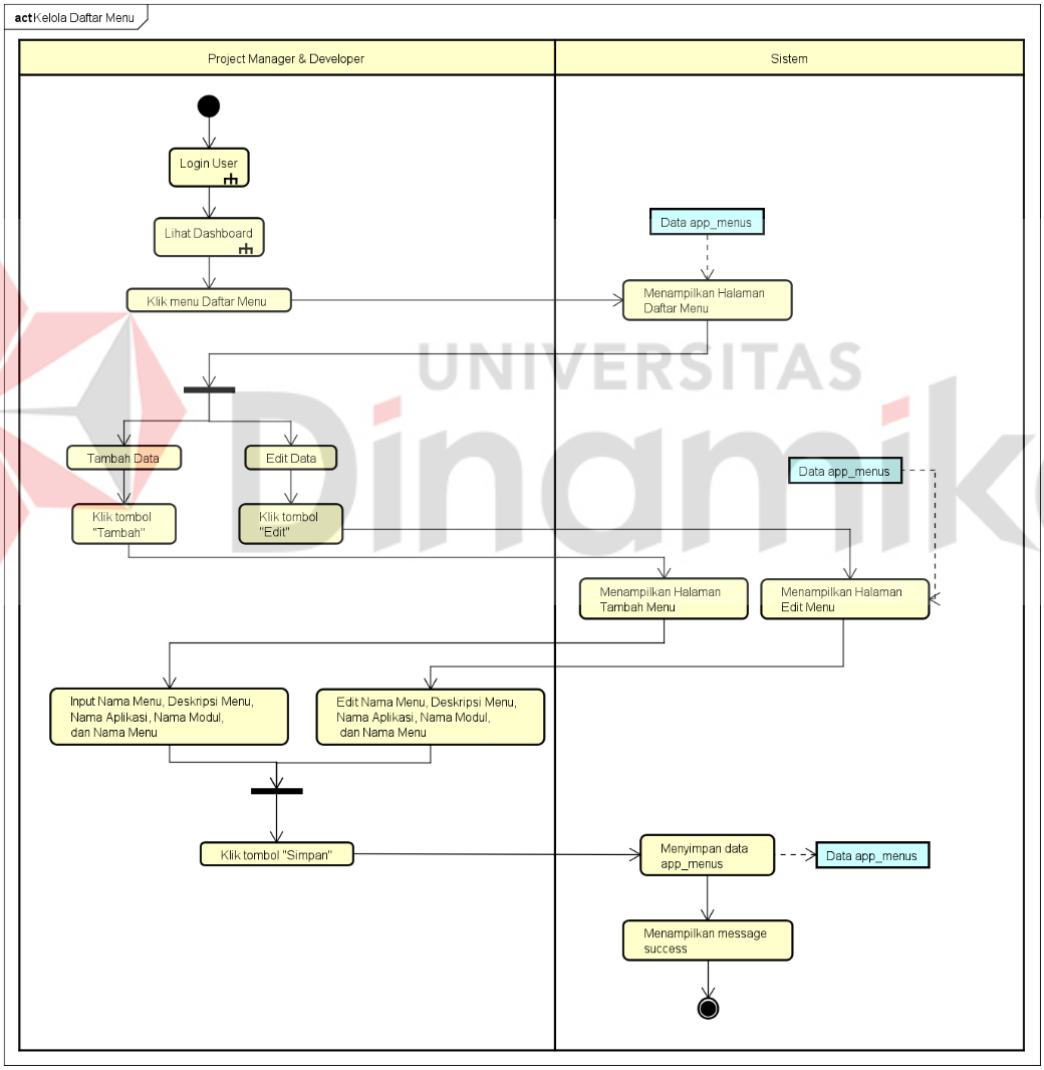


Gambar 4. 7 Activity Diagram Kelola Daftar Modul

Pada Gambar 4.7 menjelaskan proses pengelolaan data modul oleh *Project Manager* atau *Developer*. Proses dimulai dari login, melihat *dashboard*, lalu memilih menu Daftar Modul. Sistem akan menampilkan daftar modul yang sudah ada. Pengguna dapat memilih untuk menambah modul baru atau mengedit

modul yang sudah ada. Untuk menambah, pengguna mengisi nama modul, deskripsi, nama aplikasi terkait, dan status modul. Jika mengedit, pengguna memperbarui data yang sama. Setelah selesai, pengguna menekan tombol Simpan, dan sistem akan menyimpan data ke dalam *database* serta menampilkan pesan sukses.

B.5. Kelola Daftar Menu

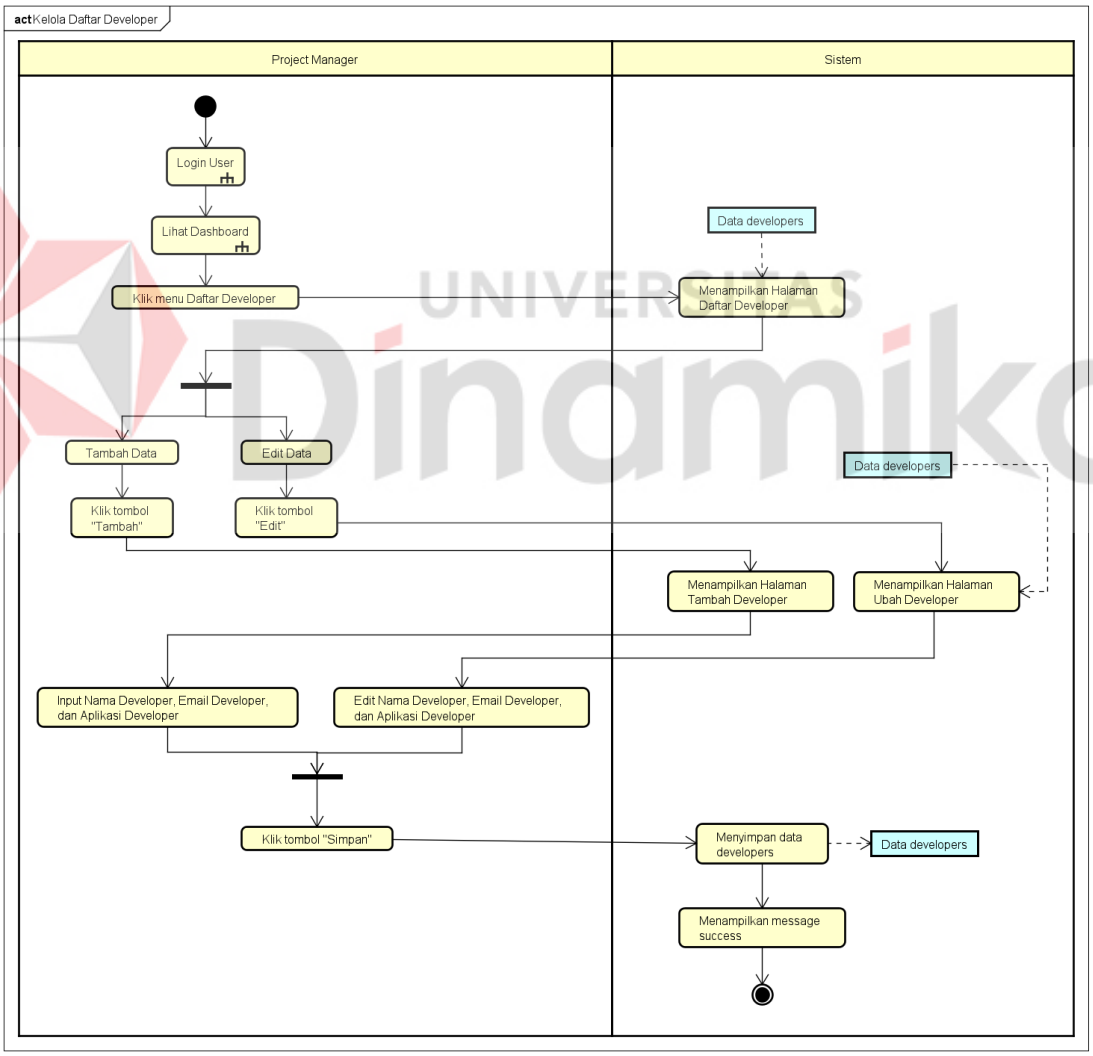


Gambar 4. 8 Activity Diagram Kelola Daftar Menu

Pada Gambar 4.8 ini menjelaskan alur kerja *Project Manager* atau *Developer* dalam mengelola data menu pada sistem. Setelah *login* dan masuk ke

dashboard, pengguna memilih menu Daftar Menu, lalu sistem menampilkan data menu yang ada. Pengguna dapat menambah atau mengubah data menu. Untuk menambah, pengguna mengisi nama menu, deskripsi, nama aplikasi, dan nama modul. Untuk mengedit, pengguna cukup memperbarui data yang sudah ada. Setelah mengisi, pengguna menekan tombol Simpan, dan sistem akan menyimpan data ke database serta menampilkan pesan sukses.

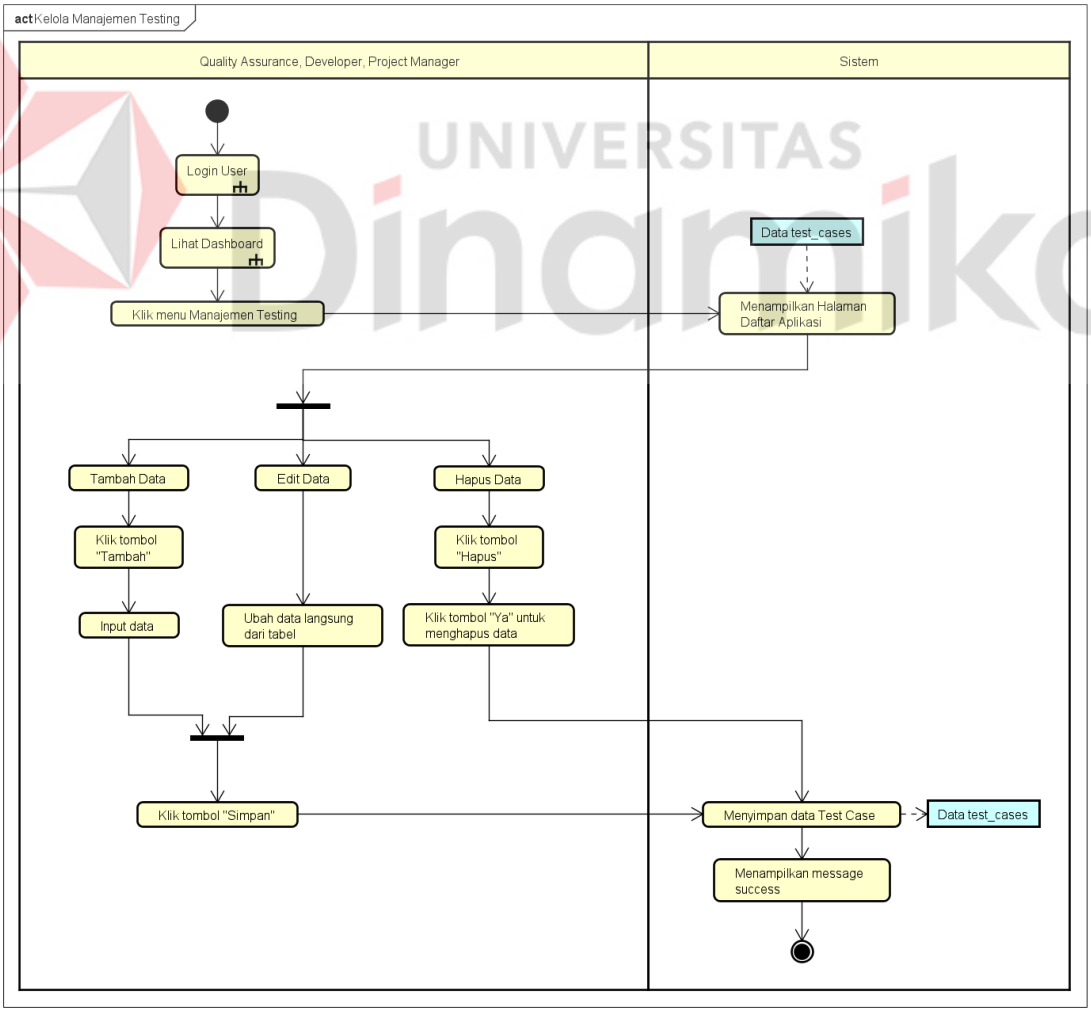
B.6. Kelola Daftar Developer



Gambar 4. 9 Activity Diagram Kelola Daftar Developer

Pada Gambar 4.9 ini menggambarkan proses yang dilakukan oleh *Project Manager* untuk mengelola data *developer* dalam sistem. Proses dimulai dari login, masuk ke *dashboard*, lalu memilih menu Daftar *Developer*. Sistem akan menampilkan daftar *developer* yang sudah ada. Pengguna dapat memilih untuk menambahkan data baru atau mengedit data yang sudah ada. *Input* yang diperlukan meliputi nama *developer*, *email*, dan aplikasi yang ditangani. Setelah menekan tombol Simpan, sistem menyimpan data ke *database* dan menampilkan pesan sukses.

