BAB 3
PELAKSANAAN KERJA MAGANG

Pada bab 3 merupakan bab pelaksanaan kerja magang dimana penulis akan memaparkan beberapa hal terkait pelaksaan kerja magang seperti kedudukan dan organisasi, tugas yang dilakukan, dan uraian pelaksanaan magang.

3.1 Kedudukan dan Organisasi

Kedudukan posisi penulis pada saat melaksanakan kerja magang di PT. Accelist Lentera Indonesia adalah *Full stack developer*. Peran dari *full stack developer* pada PT. Accelist Lentera Indonesia adalah membangun aplikasi berbasis situs web maupun *mobile*, melakukan pertemuan dengan tim *business analyst* terkait projek yang sedang dikerjakan, dan melakukan *bug fixing* ataupun permasalahan lainnya. Dalam pelaksanaan kerja magang, terdapat pembimbing untuk peserta magang dapat memahami teknologi yang digunakan pada PT. Accelist Lentera Indonesia yaitu Ryan Elian selaku CTO, Tie Antono selaku *chief of architect*, Devin Jastona selaku *head of* IT *Division*, dan Excleson Manuel Julio selaku *architect*. Serta pembimbing penulis dalam melaksanakan kerja magang yaitu Gina Akmalia selaku *VP human resources and finance*.

Dalam pengerjaan tugas magang penulis diberikan pelatihan terlebih dahulu selama 2 bulan secara daring melalui Google Meeting dan berkoordinasi terkait pelatihan melalui aplikasi Discord. Dalam pelaksanaan pekerjaan setelah pelatihan, penulis ditugaskan di bagian IT *Operation* atau IT *Division* dan dalam berkomunikasi dilakukan melalui Whatsapp dan untuk *daily* dan *meeting* dilakukan melalui aplikasi Discord. Dalam pengerjaan tugas dalam divisi IT *Division* penulis dibimbing oleh Retno Endah Utari selaku *Project Manager*, Albert dan Rahadi Fauzan selaku *developer*. Pembagian tugas dilakukan setelah *setup* dan *installing* aplikasi yang digunakan dan setelah menyelesaikan pekerjaan hasil pengerjaan tersebut akan di *push* ke dalam *branch* GitHub baru sesuai dengan fitur yang dibuat. Proses selanjutnya adalah melakukan *merge request* untuk dapat dievaluasi dan selanjutnya akan digabungkan ke dalam *branch* utama atau *branch main*.

3.2 Tugas yang Dilakukan

Dalam melaksanakan kerja magang, pada 2 bulan pertama yaitu di mulai pada bulan Maret 2023 peserta magang wajib mengikuti kegiatan pelatihan yang sediakan. Pelaksanaan pelatihan kerja magang terdapat 2 jenis prosedur yaitu melalui situs web Codecademy dan dilatih secara langsung oleh pelatih langsung dari PT. Accelist Lentera Indonesia melalui Google Meeting. Materi pelatihan di minggu pertama adalah .NET *Basics* dimana terdapat pembelajaran terkait bahasa pemograman C#, pengenalan terhadap perangkat lunak Visual Studio, dan pengenalan terhadap .NET dasar. Sesi pelatihan .NET *Basics* diakhiri dengan pemberian latihan untuk membangun sebuah aplikasi kasir sederhana dimana terdapat fitur daftar produk (*add product*, *update product*, dan *delete product*), fitur *add to cart*, *checkout*, dan *purchase history*.

Topik pelatihan kedua adalah lanjutan dari .NET *basics* yaitu .NET *Intermediate* dimana pada tahap ini peserta magang dilatih untuk menggunakan *versioning control* seperti GitHub dan GitLab. Serta peserta magang diberikan pengajaran tentang *windows form* pada *Visual Studio* dan dapat memahami HTTP *client*. Berbeda dengan sesi pelatihan sebelumnya, pada .NET *intermediate* terdapat ujian untuk para peserta magang. Dimana peserta magang diwajibkan untuk membuat aplikasi *desktop* sederhana menggunakan *windows form* dengan ketentuan aplikasi dapat melakukan CRUD (*create, read, update, delete*) dari *public* API.

Sejalan dengan sebelumnya, API yang digunakan dalam pelatihan menggunakan API yang sudah tersedia sebelumnya. Berdasarkan fakta tersebut pelatihan berikutnya adalah ASP.NET *Web* API dimana peserta magang dapat membuat API untuk dapat digunakan dalam projek sesungguhnya. Pada pelatihan ini, bahasa pemograman yang digunakan adalah C# dan pelatihan yang diberikan meliputi (1) *logging* menggunakan serilog, (2) *framework environment*, (3) CRUD *web* API, (4) model *request* dan status kode HTTP, (5) MARVEL *pattern* dengan menggunakan validasi, dan (6) media Swagger dan Postman. Terdapat ujian untuk membuat *web* API untuk tiket transportasi dimana API dapat mendapatkan data tiket yang tersedia, *booking* tiket yang tersedia, riwayat tiket yang sudah di *booking* sebelumnya, membatalkan tiket yang sudah di *booking*, dan melakukan pembaharuan data kuantitas tiket yang di *booking*.

Pelatihan selanjutnya adalah *Front-end programming basics* dimana pada pelatihan ini hanya diberikan tugas untuk mengerjakan 7 *course* yang tersedia di situs web Codecademy. Tahap selanjutnya adalah *Front-end programming*

intermediate menggunakan *framework* Next.JS dengan Typescript dan pada pelatihan ini diajarkan untuk mengenal *framework* mulai dari halaman hingga *routing* serta cara membuat tampilan CRUD, menghubungkan *front-end* terhadap API menggunakan NSwag Studio, Axios, dan SWR. Terdapat pelatihan *framework* lainnya yaitu tailwind yang dapat digunakan di dalam projek Next.JS. Berdasarkan pelatihan membangun *web* API sebelumnya, pada pelatihan *front-end* diberikan ujian untuk membangun API dan situs web untuk dapat membuat situs web restoran yang dapat menampilkan daftar restoran dan menu yang tersedia, *add to cart*, dan *checkout cart* yang tersambung dengan *back-end*.

Berdasarkan pelatihan - pelatihan sebelumnya, dapat dilihat bahwa pelatihan belum menggunakan basis data sehingga terdapat pelatihan terakhir yaitu SQL *programming*. Pada pelatihan ini peserta magang dilatih untuk dapat bisa menghubungkan basis data ke *back-end* menggunakan PostgreSQL. Ujian yang diberikan pada pelatihan kali ini adalah untuk membuat situs web untuk penerbangan yang terdapat beberapa fitur yaitu CRUD perusahaan penerbangan, CRUD untuk pesawat, CRUD jadwal penerbangan, dan *dashboard* yang berisi daftar pesawat yang tersedia.

Sejalan dengan pernyataan sebelumnya, sesi pelatihan SQL *programming* merupakan pelatihan terakhir yang disediakan perusahaan. Tahap berikutnya adalah penugasan peserta magang ke projek latihan maupun projek perusahaan yang sedang berlangsung. Tugas yang diberikan terhadap penulis adalah untuk melanjutkan membangun sebuah situs web *ticketing request* milik perusahaan sehingga dapat digunakan oleh karyawan perusahaan secepatnya. Pengerjaan projek ini merupakan pengerjaan projek dari bagian IT *Division* yaitu projek internal dari perusahaan.

Pada situs web *ticketing request* yang dibangun, karyawan perusahaan dapat mengaksesnya dan dapat melihat jumlah tiket yang ada pada setiap projek yang sedang dijalankan. Pada setiap tiketnya terdapat judul tiket yang menjelaskan permasalahan atau proses yang sedang dihadapi setiap *developer*, nama projek terkait, jenis permasalahan yang dihadapi seperti hanya sekedar ingin konsultasi, *bug*, dan lainnya, tingkat prioritas tiket, status tiket, tanggal tiket di *request*, tanggal tiket di respond, penjelasan aksi yang dilakukan atau diinginkan *developer*, nama pembuat *request*, dan lainnya. Pada setiap tiket yang tersedia di halaman awal terdapat tombol detail yang akan mengarahkan pengguna ke halaman detail tiket. Dimana pada halaman detail tiket terdapat tombol edit dan informasi yang lebih lengkap dari informasi yang ditampilkan pada halaman awal seperti terdapat

informasi dugaan penyebab *request* dilakukan.

Berdasarkan penjelasan sebelumnya, pada halaman detail terdapat tombol edit dimana pada awalnya input yang tersedia pada halaman bersifat *disabled* dan jika pengguna menekan tombol edit maka semua input akan menjadi *enabled* sehingga pengguna dapat melakukan perubahan atau menambahkan informasi yang kurang lengkap dari setiap tiket. Setelah pengguna telah selesai merubah atau menambahkan informasi maka pengguna akan menekan tombol update ticket yang dimana nantinya situs web akan mengirimkan request terhadap API dan apabila berhasil akan merubah data yang ada di basis data. Bila server atau *back-end* memberikan response *error* maka situs web akan memberikan peringatan terhadap pengguna.

Fitur yang dibangun selanjutnya adalah fitur melihat list tiket berdasarkan projek yang dipilih oleh pengguna. Fitur ini dapat diakses dengan pengguna menekan tombol *hamburger* pada pojok kiri atas halaman awal. Secara otomatis situs web akan menampilkan side bar yang memiliki 2 menu yaitu *projects* dan *settings* kemudian pengguna dapat memilih menu *projects*. Situs web akan mengarahkan pengguna ke halaman yang menampilkan daftar *project* yang sedang aktif atau sedang dikerjakan. Berdasarkan halaman tersebut situs web akan menampilkan halaman yang berisikan *request* tiket yang ada berdasarkan pilihan pengguna di halaman sebelumnya.

Sejalan dengan pernyataan pada paragraf sebelumnya, pada halaman ini terdapat 2 *tab* yaitu *tab ticket dashboard* dan *tab reporting*. Dimana secara otomatis situs web akan menampilkan daftar tiket yaitu tampilan yang terdapat pada *tab ticket dashboard*. *Tab reporting* merupakan halaman yang dapat menampilkan grafik laporan harian, bulanan, *SLA Report*, dan ringkasan dari seluruh laporan yang dapat di export ke dalam Microsoft Excel. Berbeda dari halaman awal yang menampilkan *request* tiket secara keseluruhan dengan projek yang berbeda - beda. Halaman pada *ticket dashboard* menampilkan daftar tiket secara spesifik yang menampilkan 10 daftar tiket pada setiap halamannya. Halaman *ticket dashboard* dapat melakukan pencarian tiket berdasarkan kode yang dimiliki oleh masing - masing tiket.

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Mengikuti *training* tentang .NET *Basics* serta melakukan instalasi terhadap perangkat lunak yang digunakan
2	Mengikuti *training* tentang .NET *Intermediate* (*Windows Form App*)
3	Mengikuti *training* tentang .NET *Intermediate* dan Git *versioning controls*
4	Mengikuti *training* tentang ASP.NET *Web* API
5	Mengikuti *training* tentang ASP.NET *Web* API dengan MARVEL *pattern*
6	Mengikuti *training* tentang *Front-end Programming Basics* (Codecademy)
7	Mengikuti *training* tentang *Front-end Programming Intermediate* dengan Next.JS with Typescript
8	Mengikuti *training* tentang SQL *Programming*
9	*Ticketing Request Website* : *on-boarding* dan pembuatan *landing page*
10	1. Mengikuti *training* tentang *Mobile Development* dengan Expo 2. *Ticketing Request Website* : Membuat tab, halaman detail dan update
11	*Re-assignment* ke *external project* dan *training preparation* untuk *external project*.
12	Mengikuti *training preparation* untuk ke *external project* dan *transfer knowledge* situs web *ticketing request*
13	Membaca dan memahami berkas - berkas terkait *project external* yang diberikan.
14	*Start project external* (*requirement gathering*)

3.3 Uraian Pelaksanaan Magang

Pelaksanaan magang ini dilakukan penulis selama 6 bulan di mulai dari tanggal 1 Maret 2023 hingga 31 Juli 2023 di PT. Accelist Lentera IndonesiaUraian pelaksanaan magang ini penulis urai menjadi beberapa poin, meliputi;

3.3.1 Teknologi yang digunakan

Dalam proses pelaksanaan magang terutama di bidang IT diperlukan berbagai perangkat keras dan berbagai perangkat lunak yang harus digunakan. Hal ini dikarenakan untuk dapat membangun sebuah situs web, terdapat berbagai perangkat lunak yang digunakan penulis, antara lain:

1. Visual Studio 2019 dan Visual Studio Code v1.78.2
2. .NET SDK v5.0.408
3. .NET runtime v6.0.16
4. Next.JS v13.0.6
5. Postgre SQL v15.2
6. pgAdmin 4 v7.0
7. Tailwind CSS v3.2.4
8. Typescript v4.9.3
9. Node v18.15.0
10. git version 2.39.2.windows.1
11. Postman v10.14.8
12. Windows 11 Home (64 bit)
13. Google Chrome v114.0.5735.110 (64 bit)

Dengan bantuan perangkat keras, yaitu:

1. Processor Intel(R) Core(TM) i5-8265U CPU 1.60GHz
2. GPU NVIDIA GeForce MX230

3. RAM 12 GB LPDDR4X

4. HDD SATA 1TB

3.3.2 Perancangan sistem

Pada proses perancangan sistem, penulis membagi bagian perancangan ke dalam beberapa poin, yaitu; (1) *Requirement*, (2) *Use case diagram*, (3) *Activity Diagram*, dan (4) *Flowchart*

A. *Requirement*

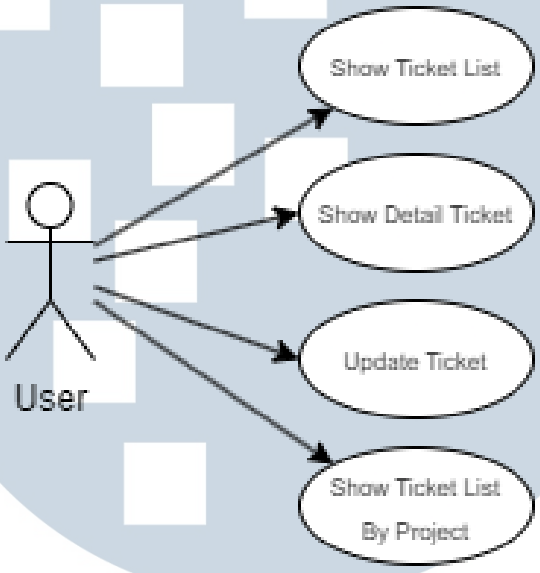
Pengguna dapat melihat tiga daftar tiket dari keseluruhan projek pada saat pertama kali melakukan *login* ke dalam situs web. Dimana halaman tersebut adalah halaman awal atau *landing page* serta terdapat informasi nama dari pengguna yang sedang *login* saat ini. Disamping daftar tiket terdapat informasi terkait jumlah informasi tiket yang sedang ada saat ini. Daftar tiket dan nama dari pengguna didapatkan situs web dari API yang dipanggil oleh situs web. Pada setiap daftar tiket yang tertera terdapat tombol *more* yang bila ditekan oleh pengguna akan memberikan informasi yang lebih lengkap dibandingkan sebelumnya dan terdapat tombol detail tiket.