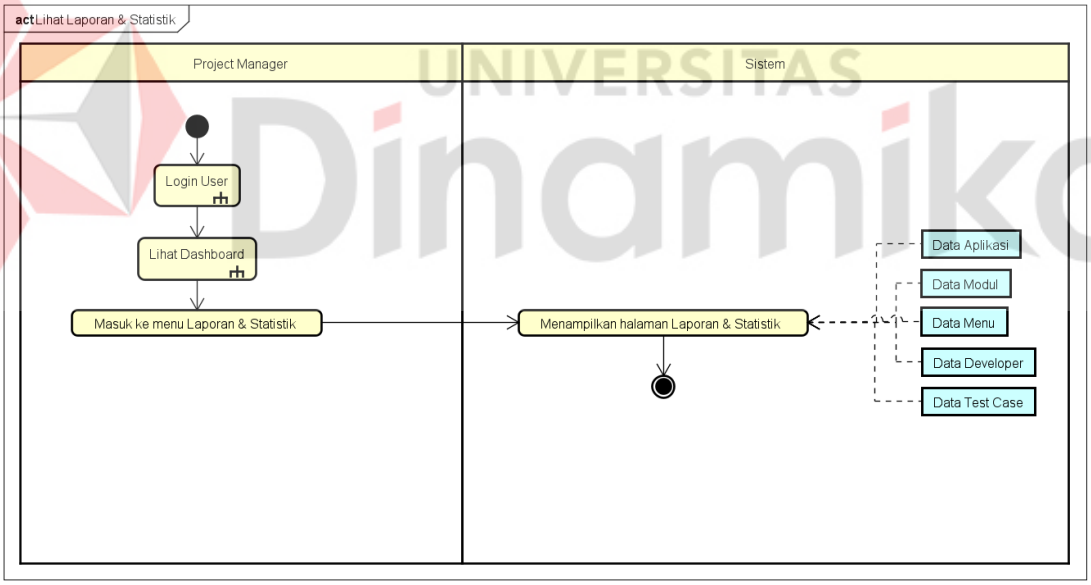
B.7. Manajemen Testing



Gambar 4. 10 Activity Diagram Manajemen Testing

Pada Gambar 4.10 menunjukkan alur aktivitas *Quality Assurance (QA), Developer*, dan *Project Manager* dalam mengelola data test case di sistem. Setelah *login* dan masuk ke *dashboard*, pengguna memilih menu Manajemen *Testing* untuk melihat atau mengelola data pengujian. Pengguna dapat melakukan tiga aksi utama: menambah, mengubah, atau menghapus data *test case*. Penambahan dilakukan dengan mengisi data baru, untuk mengubah data bisa langsung dari tabel, dan penghapusan membutuhkan konfirmasi. Setelah itu, pengguna menekan tombol Simpan, dan sistem akan menyimpan data serta menampilkan pesan sukses.

B.8. Lihat Laporan & Statistik



Gambar 4. 11 Activity Diagram Lihat Laporan & Statistik

Pada Gambar 4.11 diatas menggambarkan alur *Project Manager* saat mengakses informasi laporan dan statistik melalui sistem. Setelah login dan masuk ke *dashboard*, *Project Manager* memilih menu Laporan & Statistik.

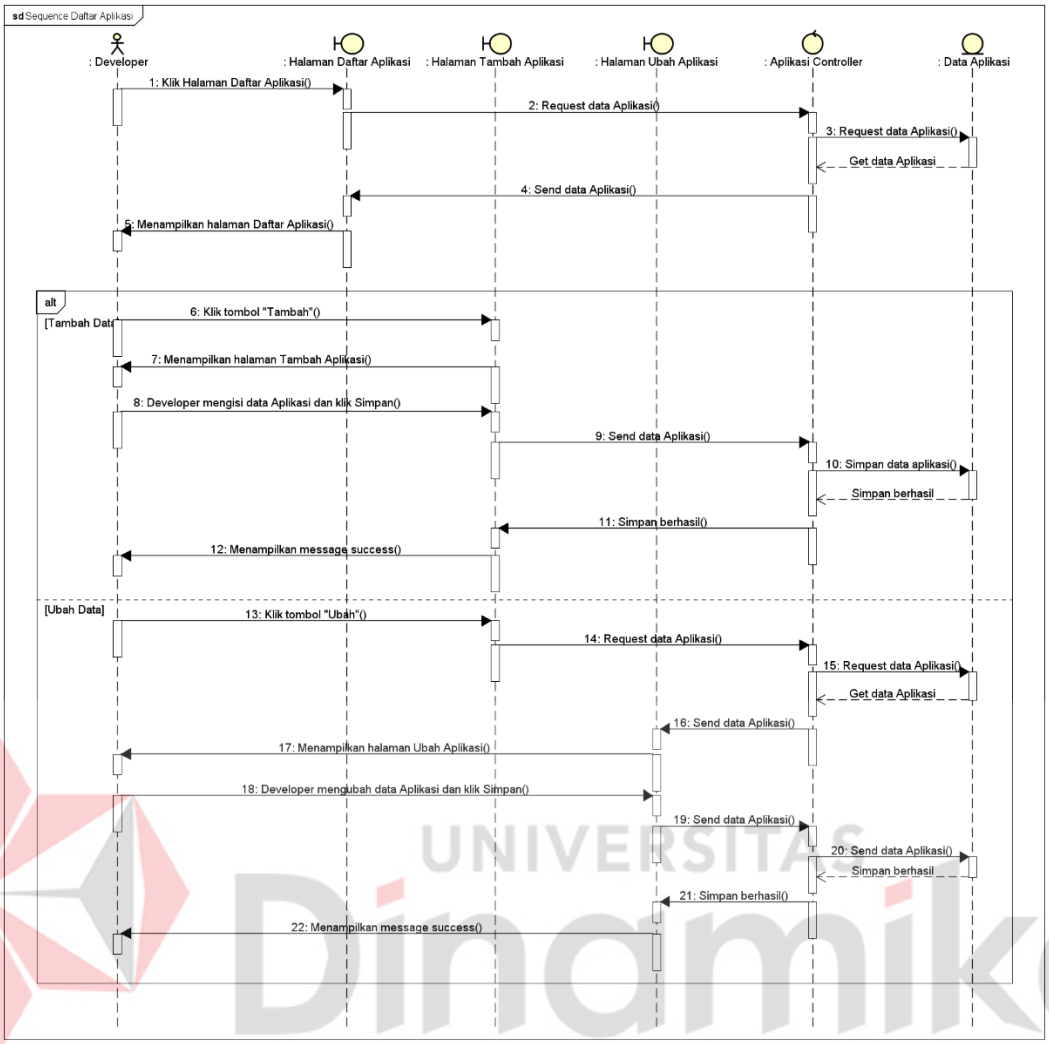
Sistem kemudian menampilkan halaman yang berisi visualisasi dan rekap data dari berbagai entitas seperti aplikasi, modul, menu, *developer*, dan *test case*.

C. Sequence Diagram

Sequence Diagram menggambarkan interaksi antar objek dalam sistem *Tester App* berdasarkan urutan waktu. Diagram ini menunjukkan aliran pesan antar komponen saat menjalankan suatu fitur atau proses tertentu, seperti saat pengguna mengelola daftar aplikasi, modul, menu, *developer*, maupun ketika melakukan manajemen *testing*. Berikut ini adalah beberapa *Sequence Diagram* yang ada pada *Tester App*:

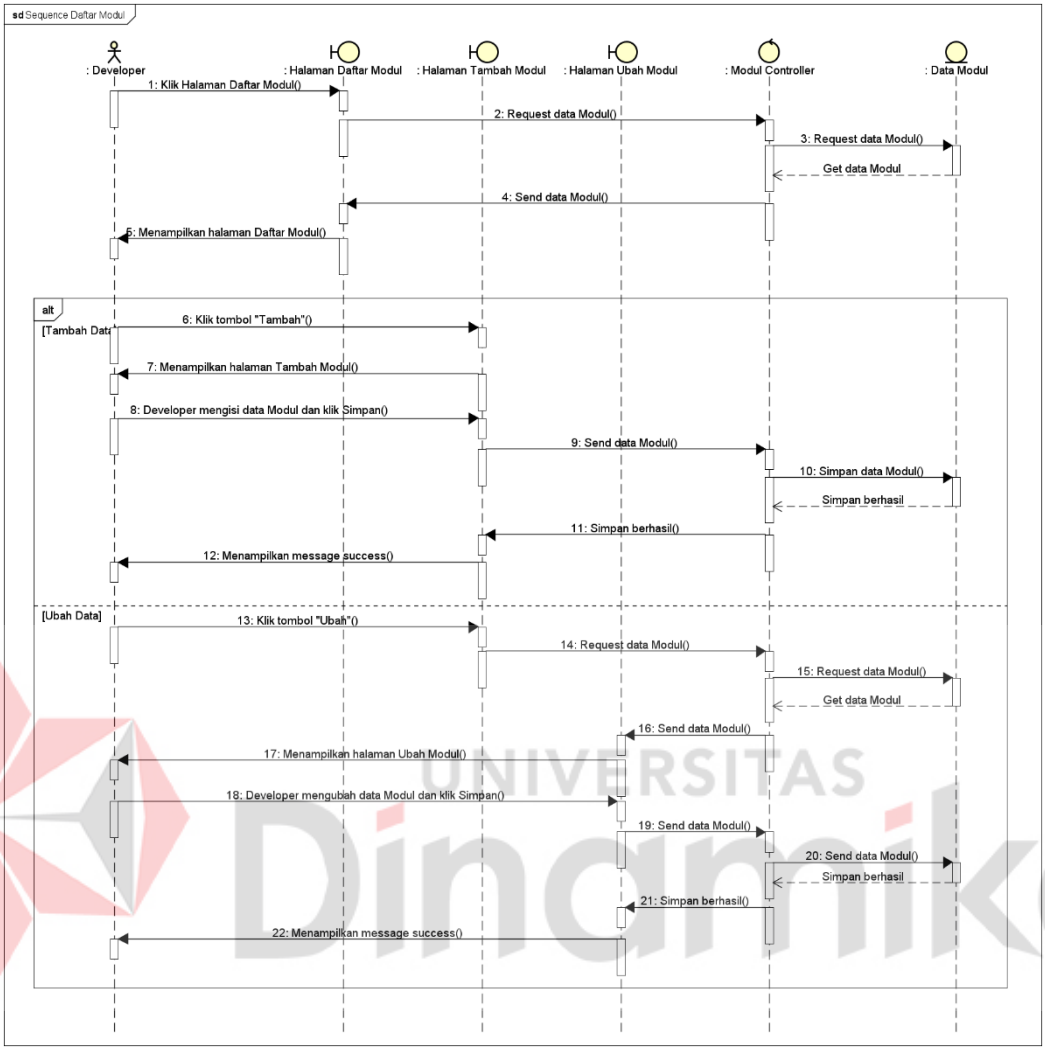
C.1. Kelola Daftar Aplikasi

Pada Gambar 4.12 diatas adalah *Sequence diagram* Daftar Aplikasi yang menggambarkan alur interaksi antara *Developer* dengan sistem saat menambahkan atau mengubah data aplikasi. Proses dimulai ketika *Developer* mengakses halaman daftar aplikasi, lalu sistem menampilkan data yang diambil dari *database*. Jika *Developer* menekan tombol "Tambah", sistem menampilkan form *input* dan setelah data diisi, sistem menyimpan data ke *database* dan menampilkan pesan sukses. Untuk proses pengubahan data, *Developer* memilih tombol "Ubah", sistem menampilkan data aplikasi yang dipilih, kemudian *Developer* dapat mengubah dan menyimpannya. Setelah data berhasil diperbarui, sistem menampilkan notifikasi bahwa proses berhasil. Diagram ini menunjukkan bagaimana komunikasi antar komponen sistem dilakukan secara berurutan dan terstruktur.