Sejalan dengan penjelasan sebelumnya, pada saat tombol ditekan situs web akan mengarahkan pengguna ke halaman detail tiket yang menampilkan keseluruhan informasi secara lengkap. Serta terdapat tombol *edit* yang apabila ditekan akan memungkinkan pengguna untuk menambahkan atau memperbaharui data yang terdapat pada tiket. Data yang dirubah atau ditambahkan pengguna akan di *request* ke dalam API dan apabila respon yang didapatkan sukses maka informasi tiket pada *database* akan diperbaharui. Sebaliknya, apabila respon yang didapatkan adalah *error* maka situs web akan menampilkan kesalahan pada saat pengguna melakukan input.

Situs web diharapkan memiliki *side bar* yang memungkinkan pengguna melakukan perpindahan halaman. Menu yang dimiliki oleh *side bar* dapat mengarahkan pengguna ke halaman yang berisi daftar projek yang sedang berlangsung. Setiap projek yang ada di daftar dapat di tekan dan akan mengarahkan pengguna ke halaman yang memiliki *tab ticket dashboard* dan *reporting*. *Tab ticket dashboard* berisi daftar tiket sesuai projek pilihan yang didapatkan dari API

dengan situs web memberikan id yang dimiliki projek ke *back-end*. Pada setiap tiket memiliki tombol *more* dan *detail ticket* seperti yang terdapat pada *landing page*.

B. *Use Case Diagram*



Gambar 3.1. Use Case Diagram

Pada situs web *ticketing request* yang dibangun terdapat beberapa aktor yaitu admin, *developer*, dan *business analyst*. Dimana pada masing - masing aktor dapat melakukan berbagai hal yaitu menampilkan *ticket request* secara keseluruhan projek, menampilkan detail dari tiket, melakukan *update* pada tiket, dan juga menampilkan daftar tiket pada setiap projek secara terpisah satu sama lain. Hasil dari perancangan *use case diagram* untuk situs web *ticketing request* terutama pada fitur *landing page*, *detail and update ticket*, dan *ticket list by project* dapat dilihat pada gambar 3.1.

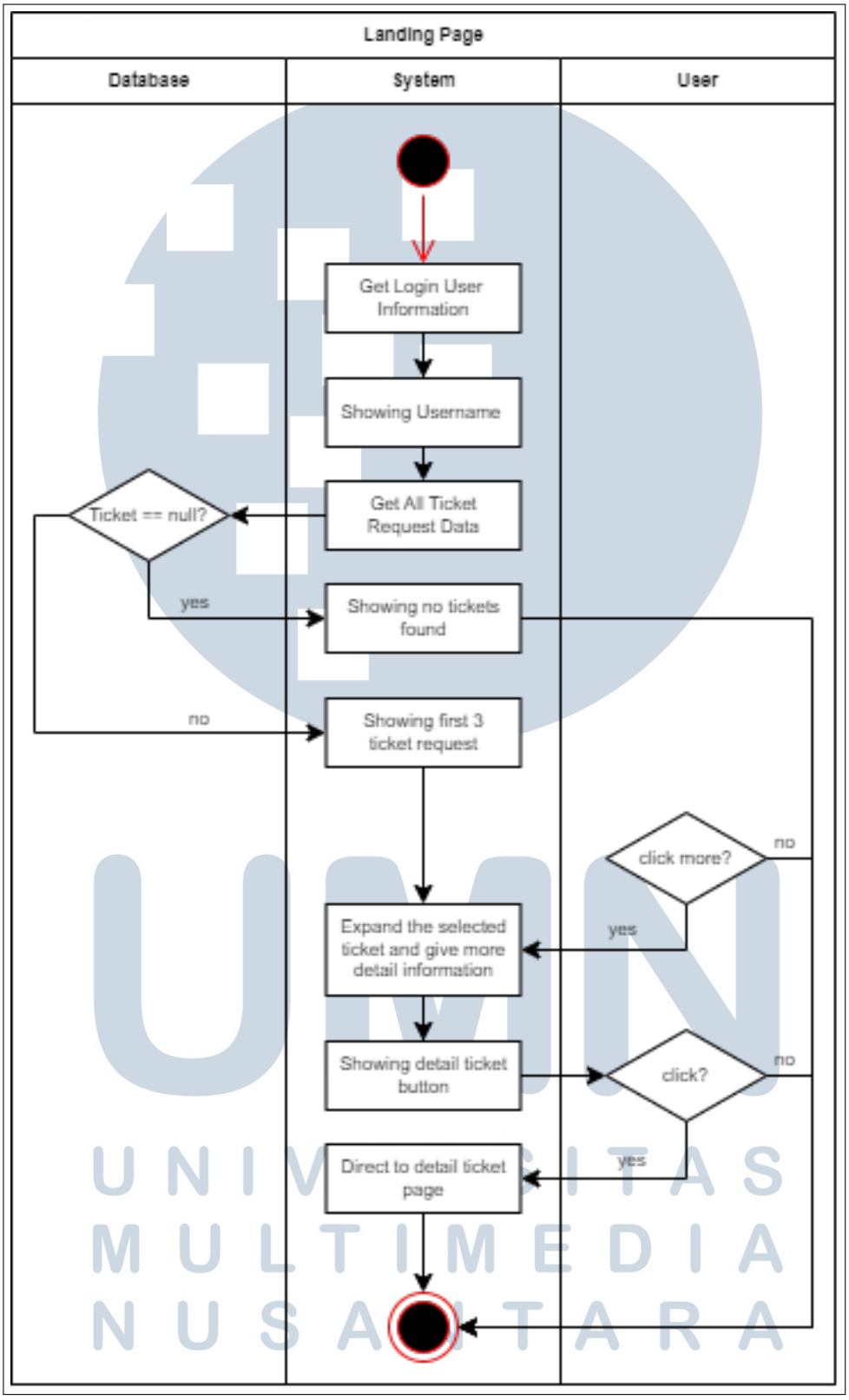
C. *Activity Diagram*

1. *Landing Page*

   Pada halaman awal atau *landing page* terdapat beberapa fitur yang dimana pada posisi awal pengguna sudah berhasil melakukan *login*. Situs web akan secara otomatis untuk menampilkan nama pengguna untuk membantu pengguna mengenali akun yang di *login* benar atau tidak. Situs web akan

secara otomatis mengambil semua data yang terdapat pada *database* bila data tidak tersedia maka situs web akan menampilkan *data not found*. Sebaliknya, bila data berhasil didapatkan maka situs web akan menampilkan data *request ticket* sebanyak 3 tiket di awal.

Pada setiap kolom tiket akan terdapat tombol *more* atau lebih banyak yang memiliki fungsi untuk membesarkan kolom tiket yang ditekan. Pembesaran pada kolom mengakibatkan pengguna dapat melihat data tiket yang lebih detail dari sebelumnya dan terdapat tombol *detail ticket* pada bawah kolom. Tombol *detail ticket* akan mengarahkan pengguna ke halaman dimana terdapat informasi lengkap soal tiket. Rancangan terhadap fitur halaman ini dapat dilihat pada gambar 3.2.

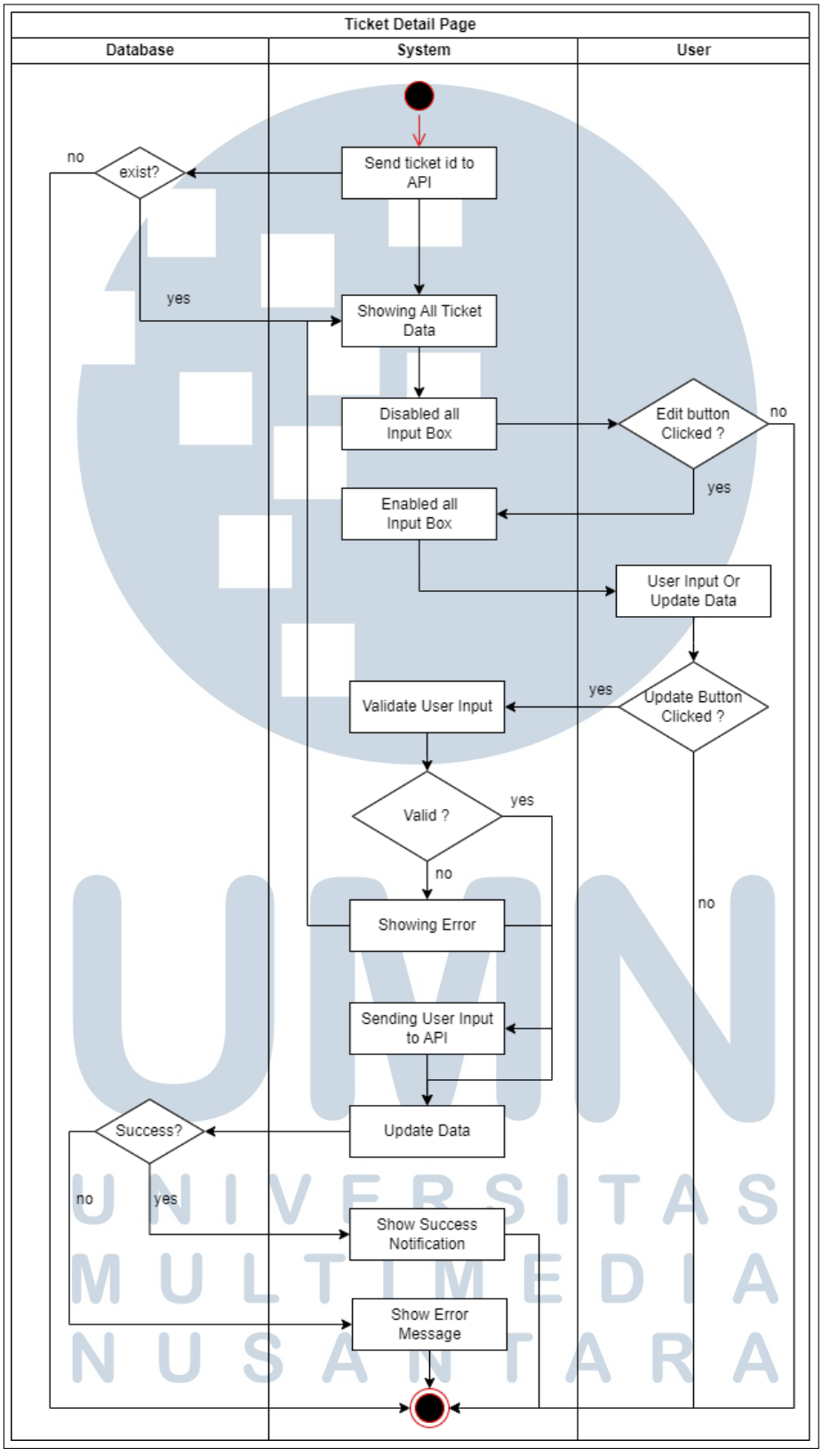


Gambar 3.2. Activity Diagram - Landing Page

2. *Ticket Detail* dan *Update Ticket*

Rancangan selanjutnya adalah *activity diagram* dari halaman detail tiket dimana pada awal proses situs web akan secara otomatis mengambil id dari tiket yang dipilih pengguna. Id dari tiket akan dikirimkan ke API untuk mendapatkan data lengkap milik tiket yang dipilih dan semua data dimasukkan ke dalam *input box* dan *dropdown* yang besifat *disabled*. Pada halaman ini terdapat tombol *edit* yang memungkinkan pengguna untuk melakukan perubahan data.

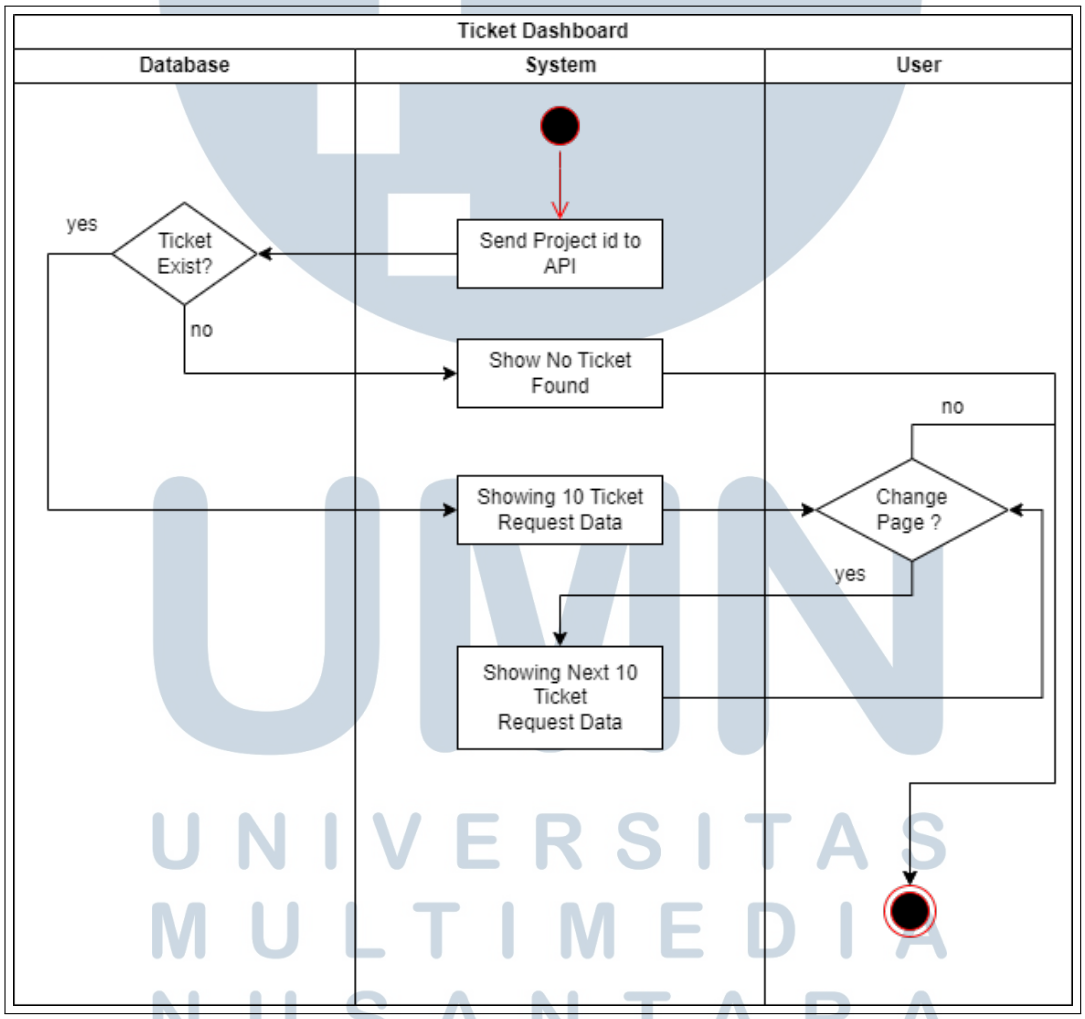
Sejalan dengan pernyataan sebelumnya, tombol *edit* akan membuat semua *input box* dan *dropdown* menjadi *enabled* atau dapat dilakukan *input*. Proses selanjutnya pengguna akan memasukkan data tambahan atau merubah data dari tiket yang dipilih sebelumnya setelah itu sistem akan melakukan validasi. Hasil dari validasi yang dilakukan akan menuju proses selanjutnya apabila sukses dan sebaliknya bila gagal maka situs web akan menampilkan kesalahan yang dilakukan. Proses terakhir dari halaman ini adalah setelah melewati proses validasi data akan dikirimkan ke API dan akan diperiksa apakah hasilnya sukses atau tidak.



Gambar 3.3. Activity Diagram - Detail and Update Ticket

3. *Ticket List by Project*

Pada halaman ini merupakan rancangan untuk menampilkan daftar tiket *request* yang dimiliki oleh masing - masing projek. Situs web akan mengambil id dari projek yang telah dipilih oleh pengguna sebelumnya dan kemudian dikirimkan ke API. Situs web akan menerima respon dari API, apabila terdapat tiket pada *database* maka situs web akan menampilkan tiket secara berhalaman dengan 10 tiket pada setiap halaman. Berbeda dengan hasil sebelumnya apabila hasil yang diterima tidak ada tiket yang terbaca maka situs web akan menampilkan *No ticket found*.



Gambar 3.4. Activity Diagram - Ticket List by Project

D. *Flowchart*

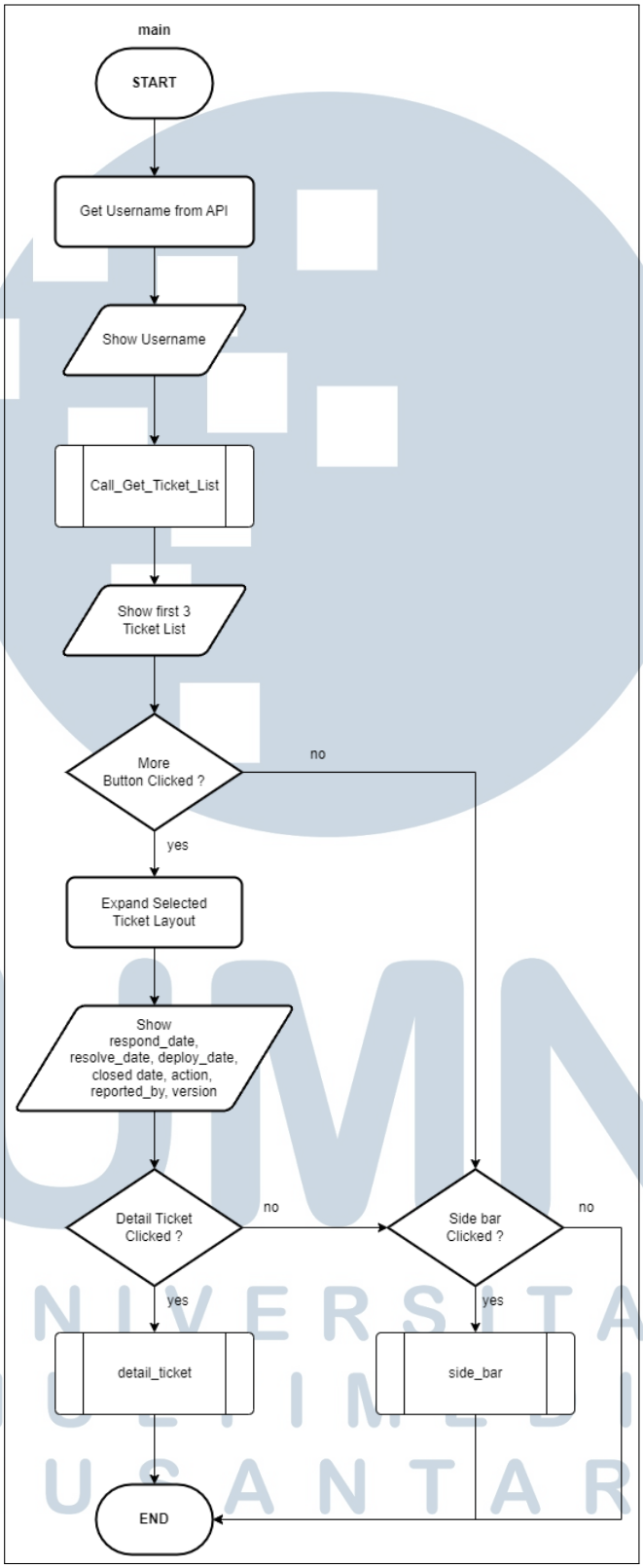
Flowchart merupakan diagram yang menggambarkan langkah - langkah dari sebuah sistem atau algoritma. Dengan pengunaan simbol - simbol yang terdapat pada *Flowchart* memudahkan *developer* saat sedang membangun sebuah aplikasi. Simbol yang dimiliki *flowchart* beragam contohnya persegi panjang yang menunjukkan proses, jajar genjang yang menampilkan input dan ouput, dan lain - lain. Berikut penulis sertakan *flowchart* dari perancangan setiap fitur pada situs web yang dikerjakan.

1. *Landing Page* atau *main*

   Pada *landing page* atau halaman awal, proses yang akan dilakukan oleh pengguna adalah mendapatkan nama pengguna dari API. Situs web akan melakukan proses selanjutnya yaitu mendapatkan 3 data tiket *request* pertama yang terdapat pada *database* dengan memanggil API. Hasil atau respon dari API dan tombol *more* akan ditampilkan oleh situs web sehingga dapat dilihat oleh pengguna pada halaman awal. Tombol *more* berfungsi untuk mendapatkan informasi tambahan seperti tanggal respon, tanggal deploy, pelapor, dan versi.

   Sejalan dengan pernyataan sebelumnya terdapat tombol *detail ticket* yang dapat ditekan oleh pengguna. Tombol *detail ticket* akan mengarahkan pengguna ke halaman *detail ticket*. Terdapat *side bar* pada *landing page* yang dapat mengarahkan pengguna ke halaman yang ingin dituju. Dimana terdapat menu *projects* dan *settings* di dalamnya dan terdapat berbagai fitur lainnya pada setiap menunya.

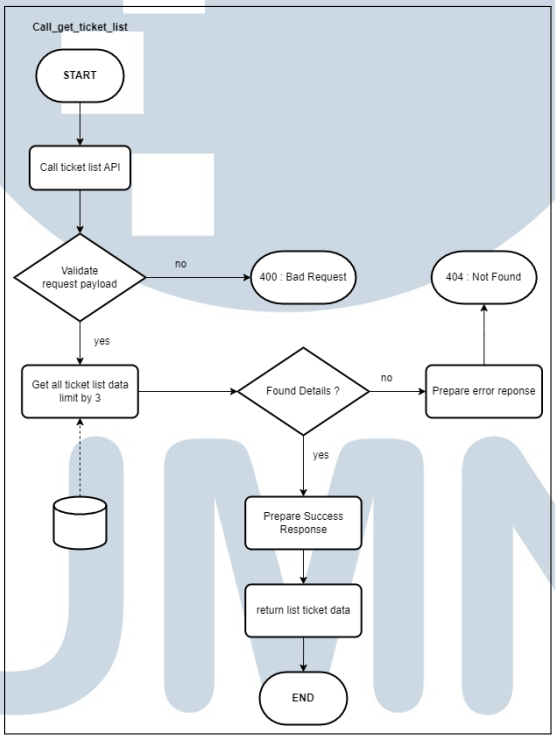
   Rancangan diagram alur atau *flowchart* dari *landing page* dapat dilihat pada gambar 3.5.



Gambar 3.5. Flowchart - Landing Page

2. *Get All Ticket List*

Pada perancangan *get all ticket list* adalah proses dari API saat mengambil data *ticket request* yang terdapat pada *database*. Pada proses awal dimana situs web melakukan *request payload* kepada API dan apa bila *request* yang diberikan tidak valid maka API akan mengirimkan respon 400 atau *Bad Request* sedangkan apabila valid maka sistem akan mengambil data dari *database* dan sistem akan melakukan pemeriksaan kembali apakah data tersedia atau tidak. Bila data tidak ditemukan maka sistem akan mengembalikan 404 atau *Not found* sebaliknya sistem akan mengirimkan data - data yang dibutuhkan oleh situs web.



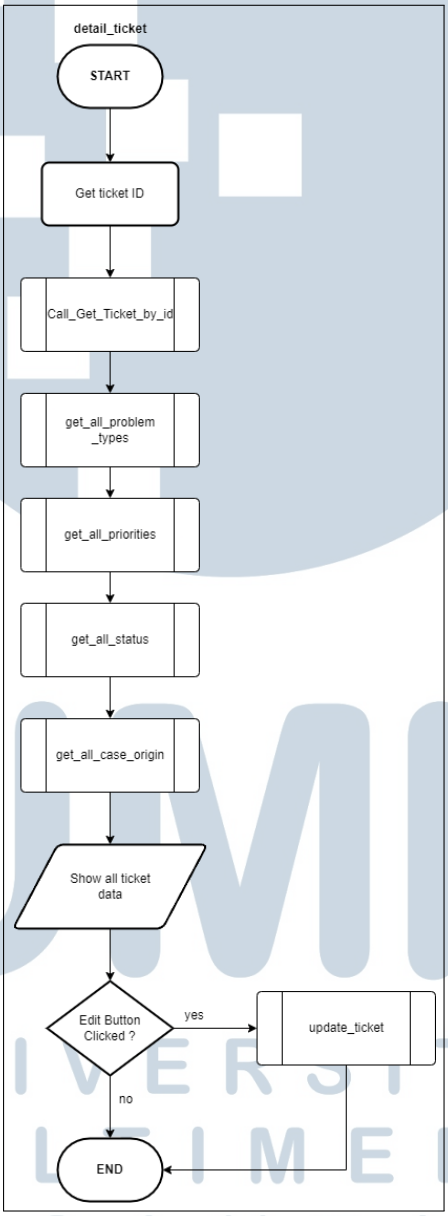
Gambar 3.6. Flowchart - Get All Ticket List

3. *Ticket Detail*

Pada perancangan halaman *ticket detail*, situs web akan mengambil id dari tiket. Secara otomatis situs web akan melakukan pemanggilan API yang dibutuhkan oleh halaman seperti data lengkap tiket, tipe masalah yang tersedia, tingkatan prioritas, daftar status, dan daftar *case origin*. Pemanggilan API untuk data lengkap tiket dibutuhkan dalam proses detail

tiket dan juga *update* tiket sedangkan pemanggilan API lainnya dibutuhkan pada saat melakukan *update*. Respon yang diberikan dari data lengkap tiket akan dimasukkan ke dalam input box dan ketika tombol *edit* ditekan maka situs web akan mengarahkan ke proses *update ticket*.

Alur dari perancangan halaman detail tiket dapat dilihat pada gambar 3.7.

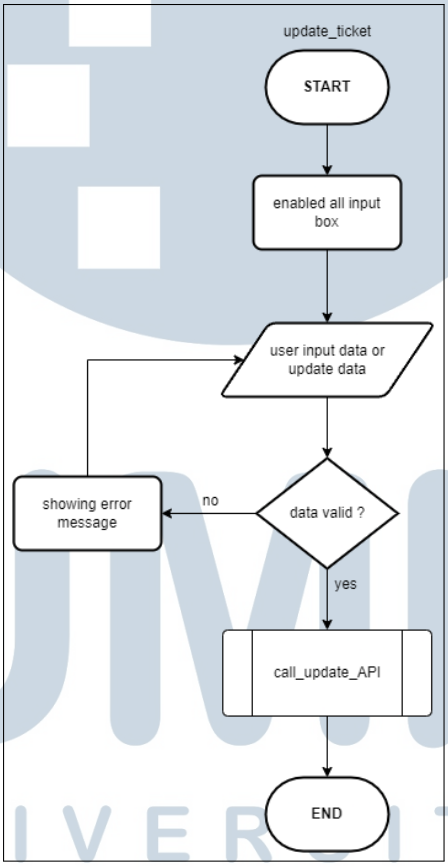


Gambar 3.7. Flowchart - Ticket Detail

4. *Update Ticket*

Pada fitur *update ticket*, sistem atau situs web akan secara otomatis membuat sifat dari semua input menjadi *enabled*. Hal ini membuat pengguna dapat memasukkan data - data yang diinginkan ataupun mengubah data yang sudah ada. Sistem akan mendapatkan inputan dari pengguna dan melakukan pemeriksaan terhadap data - data tersebut apakah valid atau tidak. Sejalan dengan kalimat sebelumnya, sistem akan menampilkan pemberitahuan bila terjadi kesalahan dalam input dan sistem akan melakukan *request* ke API bila data yang di input oleh pengguna valid.

Rancangan dari proses pembuatan fitur *update ticket* dapat dilihat pada gambar 3.8.