Gambar 4. 12 Sequence Diagram Kelola Daftar Aplikasi

C.2. Kelola Daftar Modul

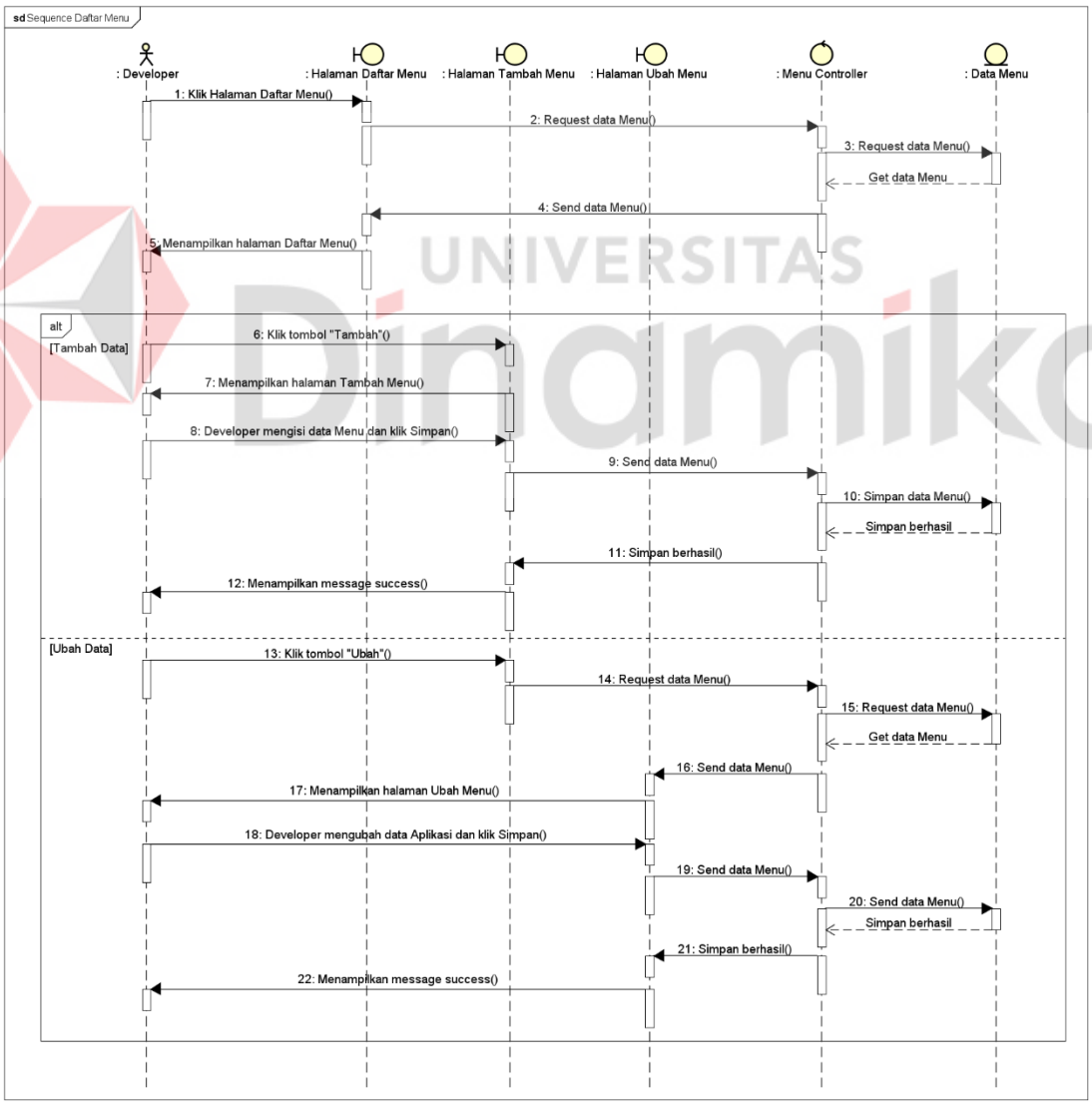


Gambar 4. 13 Sequence Diagram Kelola Daftar Modul

Pada Gambar 4.13 diatas adalah Sequence diagram Daftar Modul yang menjelaskan proses interaksi antara Developer dan sistem saat melakukan pengelolaan data modul. Proses dimulai ketika *Developer* mengakses halaman daftar modul dan sistem menampilkan data yang diambil dari *database*. Untuk menambah data, *Developer* menekan tombol “Tambah”, lalu mengisi form pada halaman tambah modul dan menekan tombol “Simpan”. Data yang diinput dikirim ke *controller* untuk disimpan ke dalam *database*, dan jika berhasil, sistem

akan menampilkan pesan sukses. Sedangkan untuk mengubah data, *Developer* menekan tombol “Ubah” pada modul yang ingin diedit, sistem memuat data terkait, lalu *Developer* melakukan perubahan dan menyimpannya. Sistem kemudian memperbarui data di *database* dan memberikan notifikasi bahwa proses berhasil. Diagram ini menggambarkan alur komunikasi yang runtut dan logis dalam pengelolaan modul aplikasi.

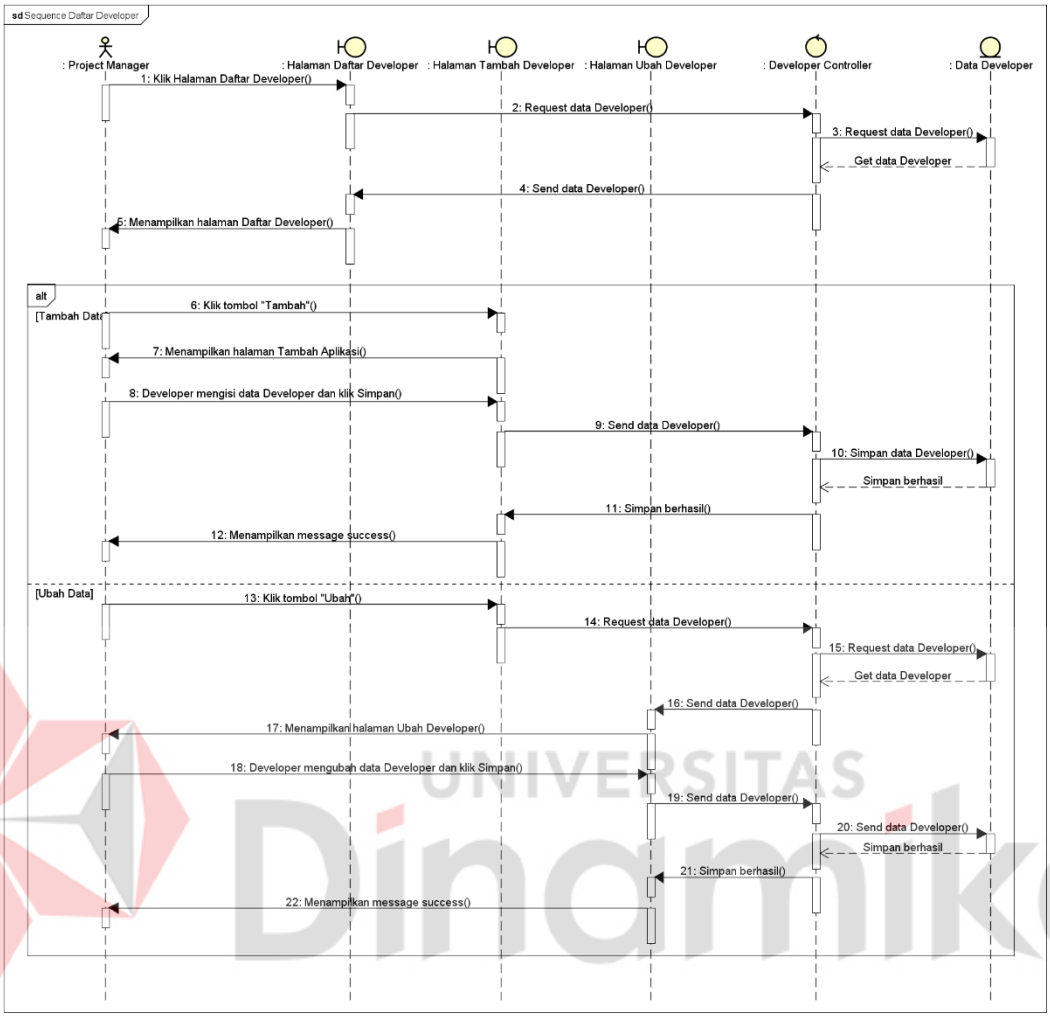
C.3. Kelola Daftar Menu



Gambar 4. 14 Sequence Diagram Kelola Daftar Menu

Pada Gambar 4.14 diatas adalah *Sequence diagram* Daftar Menu yang menggambarkan alur proses yang dilakukan oleh *Developer* saat menambahkan atau mengubah data menu dalam sistem. Proses dimulai ketika *Developer* mengakses halaman daftar menu, lalu sistem mengambil dan menampilkan data menu dari *database*. Jika *Developer* menekan tombol "Tambah", sistem akan menampilkan halaman form tambah menu. Setelah *Developer* mengisi data dan mengklik tombol Simpan, data dikirim ke *controller* untuk disimpan ke dalam *database*. Jika penyimpanan berhasil, sistem memberikan pesan sukses. Pada proses pengubahan data, *Developer* memilih tombol "Ubah" pada menu tertentu. Sistem mengambil data menu yang bersangkutan dan menampilkannya di halaman ubah menu. *Developer* kemudian mengedit data dan menyimpannya. Setelah proses penyimpanan berhasil, sistem kembali memberikan notifikasi sukses. Diagram ini menunjukkan komunikasi antar objek dalam sistem secara runtut saat melakukan operasi *CRUD* pada entitas menu.

C.4. Kelola Daftar Developer

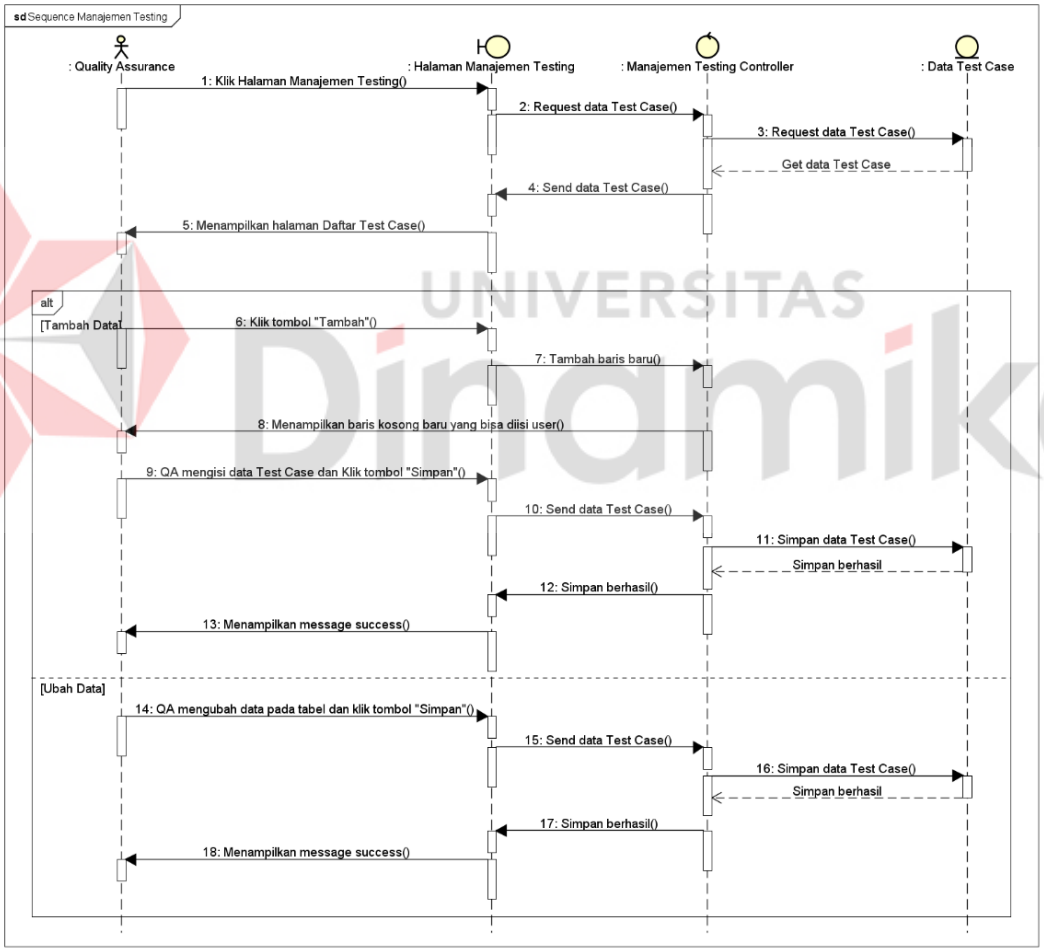


Gambar 4. 15 Sequence Diagram Kelola Daftar Developer

Pada Gambar 4.15 diatas adalah *Sequence diagram* Daftar Developer yang menggambarkan alur proses yang dilakukan oleh *Project Manager* dalam mengelola data *Developer*. Proses dimulai ketika *Project Manager* membuka halaman daftar *developer*, kemudian sistem menampilkan data yang diambil dari *database*. Untuk menambah data, *Project Manager* menekan tombol “Tambah”, lalu sistem menampilkan halaman tambah *developer*. Setelah mengisi *form* dan mengklik tombol *Simpan*, data dikirim ke *controller* dan disimpan ke *database*. Jika penyimpanan berhasil, sistem menampilkan pesan sukses. Sedangkan untuk

mengubah data, *Project Manager* memilih tombol “Ubah” pada *developer* yang ingin diubah. Sistem menampilkan data yang bersangkutan, lalu *Project Manager* melakukan perubahan dan menyimpannya. Data yang diperbarui dikirim kembali ke sistem dan disimpan, lalu sistem kembali menampilkan notifikasi bahwa proses berhasil. Diagram ini menjelaskan interaksi terstruktur dalam proses penambahan dan pengubahan data *developer* oleh *Project Manager*.