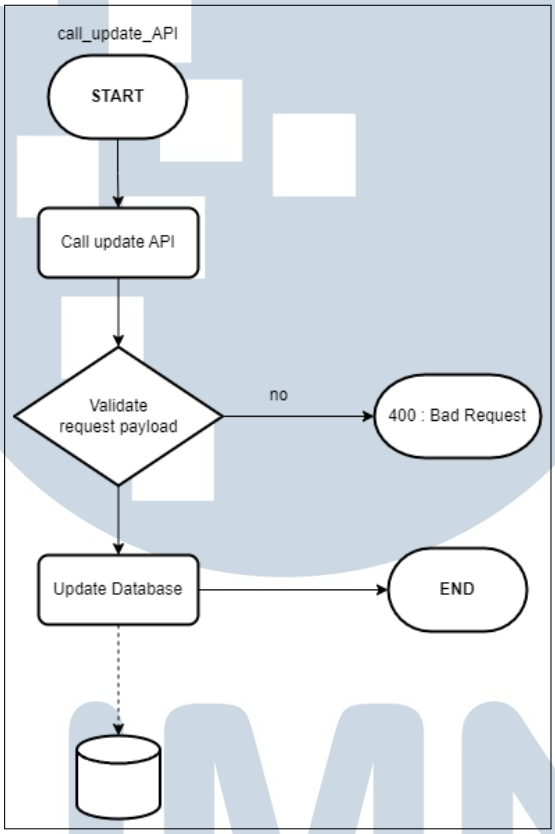
Gambar 3.8. Flowchart - Update Ticket

5. *Update* API

Diagram selanjutnya adalah diagram dari proses API ketika sistem sudah menyatakan input dari pengguna valid. Sistem akan mengirim *request* dari pengguna menjadi *payload request* yang akan di verifikasi oleh sistem. Saat

terjadi kesalahan dalam validasi maka sistem akan mengembalikan 400 *Bad Request* sehingga tidak ada yang dapat ditampilkan di situs web. Sebaliknya, apabila proses validasi dapat dilalui maka sistem akan melakukan *update* pada database milik tiket yang sedang dipilih.

Alur perancangan untuk *Update* API dapat dilihat pada gambar 3.9.



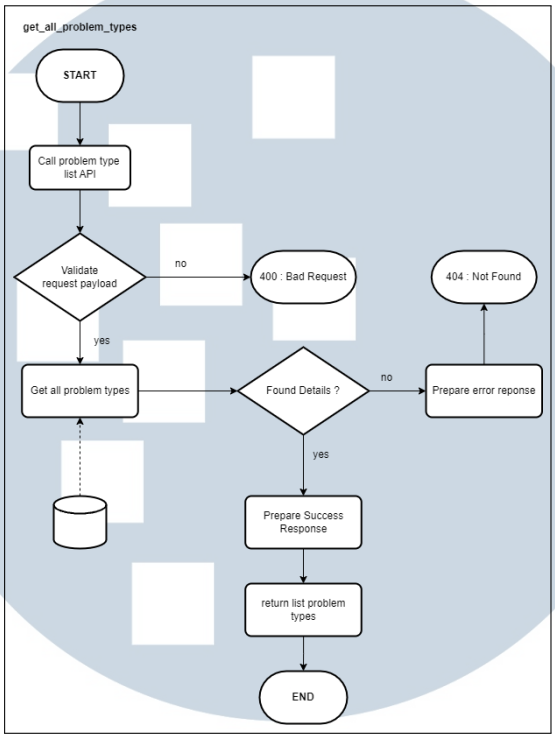
Gambar 3.9. Flowchart - Update API

6. *Get All Problem Types*

Fitur ini akan dipanggil ketika pengguna berada pada halaman *detail ticket* dengan tujuan untuk mendapatkan daftar permasalahan yang dapat terjadi dalam pengerjaan projek. Saat pertama kali halaman *detial ticket* dijalankan maka sistem akan secara otomatis melakukan *request* terhadap *back-end* dan mengambil daftar data yang ada pada *database*. *Back-end* akan melakukan validasi terkait *request* yang dikirimkan apa bila terjadi *error* maka sistem akan mengembalikan *error* sedangkan berhasil maka sistem akan mengembalikan daftar jenis permasalahan yang ada. Daftar yang

didapatkan akan ditampilkan nantinya pada proses *update*.

Perancangan proses terkait *problem types* dapat dilihat pada gambar 3.10.



Gambar 3.10. Flowchart - Get All Problem Types

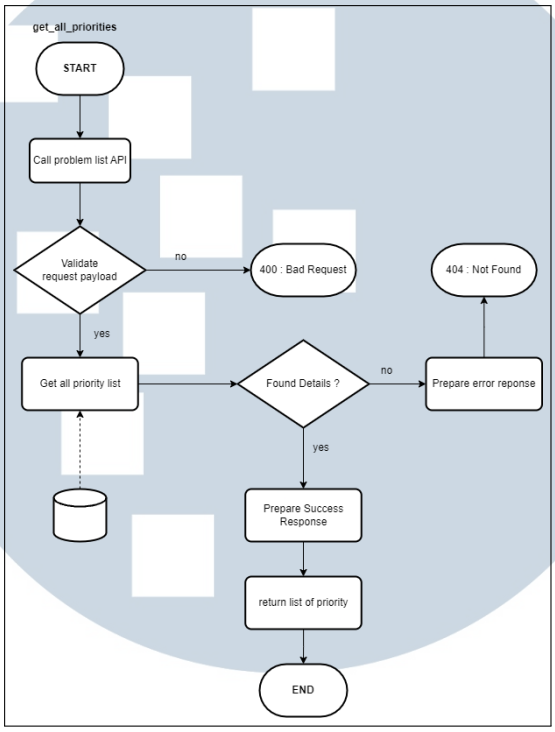
7. *Get All Priority*

Sejalan dengan pembahasan sebelumnya, fitur ini akan dipanggil pada saat pengguna memasuki halaman *detail ticket*. Tujuan dari fitur ini adalah untuk mendapatkan daftar tingkatan prioritas tiket untuk segera dikerjakan. Daftar dari tingkatan prioritas akan ditampilkan pada *dropdown* pada bagian *update*. Proses yang dilakukan adalah sistem melakukan *request* terhadap *back-end* untuk mendapatkan daftar tingkatan prioritas secara keseluruhan dari *database*.

Hasil yang didapatkan pada proses sebelumnya akan di validasi dan pada proses validasi ketika menemukan kesalahan maka sistem akan mengembalikan *Not found* yang tidak akan menampilkan apapun nantinya ke pengguna. Berlainan dengan hasil validasi sebelumnya, ketika hasil validasi menunjukkan valid maka sistem akan mengembalikan daftar prioritas secara keseluruhan. Hasil pengembalian ditampilkan pada *dropdown* yang terdapat

di fitur *update* sebelumnya.

Hasil perancangan untuk proses mendapatkan daftar tingkatan prioritas dapat dilihat pada gambar 3.11.

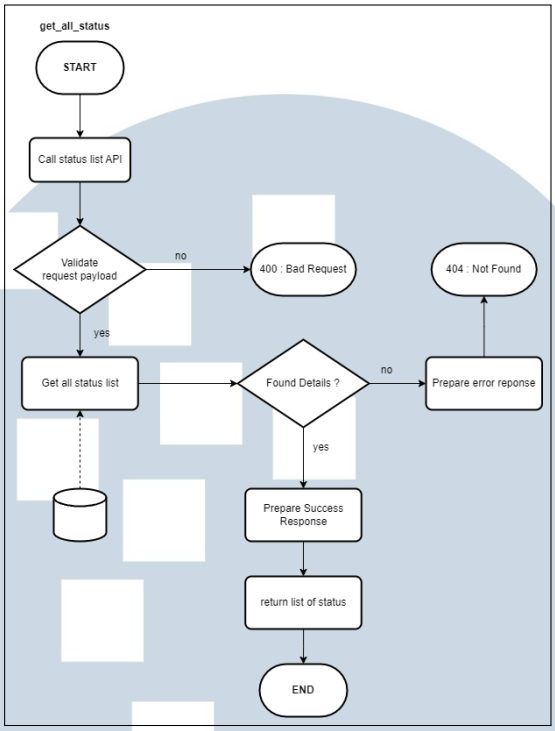


Gambar 3.11. Flowchart - Get All Priority

8. *Get All Status*

Berdasarkan 2 proses sebelumnya, proses mendapatkan daftar status ini memiliki alur proses yang serupa. Tujuan dari proses ini adalah mendapatkan daftar status dari *database* untuk dapat ditampilkan kepada pengguna. Proses yang dilalui yaitu sistem melakukan *request* kepada *back-end* untuk mendapatkan keseluruhan daftar status yang ada pada *database*. Proses tersebut akan dilakukan validasi oleh sistem untuk mendapatkan data yang sesuai atau data yang valid. Data yang bersifat valid akan dikembalikan sistem sehingga dapat ditampilkan kepada pengguna.

Hasil dari proses perancangan untuk mendapatkan daftar status dapat dilihat pada gambar 3.12.



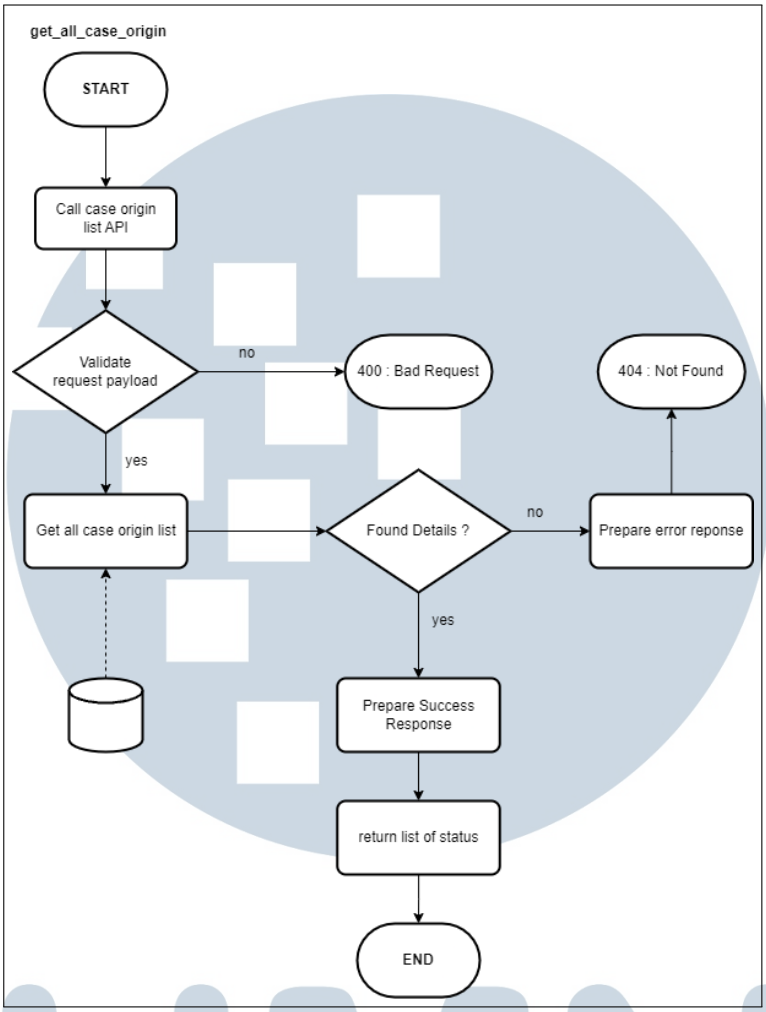
Gambar 3.12. Flowchart - Get All Status

9. *Get All Case Origin*

Sejalan dengan pembahasan sebelumnya, proses ini adalah proses untuk mendapatkan daftar keseluruhan *case origin* yang ada pada *database*. Fungsi dari*Case origin* adalah untuk menunjukkan kasus permasalahan yang terjadi pertama kali diinformasikan melalui media apa. Proses yang terjadi adalah sistem melakukan *request* kepada *back-end* dan *back-end* akan melakukan validasi *request*.

Hasil dari validasi bila menunjukkan tidak sesuai maka sistem akan mengembalikan *Bad Request*. Sebaliknya, sistem akan mengambil daftar *case origin* dari *database* ketika *request* yang dikirimkan valid. Sebelum daftar dikembalikan sistem akan melakukan validasi kembali terkait data yang didapat ketika *request* yang dilakukan tidak ditemukan maka sistem akan mengembalikan *not found*.

Hasil dari perancangan proses untuk mendapatkan daftar keseluruhan dari *case origin* dapat dilihat pada gambar 3.13.

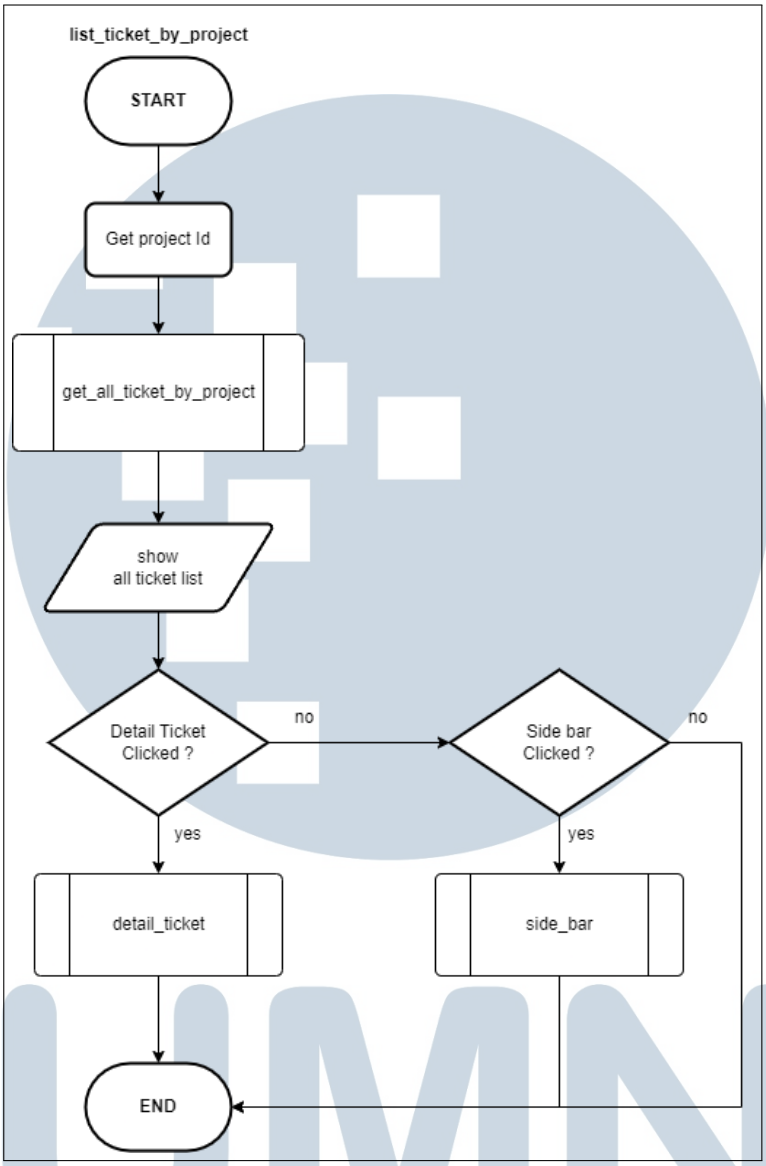


Gambar 3.13. Flowchart - Get All Case Origin

10. *List Ticket by Project*

Pada halaman ini merupakan halaman yang akan tampil setelah pengguna memilih salah satu dari daftar projek yang ditampilkan. Sistem akan mengambil informasi terkait projek yaitu id projek untuk di *request* ke API atau *back-end*. Hasil yang didapatkan atau respon dari API akan ditampilkan di halaman *list ticket* dengan fitur yang sama dengan yang ada pada *landing page* sebelumnya. Fitur tersebut seperti tombol *more*, tombol *detail ticket*, dan *side bar* yang tersedia disetiap halaman yang ada.

Untuk perancangan alur dari halaman ini dapat dilihat pada gambar 3.14.



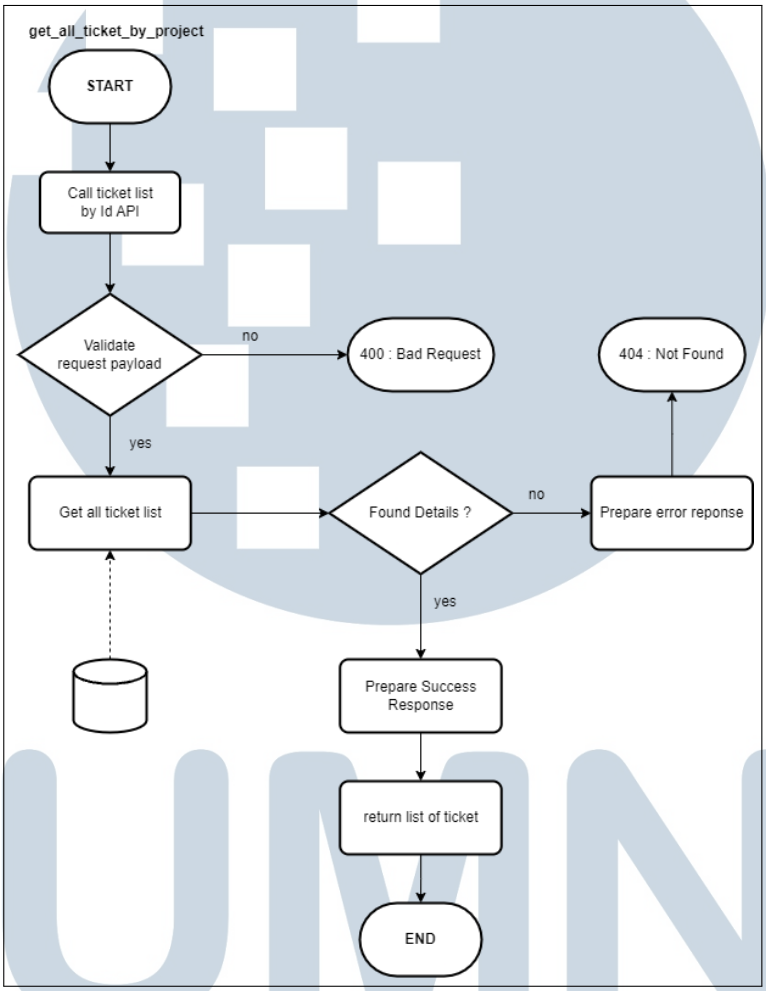
Gambar 3.14. Flowchart - List Ticket by Project

11. *Get List Ticket by Project*

Proses ini adalah proses untuk mendapatkan daftar tiket yang dimiliki oleh projek yang memiliki id yang sama dengan id yang di *request* oleh sistem. Dimana sistem akan mengirimkan *request* kepada *back-end* dan di validasi. Ketika validasi tidak sesuai maka sistem akan mengembalikan *Bad Request*. Bertolak belakang dengan validasi yang sesuai dimana sistem akan mengambil data dari *database* dan ketika *halaman* yang dituju ke API tidak ditemukan maka *back-end* akan mengembalikan *not found*. Sebaliknya,

ketika data ditemukan maka *back-end* akan mengembalikan daftar tiket sesuai dengan projek yang dipilih.

Diagram dari proses mendapatkan daftar tiket berdasarkan projek dapat dilihat pada gambar 3.15.



Gambar 3.15. Flowchart - Get List Ticket by Project

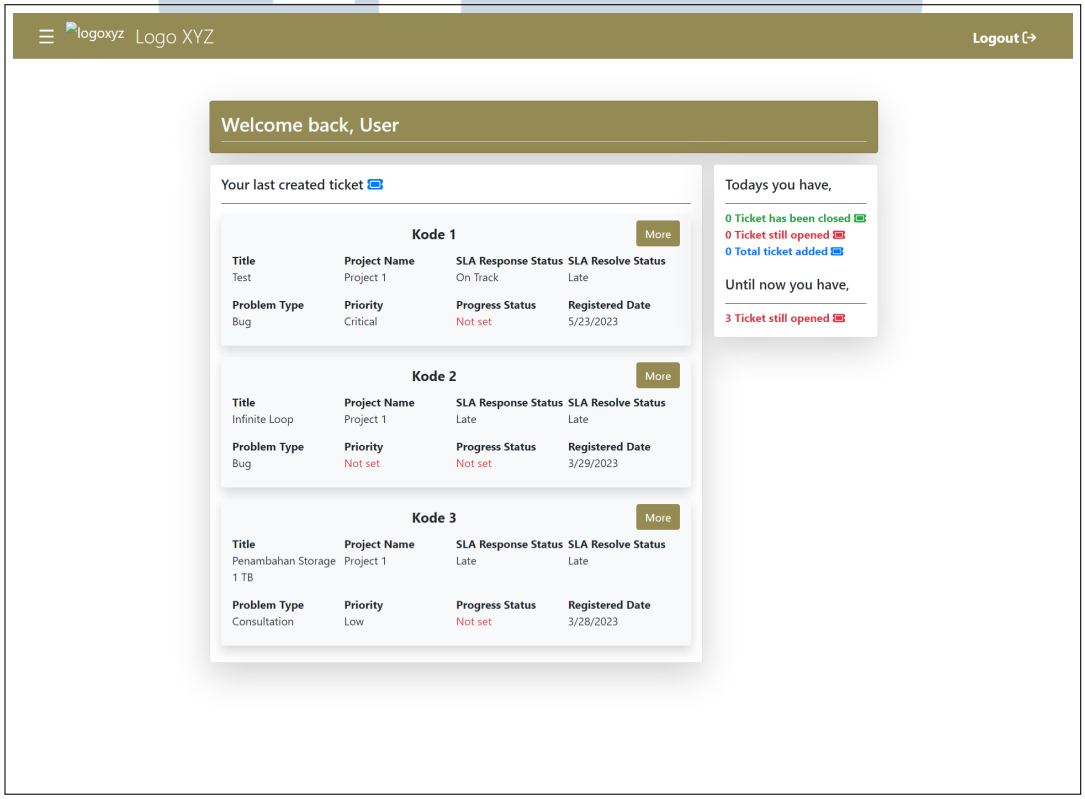
3.3.3 Implementasi

Dalam proses implementasi, posisi pembuatan *back-end* dan API telah dirampungkan terlebih dahulu. API yang telah dibuat sebelumnya telah diperiksa dengan melakukan CRUD (*Create*, *Read*, *Update*, *Delete*) secara manual menggunakan aplikasi *Postman*. Hasil yang didapatkan API dapat berfungsi dengan baik karena memberikan respon yang diinginkan. Proses yang dilakukan setelah pemeriksaan API adalah membuat tampilan situs web sesuai dengan yang sudah

dirancang.

A. Pembuatan *landing page*

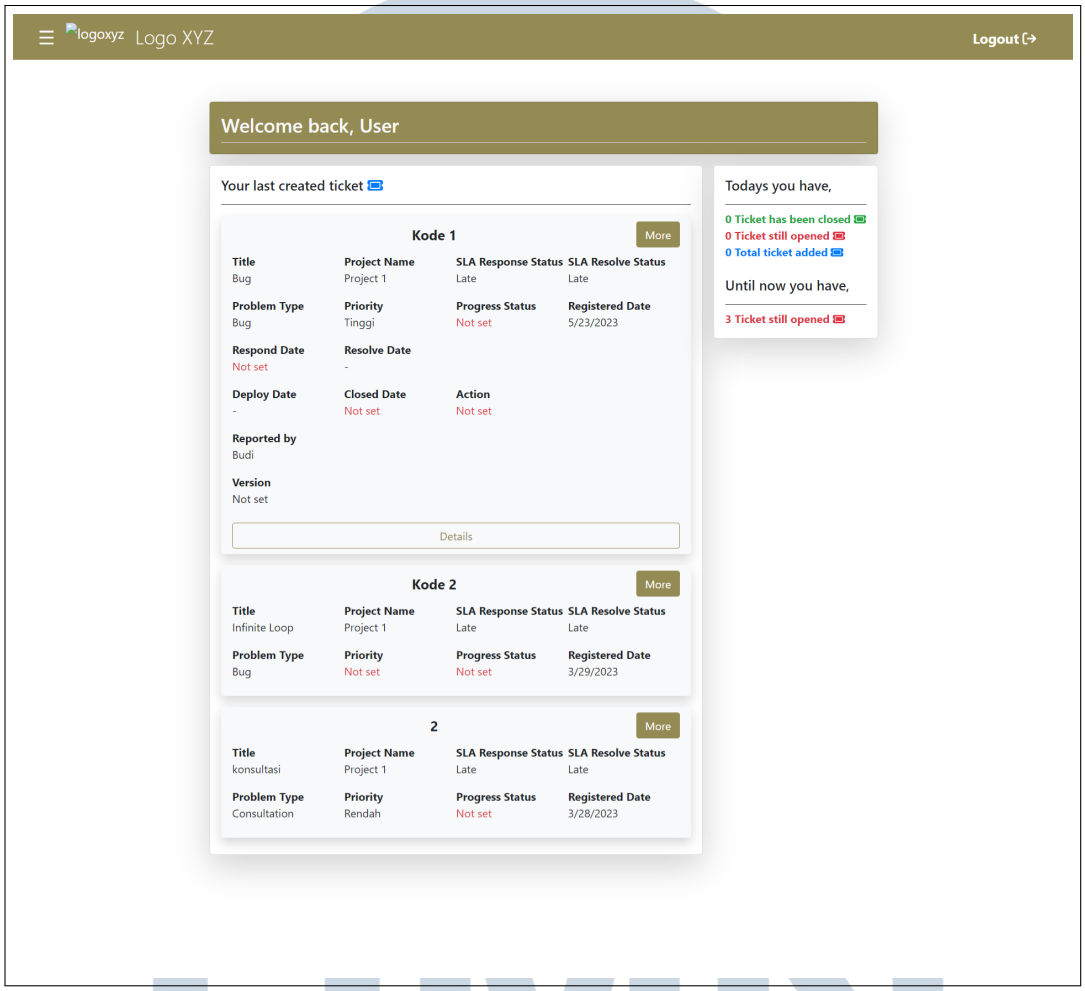
Pada pembuatan *landing page*, sesuai dengan rancangan yang sudah dijelaskan sebelumnya *landing page* memiliki berbagai fitur atau fungsi. Fitur yang dimiliki oleh *landing page* adalah *landing page* dapat mendapatkan nama dari pengguna yang sedang *login*, terdapat daftar 3 tiket dari berbagai projek teratas, terdapat *sider bar*, dan jumlah tiket yang sedang dalam posisi *open*. Hal ini dapat dilihat pada gambar 3.16.



Gambar 3.16. Halaman awal situs web

Seperti yang dapat dilihat pada gambar 3.16. informasi tiket yang diberikan pada saat pengguna masuk ke halaman *landing page* atau halaman awal adalah *title*, *project name*, *SLA response status*, *SLA revoke status*, *problem type*, *priority*, *progress status*, dan *registered date*. Segala informasi yang disebutkan sebelumnya merupakan hasil respon dari API yang dipanggil yaitu API untuk mendapatkan daftar tiket secara keseluruhan. Terdapat nama dari pengguna yang bernama ”user”

pada gambar yang didapatkan dari pemanggilan API yang mengembalikan respon nama dari pengguna yang sedang dalam posisi *login*.