C.5. Manajemen Testing

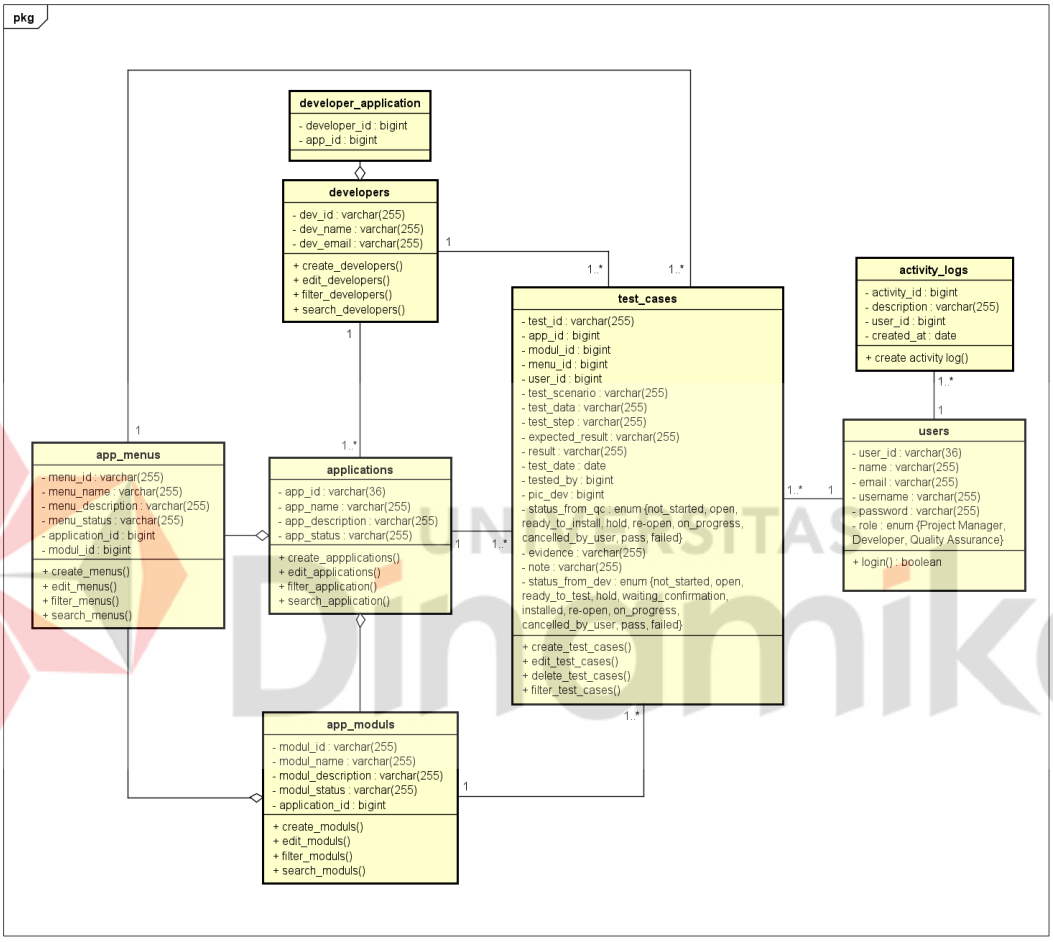


Gambar 4. 16 Sequence Diagram Manajemen Testing

Pada Gambar 4.16 diatas adalah *Sequence diagram* Manajemen *Testing* yang menggambarkan alur interaksi antara *Quality Assurance (QA)* dengan sistem

saat melakukan pengelolaan data *test case*. Proses dimulai saat *QA* mengakses halaman Manajemen Testing, kemudian sistem mengambil dan menampilkan data *test case* yang tersimpan di database.

D. Class Diagram



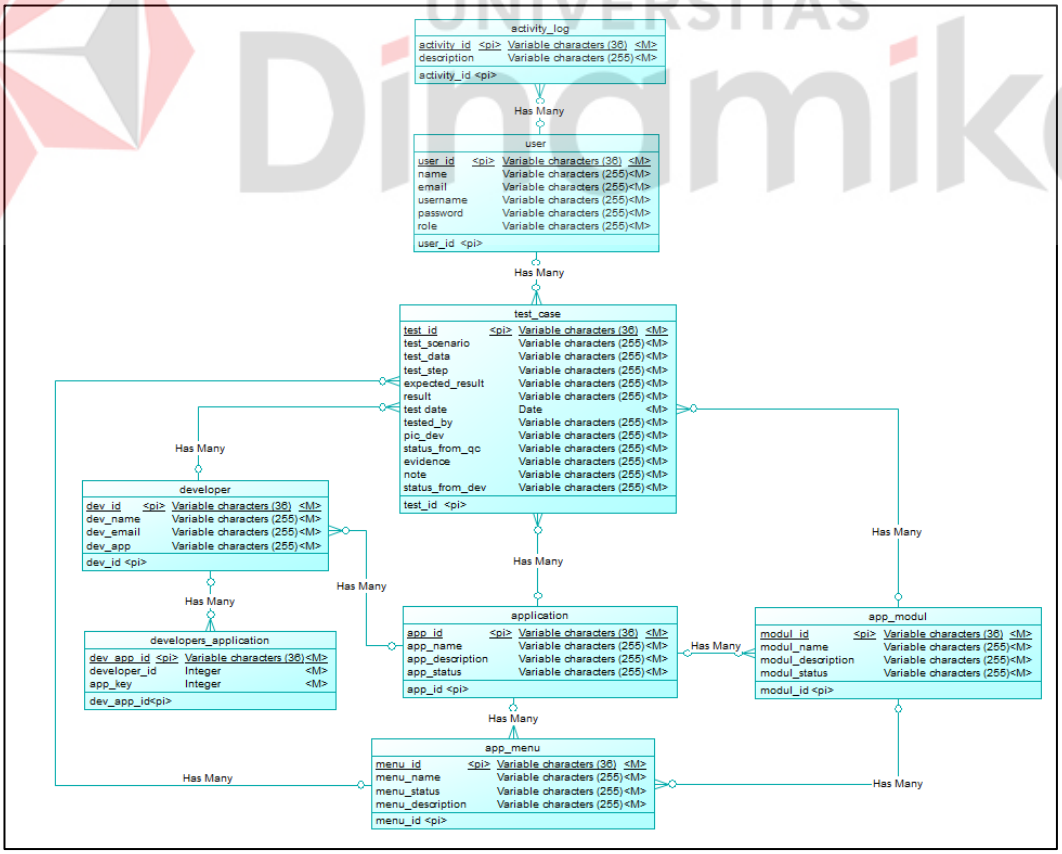
Gambar 4. 17 Class Diagram Tester App

Pada Gambar 4.17 diatas adalah *Class diagram* sistem *Tester App* yang menggambarkan struktur logis dan relasi antar entitas yang digunakan dalam proses pengujian perangkat lunak di PT. Sofco Graha. Diagram ini terdiri dari delapan entitas utama: *users, applications, app_moduls, app_menus, test_cases, developers, developer_application,* dan *activity_logs.*

Entitas users menyimpan data pengguna beserta perannya (*Project Manager, Developer, atau Quality Assurance*) dan terhubung dengan *activity_logs* untuk mencatat aktivitas pengguna. Entitas *applications* merepresentasikan aplikasi yang diuji, yang memiliki relasi ke *app_moduls* dan *app_menus* sebagai bagian dari struktur aplikasi.

Entitas *test_cases* menjadi pusat pencatatan skenario pengujian, mencakup data seperti langkah pengujian, hasil yang diharapkan, hasil aktual, serta status dari *QA* dan *Developer* yang disimpan dalam bentuk *enum*. *Test case* juga terhubung dengan data aplikasi, modul, menu, user pembuat, *QA*, dan *developer* penanggung jawab.

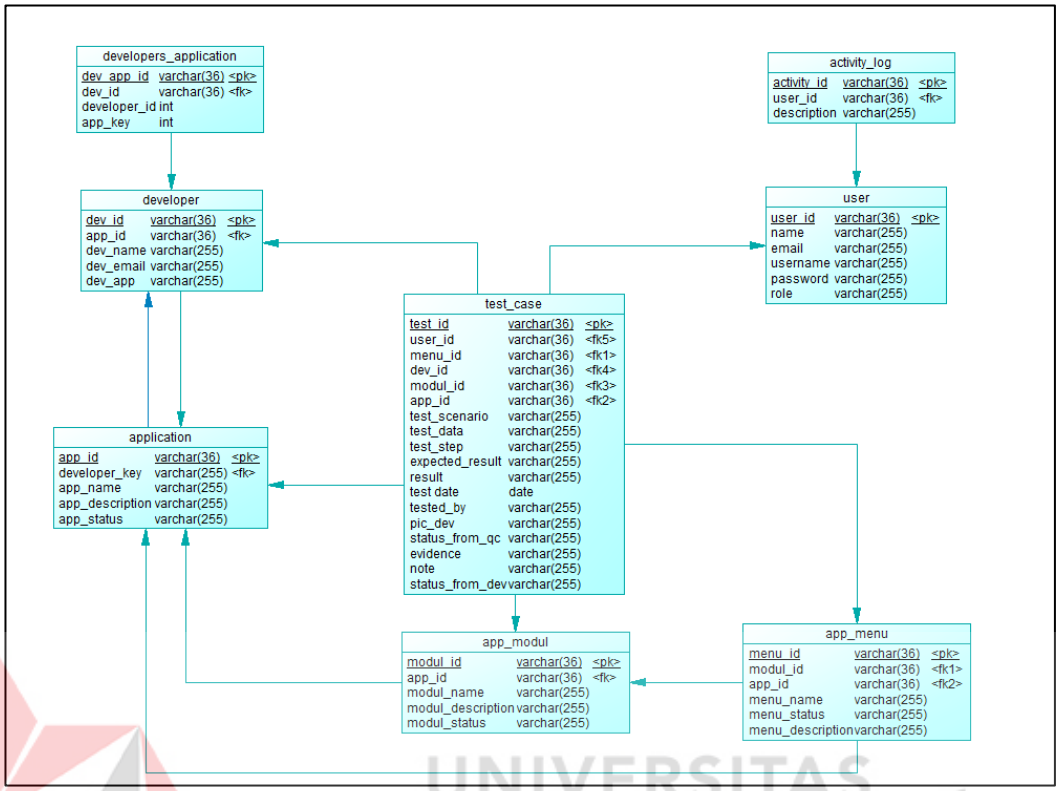
E. Conceptual Data Model (CDM)



Gambar 4. 18 Conceptual Data Model(CDM) Tester App

Pada Gambar 4.18 diatas adalah *Conceptual Data Model (CDM)* pada sistem *Tester App* dirancang untuk menggambarkan hubungan antar entitas utama dalam proses pencatatan pengujian perangkat lunak di PT. Sofco Graha. *CDM* ini merepresentasikan struktur data yang digunakan untuk mendukung fitur-fitur sistem, seperti pencatatan aplikasi, modul, menu, *test case*, *developer*, serta pelacakan aktivitas pengujian. Dalam model ini, ditunjukkan bahwa satu entitas *application* dapat memiliki banyak *app_modul*, dan setiap *app_modul* dapat memiliki banyak *app_menu*. Seluruh entitas tersebut menjadi acuan utama dalam proses pencatatan pengujian oleh *Quality Assurance (QA).* Entitas *test_case* menjadi pusat dari proses pengujian, karena mencatat detail pengujian berdasarkan aplikasi, modul, menu, serta user yang terlibat. Selain itu, *CDM* juga menampilkan entitas user sebagai representasi pengguna sistem yang memiliki peran tertentu (*QA, Developer, Project Manager*), serta entitas *developer_application* yang merepresentasikan hubungan *many-to-many* antara *developer* dan *application*. Hubungan antar entitas ditunjukkan melalui asosiasi "*Has Many*" untuk menunjukkan bahwa satu entitas induk dapat memiliki banyak data anak yang berkaitan. Model ini dibuat untuk menangkap kebutuhan bisnis yang kompleks dari sisi *QA* dan *Developer*, namun tetap mudah dipahami sebagai dasar perancangan basis data.