Gambar 3.17. Tampilan setelah tombol *more* ditekan

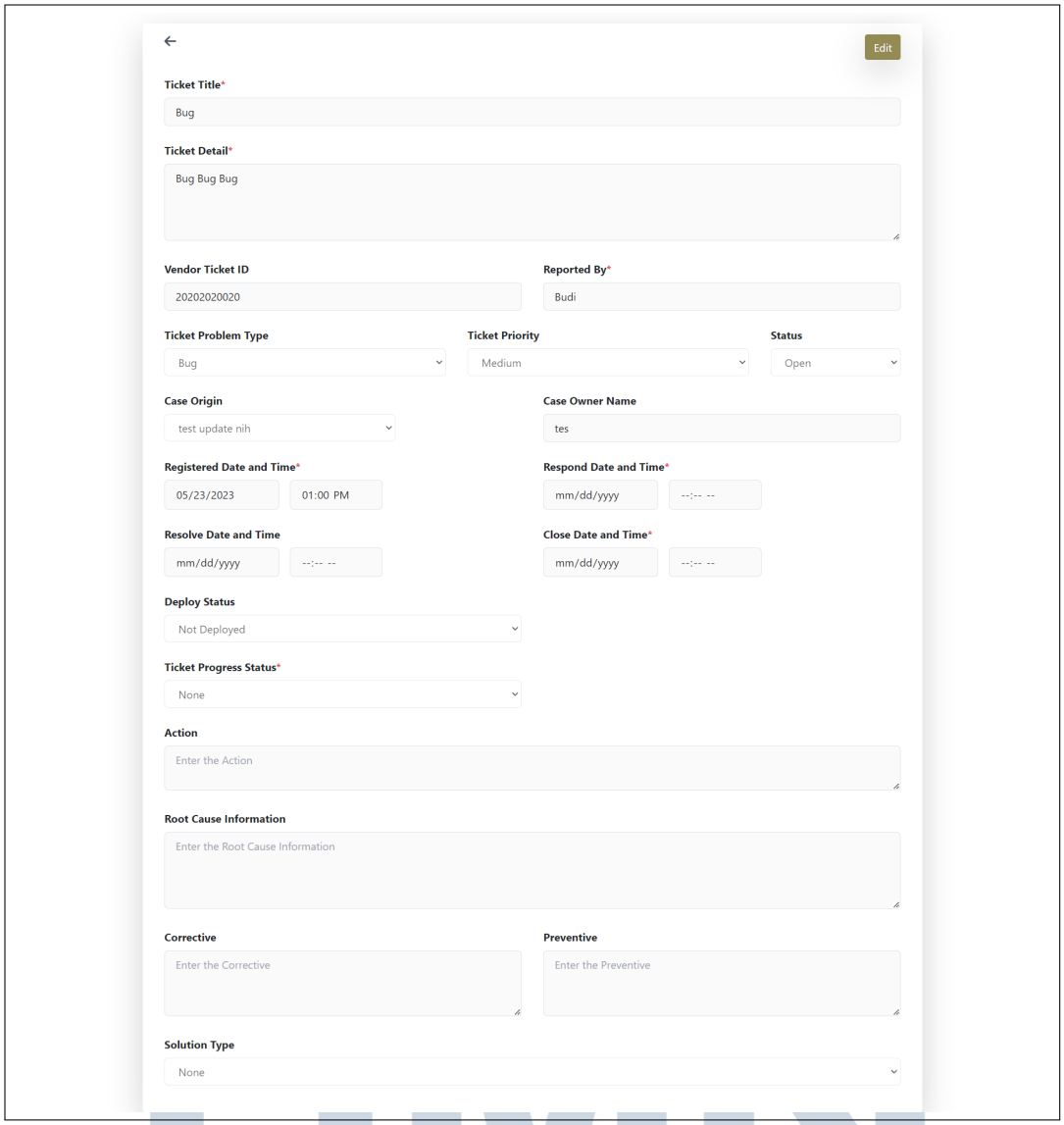
Pada gambar 3.17. dapat dilihat bahwa saat tombol *more* yang ada pada setiap tiket maka secara otomatis situs web akan membesarkan kolom pada tiket yang dipilih. Serta dapat dilihat bahwa terdapat tambahan informasi terkait yaitu *respond date*, *resolve date*, *deploy date*, *closed date*, *action*, *reported by*, dan *version*. Pada fitur ini terdapat tombol tambahan yang akan terlihat ketika pengguna menekan tombol *more* yang terdapat pada tiket yang dipilih yaitu tombol *details*. Tombol details berfungsi untuk mengarahkan pengguna ke halaman yang menampilkan informasi tiket secara lengkap. Halaman yang bertugas untuk menampilkan informasi secara lengkap adalah halaman *ticket detail*.

B. Pembuatan Ticket Detail

Halaman *ticket detail* merupakan halaman yang berfungsi untuk menampilkan informasi tiket yang sangat lengkap. Tepat setelah pengguna memasuki halaman *ticket detail*, sistem secara otomatis akan memanggil berbagai API yang dibutuhkan seperti *ticket data* yang didapatkan dari melakukan *request* id tiket yang didapatkan dari URL, API *get ticket problem type* yang akan mengembalikan daftar tipe permasalahan yang ada, *get all priority* yang mengembalikan daftar tingkatan prioritas tiket yang harus segera dikerjakan, *get all status* yang akan mengembalikan daftar status yang ada yaitu *open* dan *closed*, *get all case origin* yang dimana untuk mendapatkan daftar *case origin* yang dimiliki, dan *get all ticket progress status* untuk mendapatkan daftar progres status yang tersedia.

Dimana pada setiap informasi yang ditampilkan dimasukkan ke dalam *input text* dan *dropdown*. Seperti yang dapat dilihat pada gambar 3.18. Informasi yang ditampilkan pada *ticket detail*, meliputi.

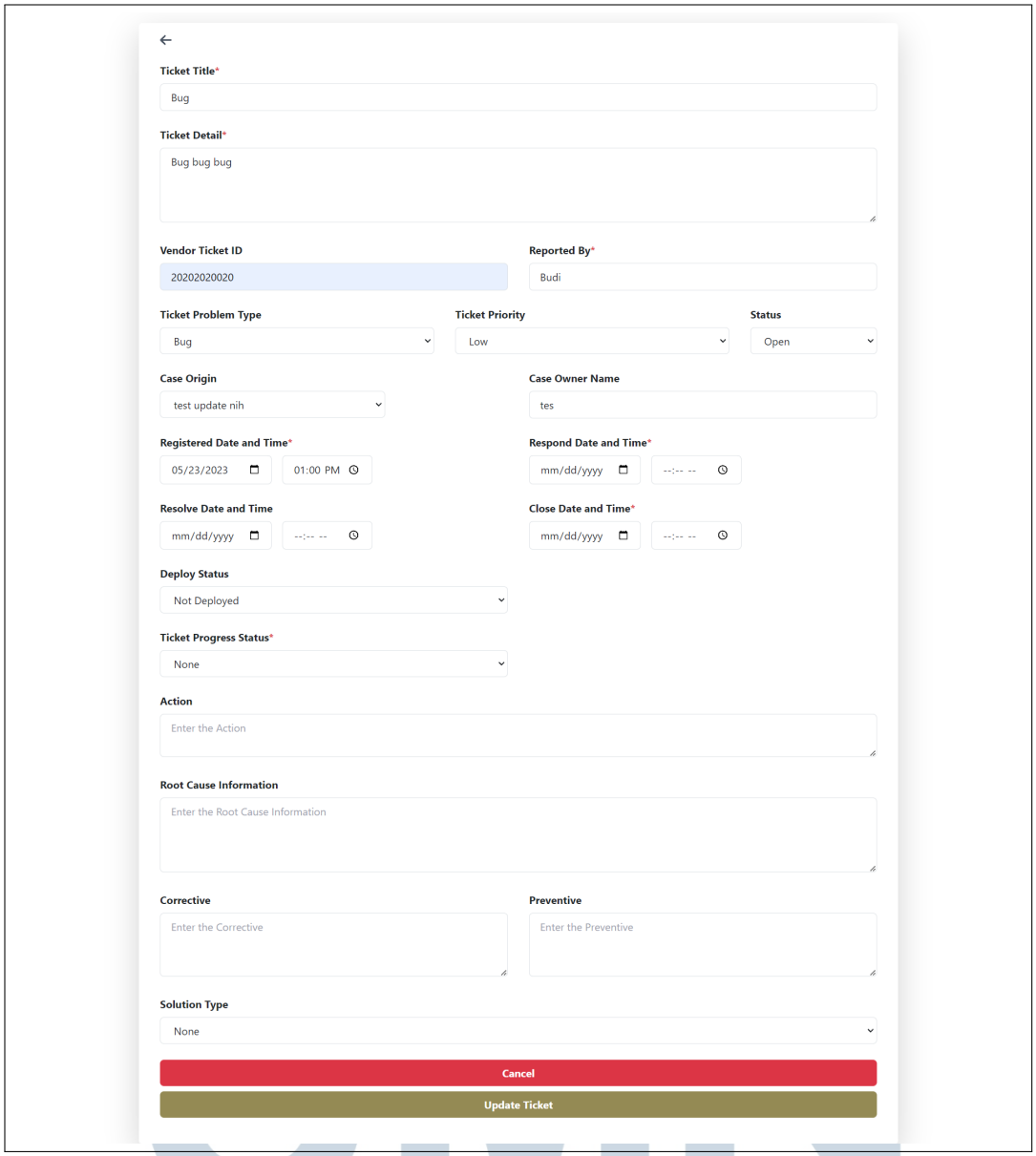
1. *Ticket title*
2. *Details*
3. *Vendor ticket id* dan *reported by*
4. *Ticket problem type*, *priority*, dan *status*
5. *Case origin* dan *case owner name*
6. *Registered date and time* dan *respond date and time*
7. *Resolved date and time* dan *closed date and time*
8. *Deploy status*
9. *Ticket progress status*
10. *Action*
11. *Root cause information*
12. *Corrective* dan *preventive*
13. *Solution type*



Gambar 3.18. Tampilan Detail Page

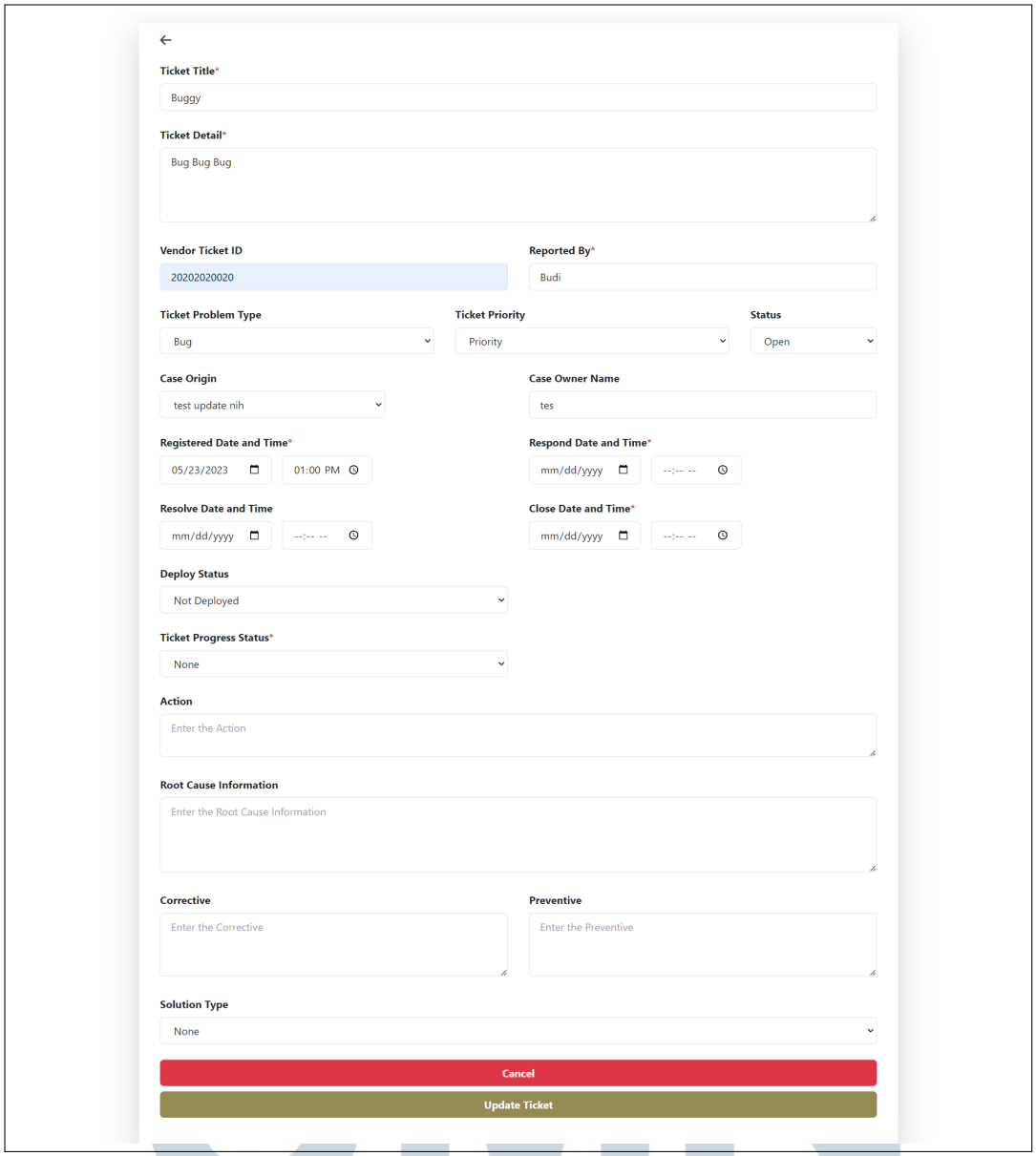
Keseluruhan informasi pada *ticket detail* bersifat *disabled* yang dimana pengguna tidak dapat melakukan perubahan pada data tersebut. Terdapat tombol *edit* yang terdapat pada pojok kanan atas halaman yang apabila ditekan secara otomatis semua *input text* dan *dropdown* menjadi *enabled*. Hal ini menandakan bahwa seluruh kolom yang ada dapat dirubah ataupun diisi oleh pengguna yang sedang *login*. Serta tombol *edit* yang sebelumnya ada akan disembunyikan dari penglihatan pengguna sehingga tidak dapat terlihat kembali.

Berdasarkan pernyataan diatas, perubahan - perubahan yang terjadi pada halaman dapat dilihat pada gambar 3.19. yang menunjukkan halaman *update ticket*.