F. Physical Data Model (PDM)



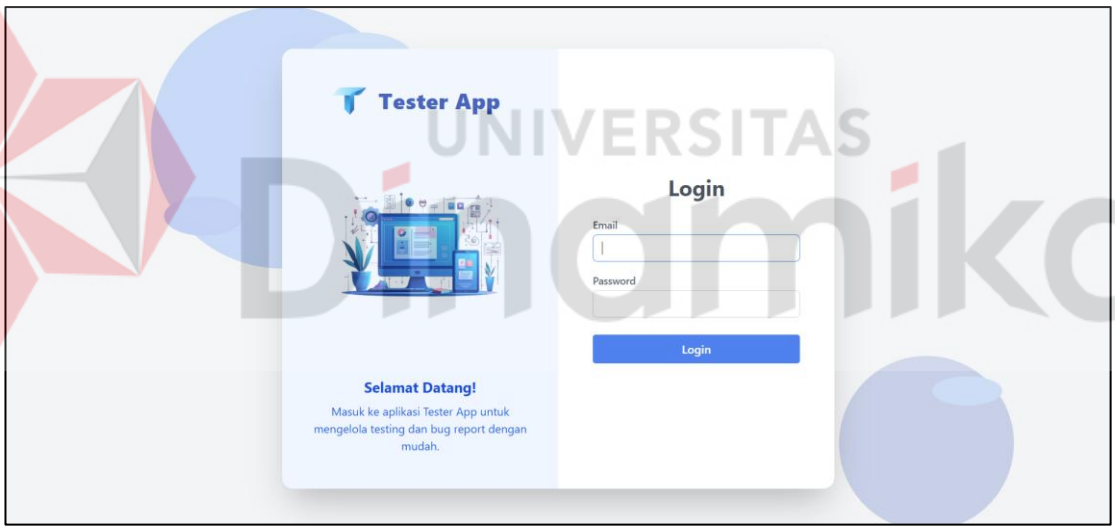
Gambar 4. 19 Physical Data Model (PDM)

Pada Gambar 4.19 diatas adalah *Physical Data Model (PDM)* ini merupakan bentuk implementasi nyata dari *CDM* ke dalam struktur tabel database yang digunakan pada sistem. Dalam *PDM* ini, setiap entitas telah diterjemahkan menjadi tabel lengkap dengan penentuan *primary key (<pk>), foreign key (<fk>),* dan tipe data spesifik seperti *varchar(255), int,* dan *date* untuk masing-masing atribut. Tabel *test_case* menjadi tabel utama yang menyimpan informasi detail proses pengujian, termasuk referensi ke *user_id, app_id, modul_id, menu_id, serta atribut tambahan seperti status_from_qc* dan *status_from_dev* yang disimpan dalam bentuk *varchar* untuk menampung status pengujian dari masing-masing pihak. Setiap relasi antar entitas terhubung melalui *foreign key* yang mendukung integritas data, seperti hubungan antara *developer_application* dengan

developer dan *application*, serta hubungan *activity_log* dengan *user*. Selain itu, penyesuaian struktur juga dilakukan dengan memberikan nama atribut yang konsisten dan sesuai dengan kebutuhan sistem Laravel yang digunakan sebagai *framework* pengembangan aplikasi. *PDM* ini menjadi acuan dalam pembuatan migrasi database di *Laravel* dan memastikan bahwa struktur data di *backend* dapat berjalan optimal serta mendukung seluruh fitur sistem seperti manajemen *testing*, pelaporan, dan dokumentasi *bug* secara terstruktur dan efisien.

G. Implementasi Aplikasi

G.1. Halaman Login

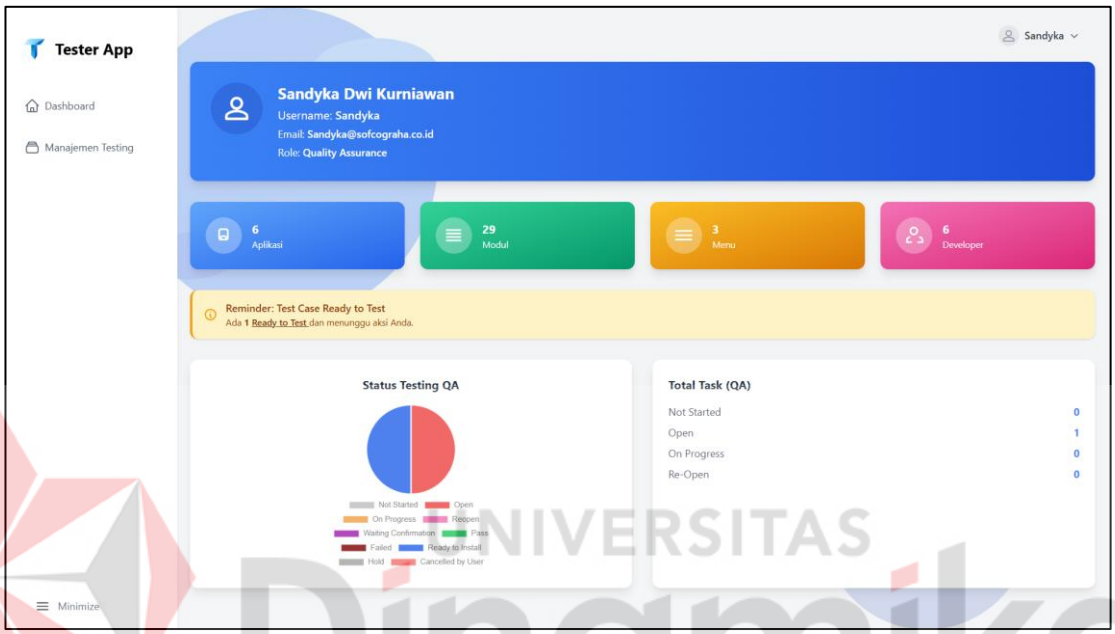


Gambar 4. 20 User Interface Halaman Login

Pada Gambar 4.20 diatas adalah Halaman *login* yang merupakan titik awal bagi pengguna untuk masuk ke dalam sistem *Tester App*. Pada halaman ini, pengguna diminta untuk memasukkan alamat *email* dan kata sandi yang telah terdaftar guna melakukan proses autentikasi. Selain sebagai sarana keamanan untuk membatasi akses, halaman *login* juga digunakan untuk membedakan hak

akses berdasarkan role pengguna, yaitu *Quality Assurance (QA), Developer*, dan *Project Manager*. Setiap *role* akan diarahkan ke tampilan *Dashboard* dan akses menu yang sesuai dengan *role* nya.

G.2. Halaman Dashboard QA



Gambar 4. 21 User Interface Halaman Dashboard QA

Pada Gambar 4.21 diatas adalah *Dashboard* untuk *role Quality Assurance (QA)* pada *Tester App yang* dirancang untuk memberikan gambaran umum terkait aktivitas pengujian yang sedang berlangsung. Pada bagian atas *dashboard*, ditampilkan informasi pengguna seperti nama, *username*, *email*, dan *role* pengguna. Terdapat juga ringkasan jumlah data utama seperti total aplikasi yang diuji, jumlah modul, jumlah menu, dan total developer yang terlibat dalam pengujian. Di bawahnya, terdapat pengingat atau notifikasi yang menampilkan jumlah *test case* yang sudah berstatus "*Ready to Test*" dan menunggu tindakan dari *QA*. Selain itu, dashboard ini juga dilengkapi dengan visualisasi grafik *pie*

yang menampilkan status *testing QA*, seperti *Not Started, On Progress, Pass, Failed*, hingga *Ready to Install*. Di sisi kanan terdapat informasi ringkas terkait jumlah total *task QA* berdasarkan status. Seluruh elemen pada dashboard ini bertujuan untuk membantu *QA* memantau *progress* pengujian secara menyeluruh dan efisien melalui satu tampilan antarmuka.

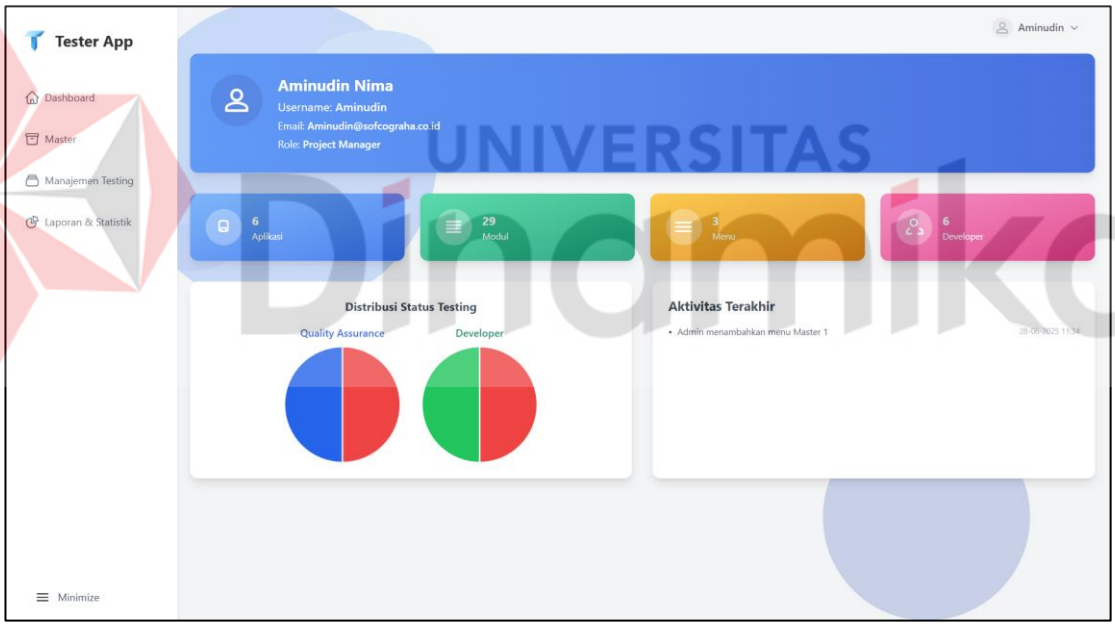
G.3. Halaman Dashboard Developer

Gambar 4. 22 User Interface Halaman Dashboard Developer

Pada Gambar 4.22 diatas adalah *Dashboard* untuk role *Developer* pada *Tester App* yang berfungsi sebagai pusat informasi terkait tugas-tugas pengembangan dan perbaikan *bug* dari aplikasi yang sedang diuji. Pada bagian atas, ditampilkan informasi identitas pengguna seperti nama, *email*, dan peran sebagai *Developer*. Tampilan *dashboard* menyajikan ringkasan jumlah total entitas penting seperti aplikasi, modul, menu, dan jumlah *developer* yang terlibat. Di bawahnya terdapat notifikasi pengingat terkait test case yang telah berstatus

"*Ready to Install*", yang menandakan bahwa *Developer* perlu segera menindaklanjuti proses instalasi. Untuk memudahkan pemantauan progres, *dashboard* juga dilengkapi grafik *pie* yang menunjukkan status *task developer* secara visual, mencakup status seperti *Not Started*, *On Progress*, *Installed*, *Ready to Test*, hingga *Closed*. Selain itu, terdapat tabel jumlah *task* berdasarkan status terkini, yang membantu *developer* dalam mengatur prioritas pekerjaan. Tampilan ini dirancang agar *Developer* dapat mengetahui status dan beban tugas secara cepat dan terfokus.

G.4. Halaman Dashboard Project Manager

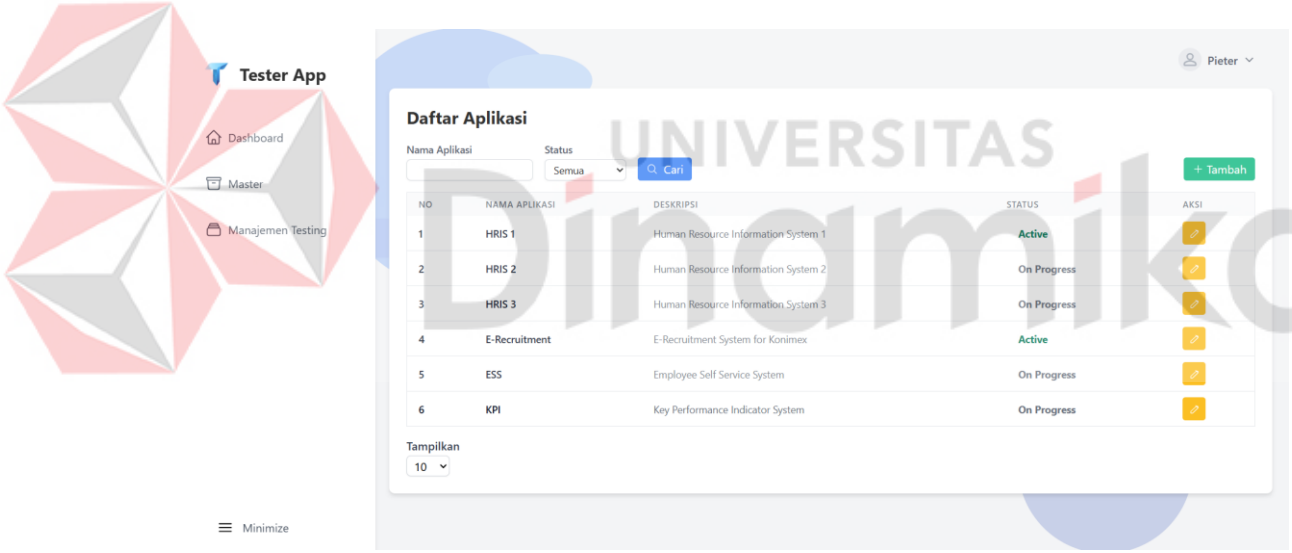


Gambar 4. 23 User Interface Halaman Dashboard Project Manager

Pada Gambar 4.23 diatas adalah *Dashboard* untuk *role Project Manager* pada *Tester App* dirancang untuk memberikan ringkasan status pengujian dari sudut pandang manajerial. Di bagian atas, *Project Manager* dapat melihat informasi pengguna seperti nama, *email*, dan peran. *Dashboard* menampilkan

total data penting seperti jumlah aplikasi, modul, menu, dan total *developer* yang terlibat, sebagai indikator cakupan sistem yang sedang dikelola. Fitur utama pada tampilan ini adalah grafik distribusi status *testing* yang dibagi berdasarkan dua peran, yaitu *Quality Assurance* dan *Developer*. Grafik ini memberikan gambaran visual mengenai sejauh mana *progress* pengujian dan perbaikan dilakukan oleh masing-masing tim. Selain itu, tersedia juga panel Aktivitas Terakhir yang mencatat *log* perubahan atau tindakan penting yang dilakukan dalam sistem, sehingga *Project Manager* dapat memantau aktivitas terbaru secara cepat.

G.5. Daftar Aplikasi



Gambar 4. 24 User Interface Halaman Daftar Aplikasi

Pada Gambar 4.24 diatas adalah Halaman *Daftar Aplikasi* pada *Tester App* yang digunakan untuk menampilkan seluruh aplikasi yang sedang dan akan dilakukan pengujian. Setiap entri pada tabel menampilkan informasi penting seperti nama aplikasi, deskripsi singkat, status pengujian, serta aksi yang dapat dilakukan. Pengguna dapat melihat status aplikasi, seperti *Active* atau *On*

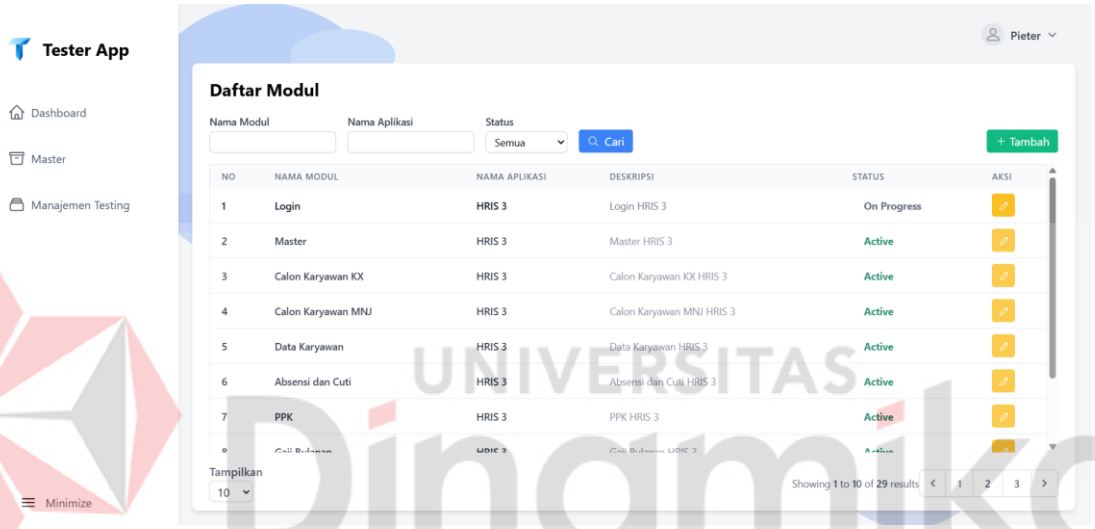
Progress, yang menunjukkan tahapan pengujian dari masing-masing aplikasi. Di bagian atas tersedia fitur pencarian berdasarkan nama dan filter status, yang memudahkan pengguna dalam menemukan aplikasi tertentu secara cepat. Selain itu, tersedia tombol aksi seperti *edit*, detail, dan hapus yang memberikan fleksibilitas kepada pengguna dalam mengelola data aplikasi. Tombol Tambah di pojok kanan atas berfungsi untuk menambahkan data aplikasi baru ke dalam sistem.

Gambar 4. 25 User Interface Halaman Tambah & Edit Aplikasi

Pada Gambar 4.25 diatas adalah Halaman *Input/Edit Aplikasi* pada *Tester App* yang digunakan untuk menambahkan data aplikasi baru maupun mengubah data aplikasi yang sudah ada. *Form* ini terdiri dari tiga komponen utama, yaitu *Nama Aplikasi*, *Deskripsi*, dan *Status Aplikasi*. Pengguna dapat mengisi nama aplikasi dan memberikan deskripsi singkat mengenai aplikasi tersebut. Pada bagian status, pengguna dapat memilih status awal aplikasi, seperti *Not Started*, yang menandakan bahwa proses pengujian untuk aplikasi tersebut belum dimulai.

Tombol *Simpan* digunakan untuk menyimpan data ke dalam sistem, sedangkan tombol *Kembali* memungkinkan pengguna untuk kembali ke halaman daftar aplikasi tanpa menyimpan perubahan. Halaman ini dirancang dengan antarmuka yang sederhana dan minimalis, sehingga memudahkan pengguna dalam mengelola data aplikasi secara cepat dan efisien.

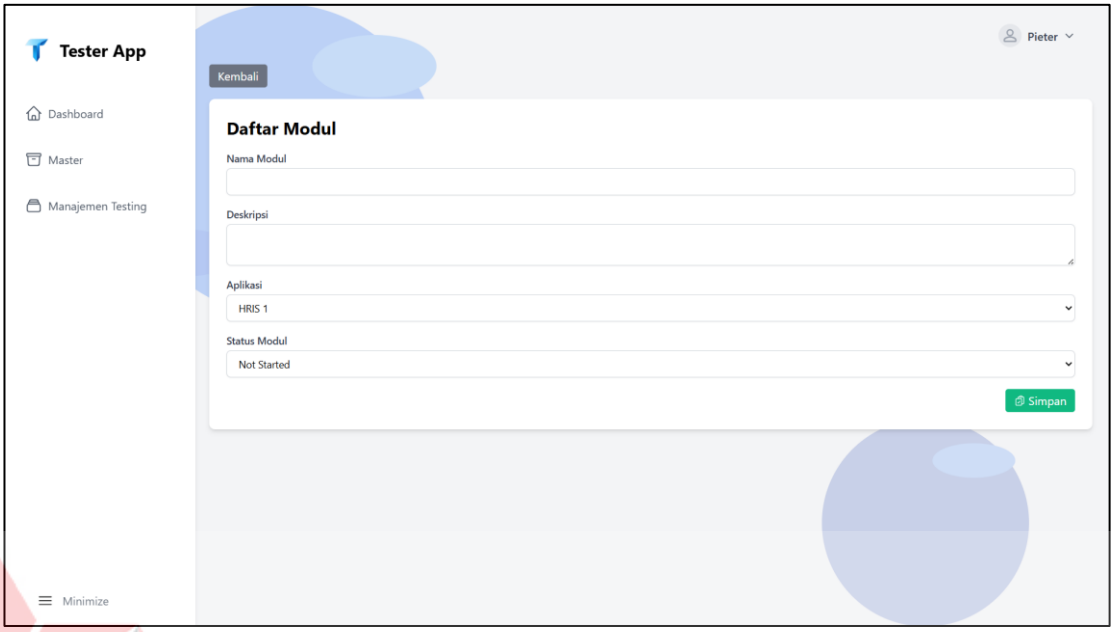
G.6. Daftar Modul



Gambar 4. 26 User Inteface Halaman Daftar Modul

Pada Gambar 4.26 diatas adalah Halaman Daftar Modul pada *Tester App* yang digunakan untuk menampilkan seluruh modul yang terdaftar dalam setiap aplikasi yang sedang diuji. Setiap entri dalam tabel memuat informasi seperti nama modul, nama aplikasi induk, deskripsi modul, serta status pengujian modul tersebut. Status dapat berupa *Active*, *On Progress*, dan lainnya, yang menggambarkan kondisi terkini dari masing-masing modul. Untuk memudahkan pencarian data, tersedia kolom pencarian berdasarkan nama modul, nama aplikasi, dan filter status. Fitur ini sangat membantu *QA* atau *Developer* saat ingin memantau modul tertentu dalam aplikasi yang sedang diuji. Tersedia pula tombol

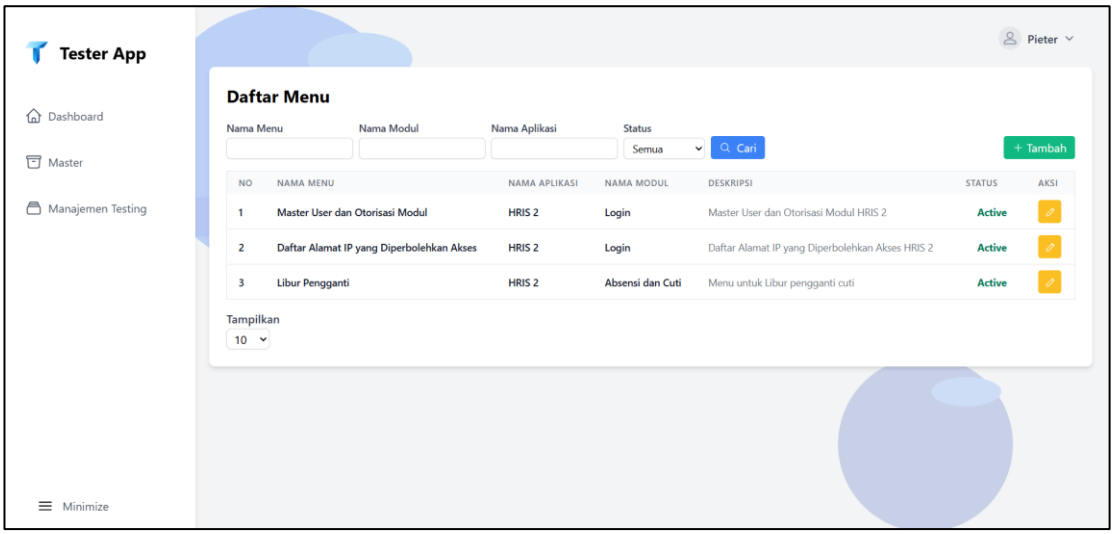
aksi untuk melakukan edit, melihat detail, dan menghapus data modul. Tombol Tambah memungkinkan pengguna untuk menambahkan modul baru ke sistem.