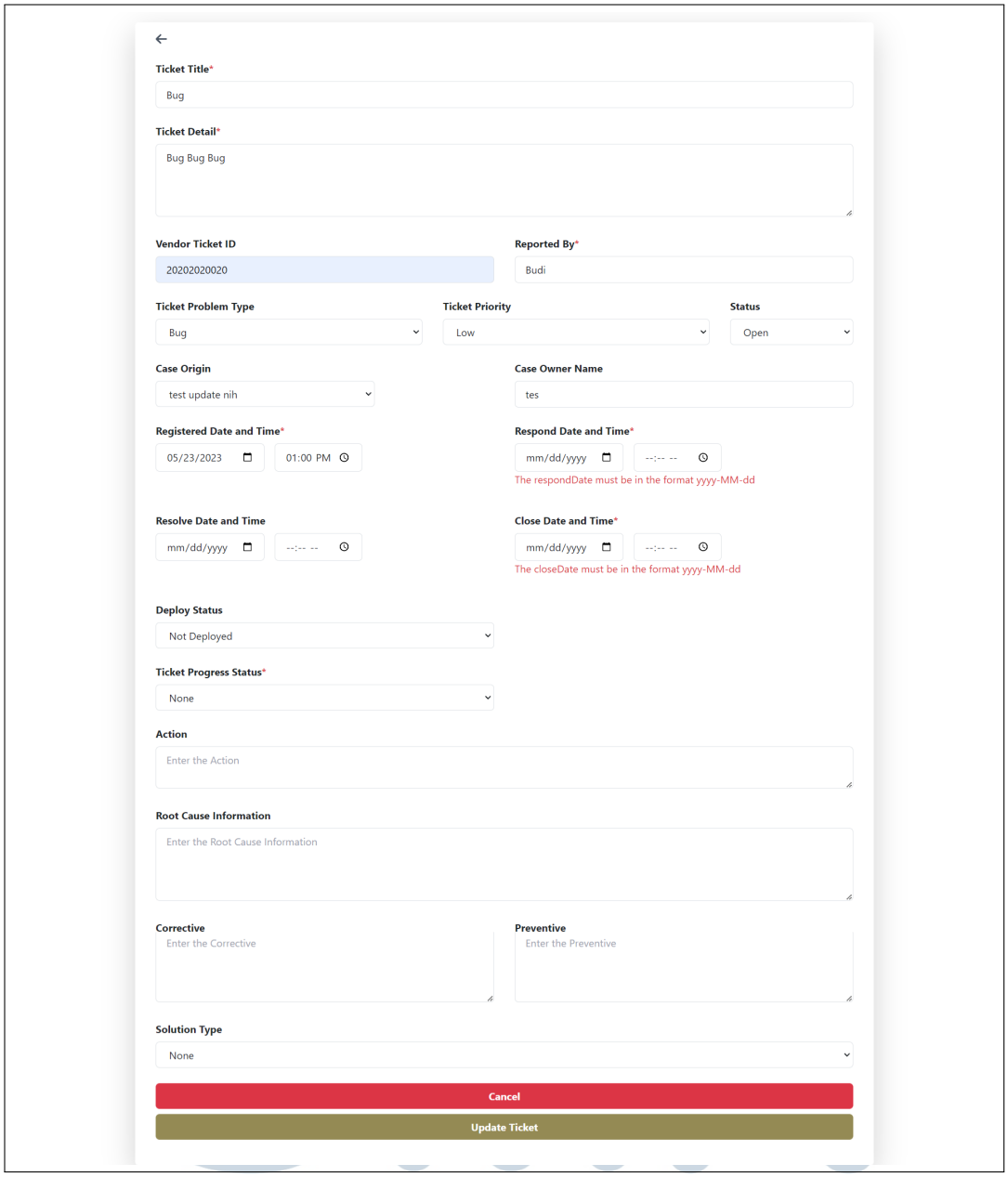
Gambar 3.19. Tampilan Update Page

Seperti yang ada pada gambar yaitu terdapat beberapa *form* yang sifatnya wajib dengan ditandai simbol (*) yaitu pada *title*, *details*, *reported by*, *registered date and time*, *respond date and time*, *close date and time*, dan *ticket progress status*. Pengguna dapat mengubah keseluruhan informasi yang ada dan tersedia seperti melakukan perubahan *title* ataupun untuk mengganti status dari tiket. Hal ini dibuktikan pada gambar 3.20. yaitu contoh tampilan bila pengguna melakukan perubahan pada *ticket title*.



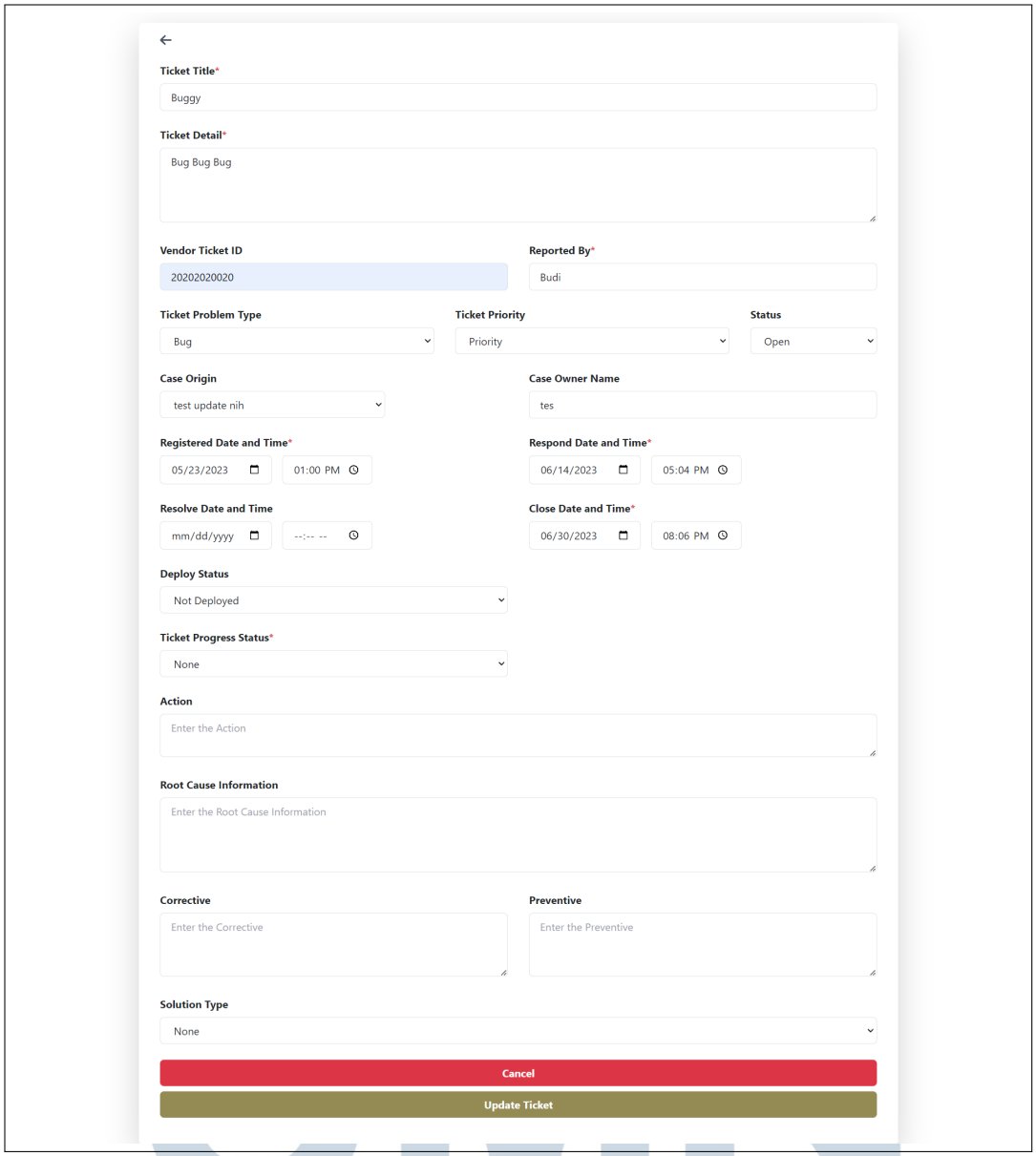
Gambar 3.20. Tampilan Saat Pengguna Melakukan Perubahan

Dalam melakukan proses *update*, sistem dibantu dengan Zod Resolver yang dimana membantu untuk melakukan validasi pada data - data yang di *input* oleh pengguna. Terdapat banyak fungsi validasi yang disediakan seperti tipe data yang diharuskan, minimum panjang karakter, maximum panjang karakter, dan menentukan apakah *form input* wajib atau tidak. Zod Resolver menyediakan *developer* untuk melakukan *custom message* untuk setiap validasi yang diinginkannya. Contoh dari hasil validasi tersebut disertakan pada gambar 3.21.



Gambar 3.21. Tampilan Validasi pada *Update Page*

Seperti yang dapat dilihat pada gambar 3.21. terdapat validasi pada *respond date and time* dan *close date and time*. Hal ini terjadi dikarenakan kolom yang bersifat wajib tidak di isi atau dikosongkan. Pesan pemberitahuan yang ditampilkan sesuai dengan apa yang penulis inginkan yaitu untuk memberitahu pengguna format yang benar untuk kolom *date* dan *time*.

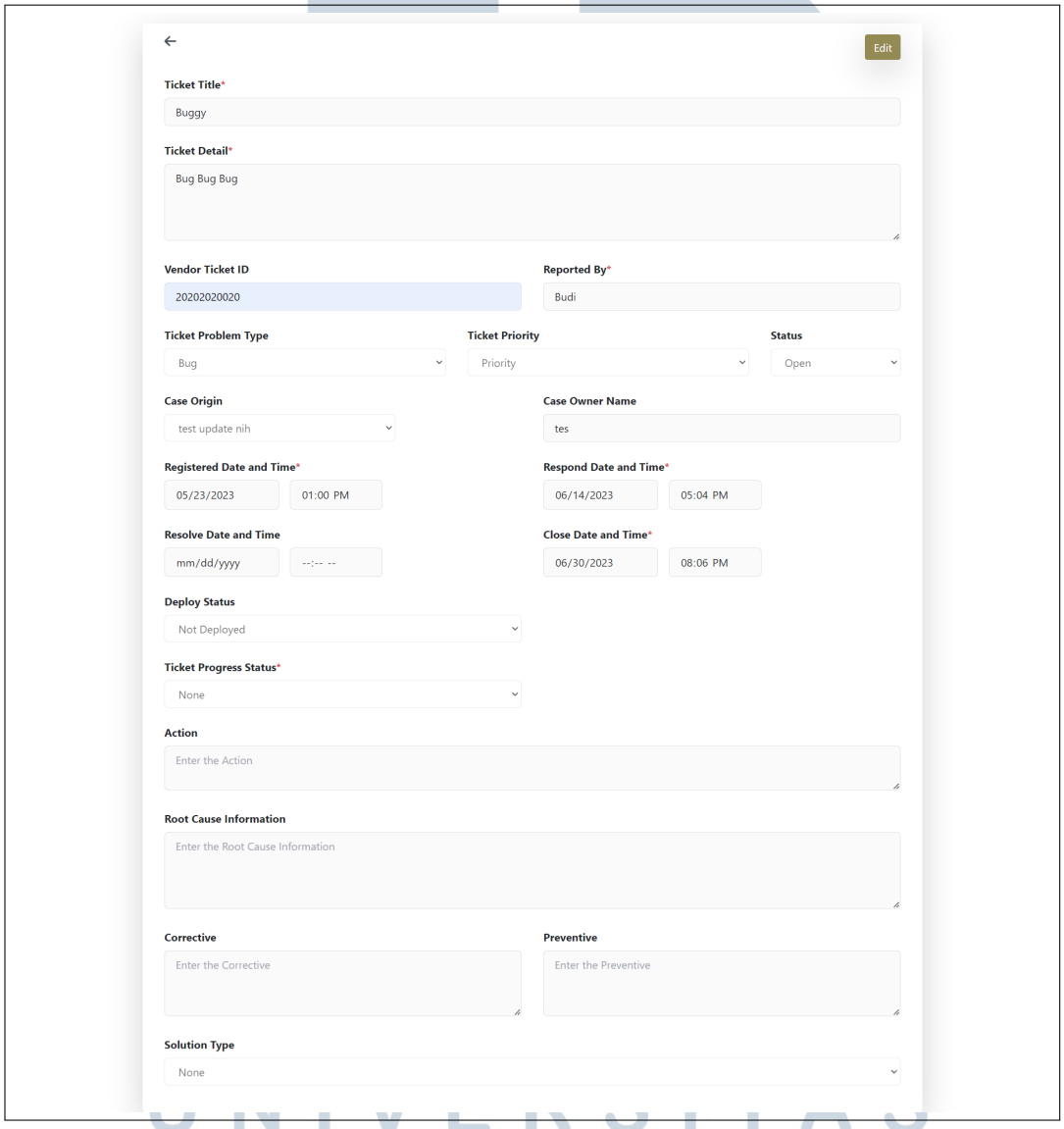


Gambar 3.22. Tampilan Pembenaran Input Pengguna

Seperti yang dapat dilihat pada gambar 3.22. terlihat jelas bahwa validasi bekerja dengan baik. Dimana pada saat pengguna melakukan pembenaran pada kesalahan yang sebelumnya dilakukan secara otomatis pesan pemberitahuan yang ditampilkan akan hilang. Hal ini dibuktikan pada gambar dimana pengguna telah memilih *close date and time* dan *respond date and time*. Setelah melewati seluruh proses validasi pada *form update* maka sistem akan mengirimkan hasil perubahan dan penambahan pengguna ke dalam *payload request* yang nantinya akan dikirim ke dalam *update ticket* API. Ketika respon yang dikembalikan API sukses atau

menunjukkan status kode 200 maka data sudah berhasil dirubah atau di *update* ke dalam *database*.

Berikut penulis sertakan gambar setelah proses *update* berhasil dilakukan pada gambar 3.23.



Gambar 3.23. Tampilan Update Berhasil Dilakukan

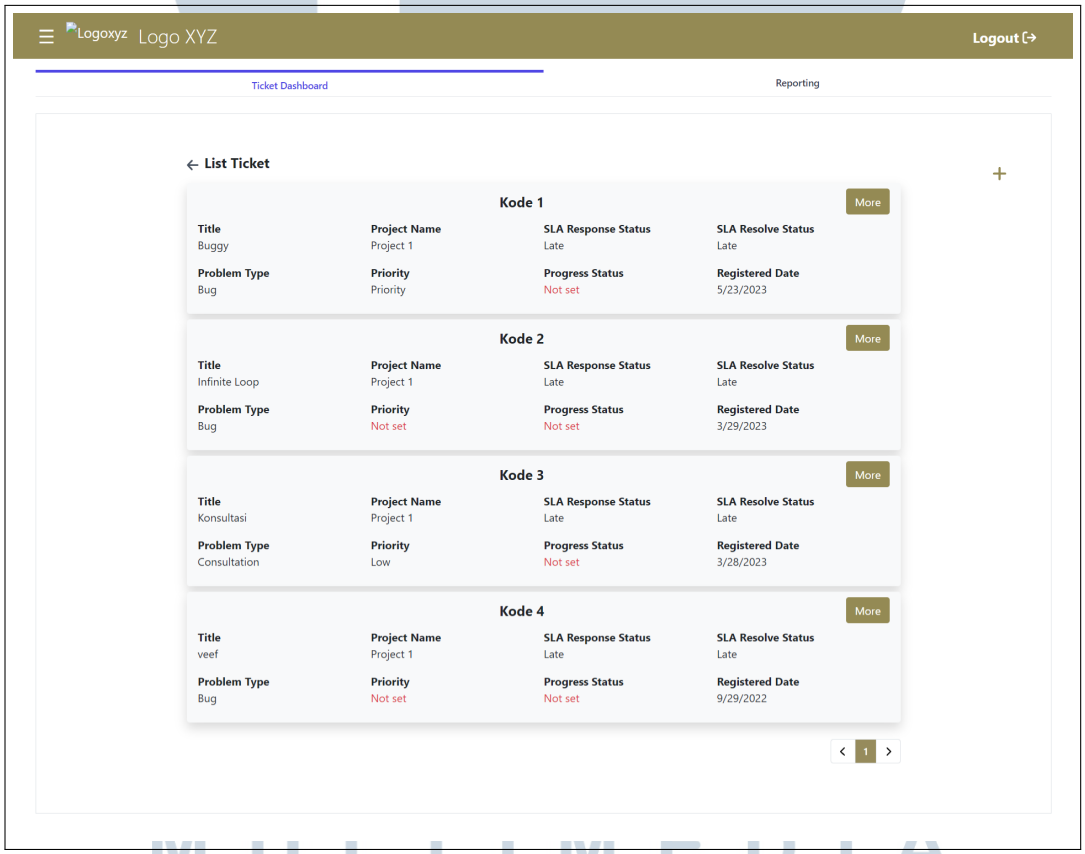
Seperti yang dapat dilihat pada gambar 3.23. dapat dilihat bahwa proses perubahan atau *update* berhasil. Hal ini dibuktikan dimana pada awalnya *ticket title* adalah ”bug” dan berhasil berubah menjadi ”buggy”. Serta terdapat penambahan pada kolom *respond date and time* dan kolom *close date and time* yang pada mulanya kosong seperti yang dapat dilihat pada gambar 3.18. Dalam proses

perubahan atau *update* sistem dibantu dengan Axios yang memungkinkan sistem dapat mengirimkan data pada *database*.