Gambar 4. 27 User Interface Halaman Tambah & Edit Modul

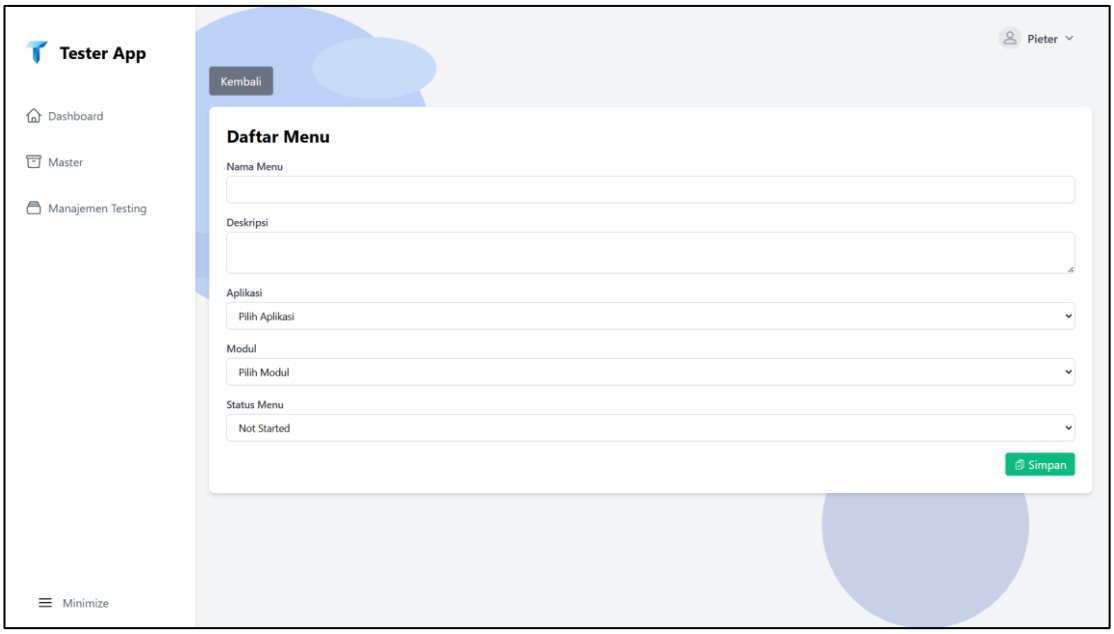
Pada Gambar 4.27 diatas adalah Halaman *Input/Edit Modul* pada *Tester App* yang berfungsi untuk menambahkan data modul baru atau mengubah informasi modul yang telah ada dalam suatu aplikasi. *Form* ini terdiri dari beberapa *field* penting, yaitu Nama Modul, Deskripsi, Aplikasi, dan Status Modul. Pengguna dapat menentukan nama modul yang akan diuji, memberikan deskripsi singkat terkait fungsionalitas modul, serta memilih aplikasi induk tempat modul tersebut berada. Selain itu, status modul juga dapat disesuaikan berdasarkan progres pengujian, seperti *Not Started*, *On Progress*, *Active*, dan lain-lain. Tombol *Simpan* digunakan untuk menyimpan data ke dalam sistem, sedangkan tombol *Kembali* memungkinkan pengguna kembali ke halaman daftar modul tanpa menyimpan perubahan

G.7. Daftar Menu



Gambar 4. 28 User Interface Halaman Daftar Menu

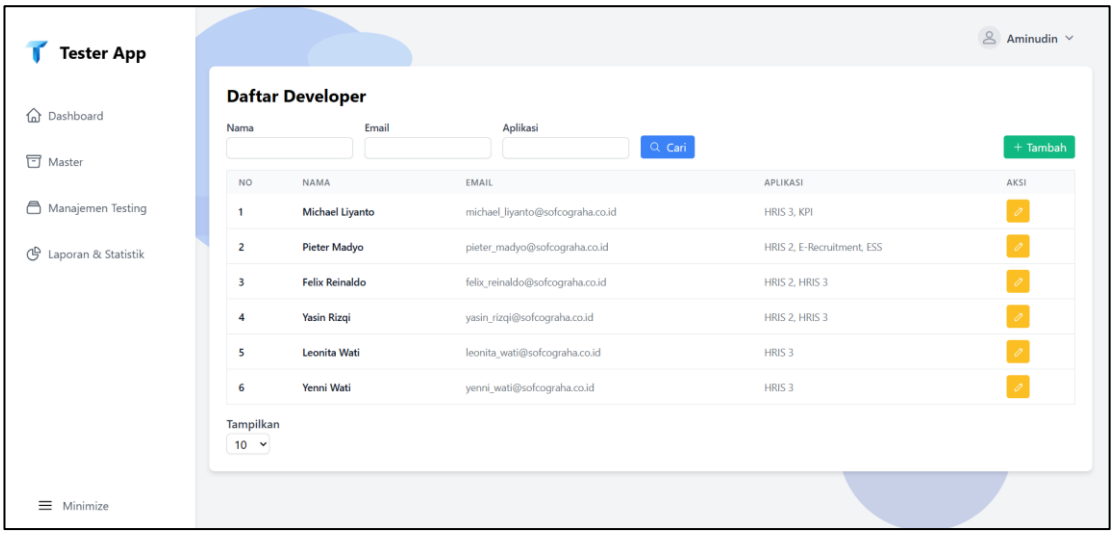
Pada Gambar 4.28 diatas adalah Halaman *Daftar Menu* pada *Tester App* yang digunakan untuk menampilkan seluruh data menu yang terdapat dalam setiap modul aplikasi yang sedang diuji. Dalam tabel ini, ditampilkan informasi penting seperti nama menu, nama modul, nama aplikasi induk, deskripsi menu, dan status pengujian. Fitur pencarian disediakan pada bagian atas tabel untuk mempermudah pengguna dalam mencari menu berdasarkan nama menu, nama modul, nama aplikasi, atau status tertentu. Status menu ditampilkan untuk menunjukkan progres pengujian, seperti *Active*, yang menandakan bahwa menu tersebut telah melalui proses pengujian dan siap digunakan. Tersedia pula tombol aksi di setiap baris untuk mengubah dan melihat detail menu. Di sisi kanan atas, terdapat tombol Tambah yang digunakan untuk menambahkan menu baru ke dalam sistem.



Gambar 4. 29 User Interface Halaman Tambah & Edit Menu

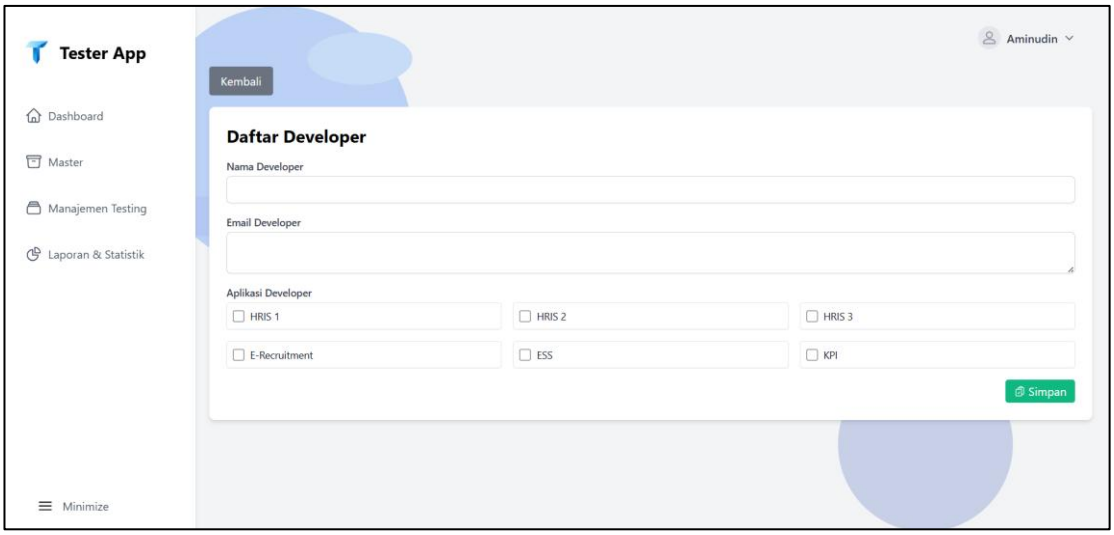
Pada Gambar 4.29 diatas adalah Halaman *Input/Edit Menu* pada *Tester App* yang digunakan untuk menambahkan menu baru atau memperbarui data menu yang telah ada dalam suatu modul aplikasi. Form ini terdiri dari beberapa isian penting, seperti Nama Menu, Deskripsi, Aplikasi, Modul, dan Status Menu. Pengguna dapat menentukan nama menu dan memberikan deskripsi singkat yang menjelaskan fungsi menu tersebut. Dropdown Aplikasi dan Modul digunakan untuk mengaitkan menu dengan struktur aplikasi dan modul yang relevan. Selain itu, pengguna juga dapat menentukan status pengujian menu, seperti *Not Started*, untuk menunjukkan bahwa pengujian terhadap menu tersebut belum dimulai. Setelah seluruh data diisi dengan benar, pengguna dapat menekan tombol *Simpan* untuk menyimpan data ke dalam sistem, atau menggunakan tombol *Kembali* untuk membatalkan proses input dan kembali ke halaman daftar menu

G.8. Daftar Developer



Gambar 4. 30 User Interface Halaman Daftar Developer

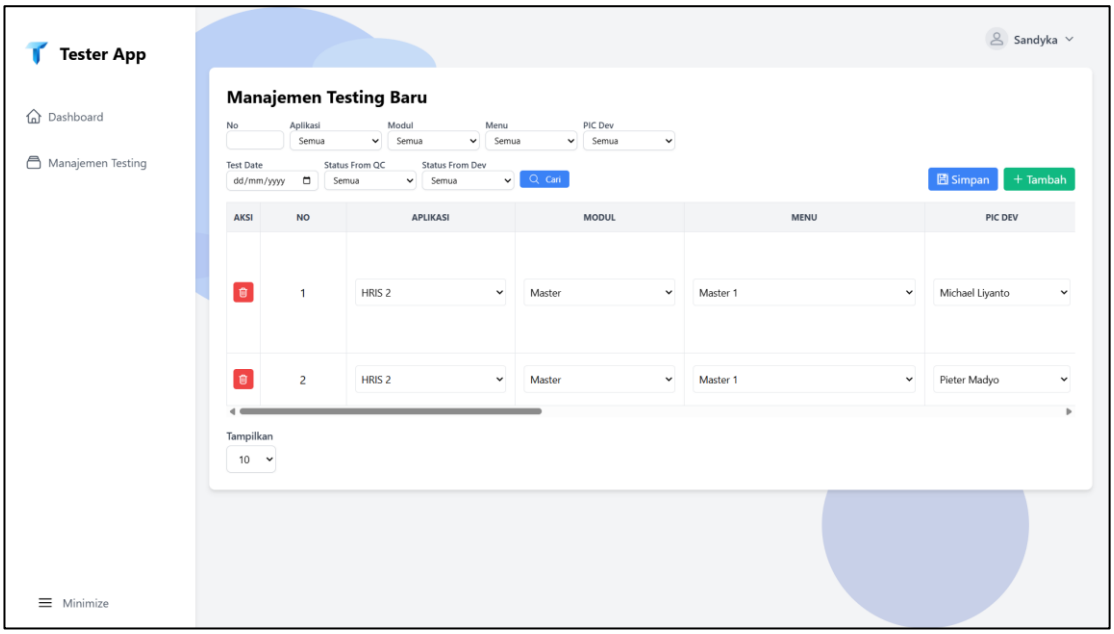
Pada Gambar 4.30 diatas adalah Halaman Daftar *Developer* pada *Tester App* yang digunakan untuk menampilkan informasi seluruh *developer* yang terlibat dalam proses pengujian aplikasi. Setiap baris dalam tabel memuat data seperti nama *developer*, alamat *email*, dan daftar aplikasi yang ditangani, sehingga memudahkan tim *QA* atau pengelola proyek dalam memantau keterlibatan masing-masing *developer*. Untuk menunjang pencarian data, tersedia kolom pencarian berdasarkan nama, email, dan aplikasi. Pengguna juga dapat menambahkan *developer* baru melalui tombol Tambah, serta melakukan aksi *edit* dan melalui tombol yang tersedia di setiap baris tabel.



Gambar 4. 31 User Interface Halaman Tambah & Edit Developer

Pada Gambar 4.31 diatas adalah Halaman *Input/Edit Developer* pada *Tester App yang* digunakan untuk melakukan *input* atau pengeditan data developer yang terlibat dalam pengujian aplikasi. Dalam halaman ini, pengguna dapat mengisi Nama *Developer*, *Email Developer*, serta memilih satu atau lebih aplikasi yang menjadi tanggung jawab *developer* melalui daftar *checkbox* yang tersedia. Setelah seluruh data terisi, pengguna dapat menyimpan informasi tersebut dengan menekan tombol Simpan, atau kembali ke halaman sebelumnya menggunakan tombol Kembali. Tampilan yang sederhana dan terstruktur ini memudahkan proses penambahan maupun pembaruan data *developer* dalam sistem.

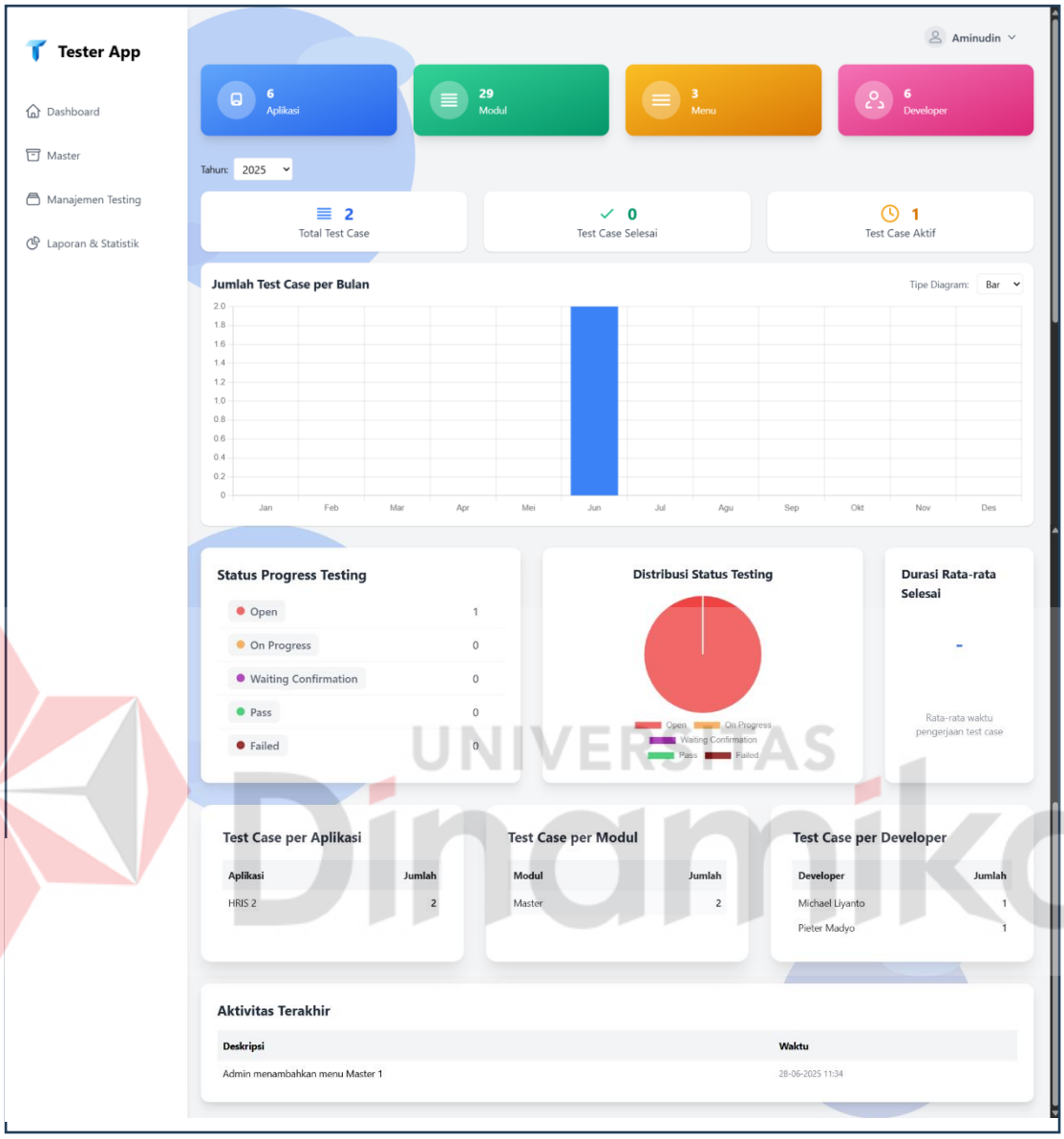
G.9. Halaman Manajemen Testing



Gambar 4. 32 User Interface Halaman Manajememt Testing

Pada Gambar 4.32 diatas adalah Halaman Manajemen *Testing* pada *Tester App* yang digunakan untuk mengelola proses pengujian setiap modul dalam aplikasi yang sedang dikembangkan. Pengguna dapat menambahkan entri pengujian dengan memilih aplikasi, modul, menu, serta menentukan *PIC Developer* (*person in charge*) yang bertanggung jawab atas pengujian tersebut. Untuk memudahkan pencarian data testing yang sudah ada, tersedia berbagai *filter* seperti nama aplikasi, modul, menu, *PIC Dev,* tanggal pengujian, serta status dari *QA* maupun *Developer*. Tombol Tambah memungkinkan pengguna menambahkan baris pengujian baru, sedangkan tombol Simpan digunakan untuk menyimpan data yang telah diinput

G.10. Halaman Laporan & Statistik



Gambar 4. 33 User Interface Halaman Laporan & Statistik

Pada Gambar 4.33 diatas adalah Halaman Laporan & Statistik pada *Tester App* yang menyajikan ringkasan data terkait aktivitas pengujian aplikasi secara menyeluruh dalam bentuk angka, grafik, dan diagram yang mudah dipahami. Pengguna dapat melihat jumlah total aplikasi, modul, menu, dan *developer* yang terdaftar, serta statistik pengujian seperti jumlah *test case, test*

case yang selesai, dan *test case* yang masih aktif. Terdapat grafik jumlah *test case* per bulan, diagram distribusi status testing (*Open, On Progress, Waiting Confirmation, Pass, Failed*), serta informasi durasi rata-rata penyelesaian *test case*. Selain itu, tersedia juga rekap *test case* berdasarkan aplikasi, modul, dan developer, serta daftar aktivitas terakhir yang dilakukan di sistem.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan pelaksanaan kerja praktik yang dilakukan di PT. Sofco Graha, serta hasil dari proses analisis, perancangan, dan implementasi sistem, dapat disimpulkan bahwa pembangunan sistem informasi pencatatan pengujian perangkat lunak *(Tester App)* telah berhasil dilakukan sesuai dengan kebutuhan pengguna. Sistem ini memberikan solusi atas permasalahan pencatatan manual yang sebelumnya dilakukan oleh tim *Quality Assurance (QA)* dengan menghadirkan fitur-fitur seperti pencatatan daftar aplikasi, modul, menu, serta pengelolaan test case dan dokumentasi bug secara terpusat. Aplikasi yang dibangun telah menyediakan fitur utama seperti *form input* data aplikasi, modul, menu, manajemen *tesing*, pelacakan status pengujian, dan visualisasi data dalam bentuk grafik dan tabel yang dapat diakses oleh *QA*, *Developer*, dan *Project Manager*. Fitur-fitur yang dikembangkan dalam *Tester App* mampu mendukung pencatatan dan pemantauan proses pengujian perangkat lunak secara lebih terorganisir, terdokumentasi, dan terpusat, sehingga memudahkan tim *QA*, *Developer*, dan *Project Manager* dalam melakukan pelacakan *bug*, monitoring progres pengujian, serta pelaporan kepada manajemen.

5.2 Saran

Untuk pengembangan sistem di masa mendatang, terdapat beberapa hal yang dapat dijadikan saran perbaikan. Pertama, sistem sebaiknya dikembangkan lebih lanjut agar dapat terintegrasi dengan *tools* manajemen proyek seperti *Jira*

atau *version control* seperti *Git*, guna meningkatkan koordinasi antara tim *QA* dan *Developer* secara lebih menyeluruh. Kedua, fitur notifikasi otomatis melalui email atau sistem *alert* disarankan untuk ditambahkan agar setiap perubahan status, progres pengujian, atau temuan bug dapat segera diketahui oleh pihak terkait. Ketiga, perlu dilakukan peningkatan keamanan sistem, terutama dalam pengelolaan hak akses pengguna, guna memastikan data pengujian hanya dapat diakses oleh pihak yang berwenang. Keempat, sistem akan lebih fleksibel jika dikembangkan dalam versi *mobile* atau memiliki desain yang lebih responsif agar nyaman diakses melalui berbagai perangkat, termasuk *smartphone*. Terakhir, disarankan agar sistem ini diuji lebih luas oleh seluruh pengguna terkait dalam lingkungan perusahaan, dan dilakukan evaluasi berkala berdasarkan masukan pengguna guna memperbaiki serta menyempurnakan fungsionalitas sistem kedepannya.

Daftar Pustaka

Anggarah, R., Nurma Feblia, K., Rizgita Amanda, K., Adzkiatul Mardiyah, Q., Chiara Amanda, F., Syarifatul Mursyidah, A., Villareal, Y., Valgiyos Aritonang, M. D., & Setiawan, Y. (2025). *Studi Perbandingan Penerapan Pola Model-View-Controller (MVC) dalam Lima Framework Web Populer: Laravel, Django, Ruby on Rails, Asp.net MVC, Spring MVC*. https://doi.org/10.70656/ijcse.v2i01.306

Dhandy, *, Hasibuan, B., Bintang, D., Politeknik, H., Medan, N., Ageva, A., Pinem, A., Negeri, P., Fauziah, M., & Politeknik, R. (2025). Pengujian Kualitas Website Sistem Pembelajaran Digital (SIPADI POLMED) menggunakan Black Box Testing dan Standar ISO/IEC 25010. *Jurnal Cakrawala Akademika (JCA)*, *2*(1), 1027–1033. https://doi.org/10.70182/JCA.v1.i1.625

Fachri, B., & Wahyu Surbakti, R. (2021). PERANCANGAN SISTEM DAN DESAIN UNDANGAN DIGITAL MENGGUNAKAN METODE WATERFALL BERBASIS WEBSITE (STUDI KASUS: ASCO JAYA). In *Journal of Science and Social Research* (Issue 3). http://jurnal.goretanpena.com/index.php/JSSR

Hidayat, M., Izzati, N., Prirhatama, A., Wijaya, Y., & Kurmilasari, S. (2025). Pengujian Perangkat Lunak pada Website Ka'Cake. *JATI (Jurnal Mahasiswa Teknik Informatika)*, *9*.

Nugraha, W., Syarif, M., & Dharmawan, W. S. (2018). PENERAPAN METODE SDLC WATERFALL DALAM SISTEM INFORMASI INVENTORI BARANG BERBASIS DESKTOP. *JUSIM (Jurnal Sistem Informasi Musirawas)*, *3*(1), 22–28. https://doi.org/10.32767/jusim.v3i1.246

Supiyandi, S., Zen, M., Rizal, C., & Eka, M. (2022). Perancangan Sistem Informasi Desa Tomuan Holbung Menggunakan Metode Waterfall. *JURIKOM (Jurnal Riset Komputer)*, *9*(2), 274. https://doi.org/10.30865/jurikom.v9i2.3986

Supriadi, E., Nurcahyo, W., Faizah, N. M., Prodi, P., Komputer, I., Jagakarsa, T., Kota, J., Selatan, D., & Khusus, I. (2025). Jurnal Ilmu Komputer dan Teknologi Informasi Perancangan Aplikasi Sistem Informasi Wisata Alam di Kota Pandeglang, Provinsi Banten, Berbasis Web dengan Metode Waterfall Menggunakan PHP dan MySQL. In *Jurnal Ilmu Komputer dan Teknologi Informasi (JIKTI)* (Vol. 2, Issue 1). https://journal.stmiki.ac.id