C. Pembuatan Halaman *Ticket Dashboard*

Halaman *ticket dashboard* dapat diakses dengan cara membuka *side bar* yang terdapat pada *landing page*. Terdapat menu *projects* pada side bar dan kemudian sistem akan mengarahkan ke halaman daftar *project*. Langkah terakhir *ticket dashboard* dapat diakses ketika pengguna telah memilih projek yang diinginkan.

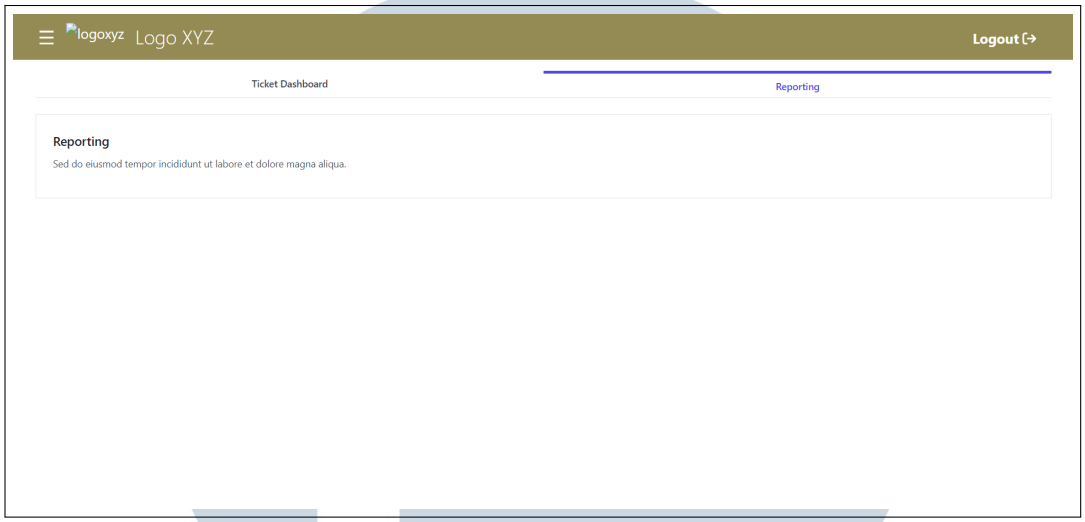
Gambar 3.24 menunjukkan tampilan awal situs web sesaat setelah pengguna memilih projek yang tertera,



Gambar 3.24. Tampilan Halaman Ticket Dashboard

Seperti yang dapat dilihat pada gambar 3.24. pada halaman *ticket dashboard* terdapat 2 tab yaitu *Ticket dashboard* dan *reporting*. Dimana sistem secara otomatis akan membuka tab *ticket dashboard* setelah pengguna memilih projek. Pengguna

dapat mengganti tab dengan menekan *reporting* dan tampilan pada halaman akan berubah seperti yang terdapat pada gambar 3.25.



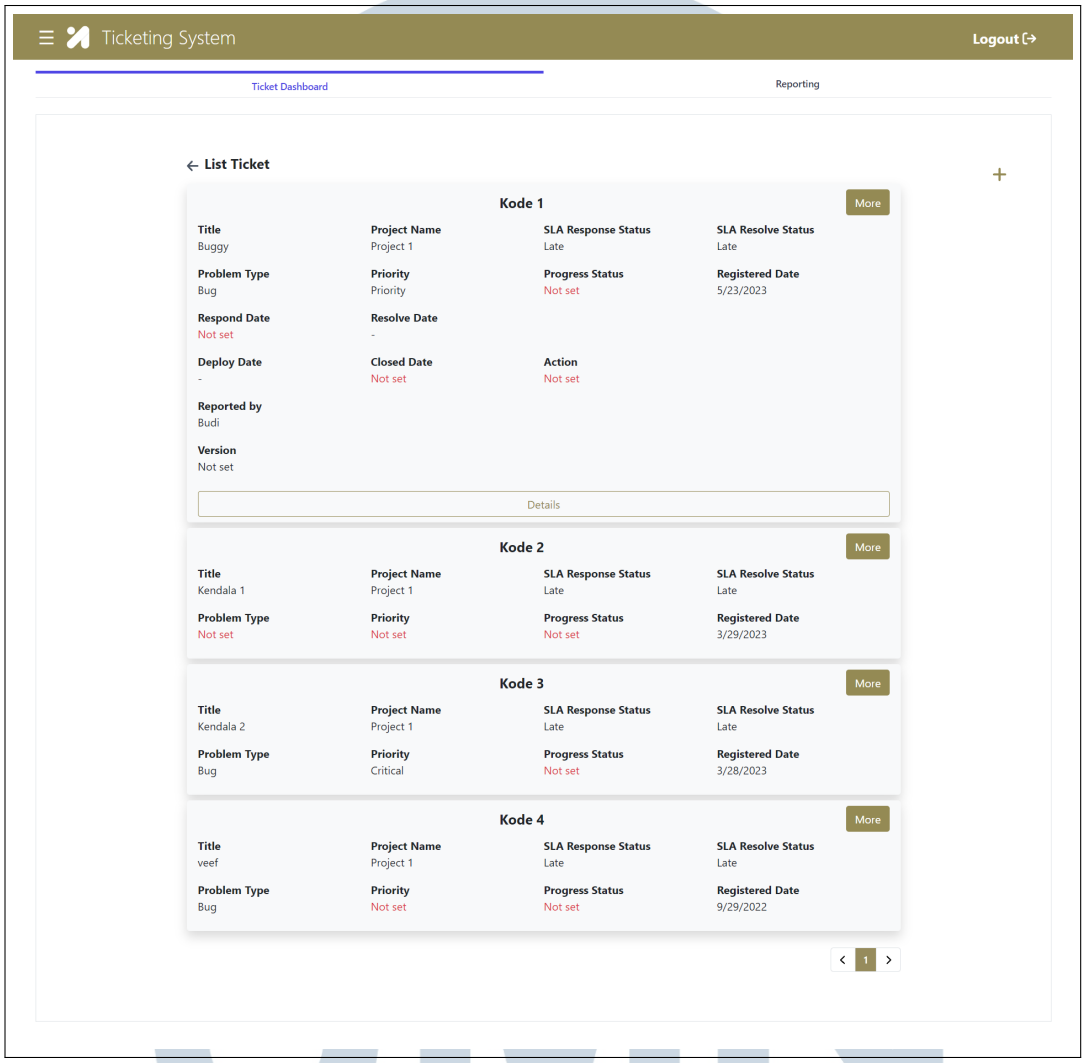
Gambar 3.25. Tampilan Tab Reporting

Informasi daftar tiket yang diberikan pada halaman tab *ticket dashboard* didapat dari pemanggilan API. Saat halaman *ticket dashboard* terbuka, sistem otomatis mengambil id dari projek melalui path URL menggunakan bantuan *react hook* yaitu useRouter. Dengan didapatkannya id projek maka sistem akan otomatis melakukan *request* menggunakan id dari projek kepada *get ticket list by id* API. Hasil dari pemanggilan API juga disertakan untuk sistem melakukan *pagination* sebanyak 10 tiket perhalamannya. Hal ini dibuktikan pada gambar 3.24. dimana karena jumlah tiket pada project 1 hanya ada 4 sehingga sistem hanya menampilkan angka 1 atau 1 halaman pada bagian kanan bawah situs web.

Fitur yang ada pada kolom tiket tidak terlalu berbeda dengan yang ada pada *landing page* atau halaman awal. Dimana terdapat tombol *more* pada setiap daftar tiket yang tertera pada halaman. Tombol *more* akan membuat tampilan kolom dari tiket yang dipilih menjadi lebih besar dan menampilkan informasi yang lebih detail juga. Serta sama seperti yang ada pada *landing page* terdapat tombol *details* pada bagian bawah kolom setelah menekan tombol *more*.

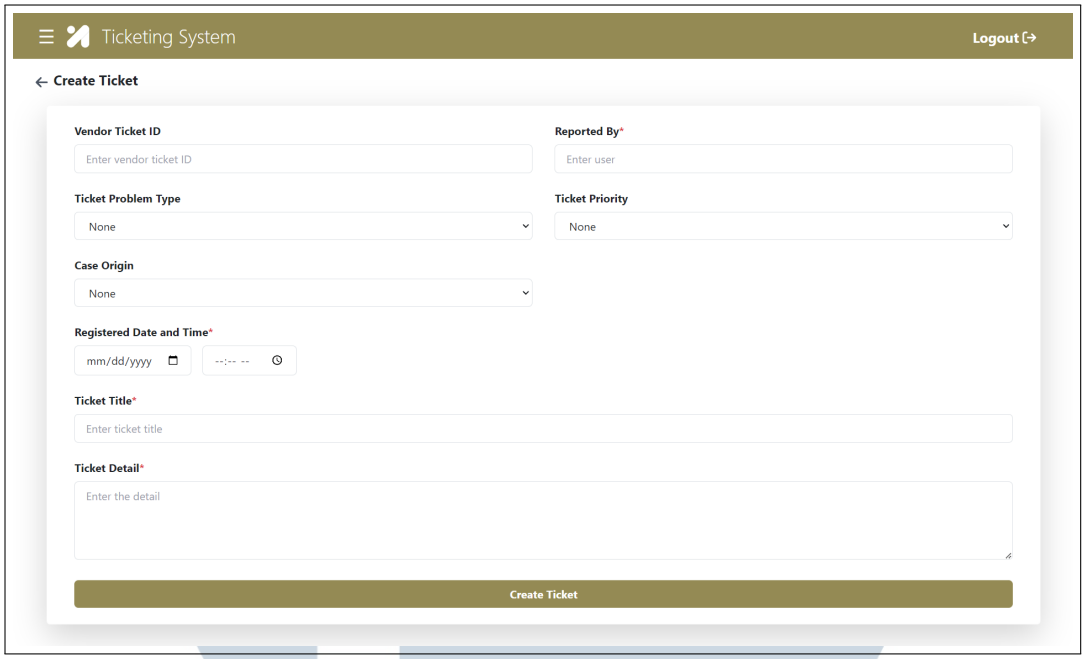
Perbedaan dari halaman *ticket dashboard* dengan *landing page* terdapat pada tampilan daftar tiket, teknik pemanggilan API, dan terdapat perbedaan fitur di dalamnya. Dimana pada tampilan pada daftar tiket pada *landing page* tidak terdapat batasan *project* dimana *landing page* menampilkan 3 tiket yang berasal dari berbagai projek sedangkan pada *ticket dashboard* tiket yang ditampilkan hanya

dalam 1 projek. Dalam segi pemanggilan API untuk *landing page* tidak disertakan *request* id seperti yang terjadi pada *ticket dashboard*.



Gambar 3.26. Tampilan Tab Ticket Dashboard Setelah Tombol More Ditekan

Dapat dilihat pada gambar 3.26. terdapat tombol details yang dimana bila ditekan akan mengarahkan pengguna ke halaman *ticket detail* yang sudah dijelaskan sebelumnya. Terdapat simbol (+) yang dapat dilihat pada gambar 3.26. pada bagian kanan atas. Simbol (+) atau plus akan mengarahkan pengguna ke halaman untuk menambahkan tiket pada projek yang telah dipilih. Perpindahan halaman dapat dilihat pada gambar 3.27.



Gambar 3.27. Tampilan Halaman Add New Ticket

3.4 Kendala dan Solusi yang Ditemukan

Pada sesi ini penulis akan menjabarkan permasalahan - permasalahan yang dihadapi serta solusi yang digunakan untuk memecahkan permasalahan tersebut.

3.4.1 Kendala yang Ditemukan

Dalam melaksanakan pengerjaan projek magang yang diberikan, terdapat beberapa kendala yang dihadapi atau ditemui. Kendala yang dihadapi terdiri dari kendala cukup besar dan terdapat kendala yang ringan. Berikut penulis sertakan kendala - kendala yang dihadapi saat melaksanakan pekerjaan projek magang.

1. Terdapat perbedaan cara dalam melakukan *setup* dan *installing* beberapa hal yang diperlukan dalam *project*.
2. Terdapat perbedaan cara penulisan dan struktur folder di file projek dengan yang diajarkan pada saat *training*.

3.4.2 Solusi dari Kendala yang Ditemukan

Dari kendala - kendala yang dihadapi terdapat beberapa solusi yang ditemukan untuk membantu dalam memecahkan permasalahan yang sedang

dihadapi, antara lain.

1. Melakukan konsultasi terkait *setup* projek kepada senior *developer* yang sudah terlibat dalam projek terlebih dahulu. (Menanyakan bila terdapat *error* pada saat melakukan *setup*)
2. Melakukan pencarian tentang *package* yang baru dipakai dan memahami cara penulisan yang ada pada *repository* projek yang sudah ada sebelumnya.