BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama pelaksanaan kerja magang di PT Mitra Integrasi Digital, mahasiswa magang ditempatkan sebagai *junior Programmer* yang berfokus pada pengembangan *website* pemesanan tiket. Sebagai *junior programmer*, mahasiswa magang bertanggung jawab untuk mengembangkan komponen antarmuka pengguna dan mengimplementasikan fitur-fitur *frontend* sesuai dengan desain yang telah ditentukan, serta turut berkontribusi dalam pengembangan API *backend* menggunakan teknologi Golang.

Selama magang, mahasiswa magang mengikuti alur kerja yang bersifat iteratif dan berbasis *task* harian. Setiap hari, mahasiswa magang menerima tugas dari mentor atau *supervisor* yang kemudian dikerjakan secara mandiri. Jika tugas belum selesai pada hari tersebut, maka akan dilanjutkan ke hari berikutnya. Setelah tugas selesai, hasil kerja akan ditunjukkan kepada *supervisor* untuk mendapatkan masukan dan validasi melalui sesi *review* di sore atau malam hari.

Selain itu, *supervisor* juga secara berkala melakukan pengecekan terhadap hasil kerja mahasiswa magang, termasuk melakukan pengujian dan *code review* apabila diperlukan. Proses ini memberikan umpan balik langsung yang membantu mahasiswa magang memahami standar kualitas kode dan praktik pengembangan perangkat lunak yang baik. Pola kerja ini mendorong pembelajaran yang efektif dan membiasakan mahasiswa magang dengan siklus iteratif dalam pengembangan perangkat lunak.

3.2 Tugas yang Dilakukan

Mahasiswa magang akan menjelaskan Tugas yang dilakukan ketika magang. Mulai dari Tugas kerja magang, penjelasan tugas, detail pelaksanaan tugas, *project requirement* dan rancangan awal, proses *development*, hingga halaman implementasi dan pengujian

3.2.1 Tugas Kerja Magang

Selama periode magang yang berlangsung dari Desember 2024 hingga April 2025, mahasiswa melaksanakan berbagai tugas yang berkaitan dengan pengembangan *website ticketing* dan *project printing services*. Berikut adalah rincian tugas yang dilakukan setiap minggunya, sesuai dengan arahan dan supervisi dari Pak Boby Hartanto dan Pak Boby Harmoko:

Tabel 3.1. Tugas yang dilakukan selama magang

Minggu	**Tugas yang dilakukan**	**Tanggal Mulai**	**Tanggal Selesai**
1	Mempelajari teknologi yang dipakai oleh perusahaan	2 Desember 2024	6 Desember 2024
2	Merancang *database* dan *slicing navbar*, *footer* dan *homepage*	9 Desember 2024	13 Desember 2024
3-7	Membuat beberapa *page* dan API, seperti *login*, *register*, *change password*, *event*, dan lain lain	16 Desember 2024	16 Januari 2025
8	Mempelajari *payment gateway* (Xendit), mempelajari FRP untuk *testing webhook*, *deploy* ke *server* dan konfigurasi *nginx*	20 Januari 2025	24 Januari 2025
9-10	Membuat fungsionalitas baru: *featured event*, *search* pada *homepage*, mempelajari PWA dan mengimplementasikannya	28 Januari 2025	7 Februari 2025
11	Menyelesaikan revisi pada *homepage* dan mempelajari untuk membuat *backend*	11 Februari 2025	14 Februari 2025
12-13	Membuat *backend* untuk mengorganisir konten-konten pada *frontend*	17 Februari 2025	26 Februari 2025

Berlanjut ke halaman berikutnya

Tabel 3.1 –Tugas yang dilakukan selama magang (Lanjutan)

Minggu	Tugas yang dilakukan	Tanggal Mulai	Tanggal Selesai
13-16	Dipindahkan ke *project* lain yang *urgent*, mempelajari teknologi baru dan menambahkan beberapa fitur	27 Februari 2025	21 Maret 2025
17-22	Membuat fitur-fitur pada *project printing* sesuai *task* dari atasan, tetap mengerjakan *project ticketing* saat tidak ada *task* dari *project printing*	24 Maret 2025	30 April 2025

3.2.2 Penjelasan Tugas

A Project Website Ticketing

Pada *project website ticketing* ini terdapat 10 halaman *frontend* dan 8 halaman *backend*. Setiap halaman mempunyai fungsi dan fiturnya masing-masing. mahasiswa magang akan menjelaskan secara singkat melalui tabel di bawah ini.

Tabel 3.2. Halaman pada *frontend*

Halaman	Fungsi	Fitur
Home	*Page* dimana pertama kali *user* masuk, dapat melihat beberapa *event* yang *featured*, dan juga *organizer-organizer*	*User* dapat langsung melakukan *search* dengan parameter seperti nama *event*, *category* dan *location*
Event List	*User* dapat melihat semua *event* yang ada	*User* dapat melakukan *searching* dengan *multiple parameter* yaitu *event name*, *location*, *category* dan *date*
Detail Event	*Page* untuk melihat detail *event* dan membeli *ticket*	Menunjukkan detail *event* dan jenis *ticket*, *page* untuk pembelian *ticket*

Berlanjut ke halaman berikutnya

Tabel 3.2 Halaman pada *frontend*(Lanjutan)

Halaman	**Fungsi**	**Fitur**
Article List	*User* dapat melihat *list article* yang tersedia	Menyediakan *article* yang tersedia dan dibagi berdasarkan kategori
Detail Article	*User* dapat membaca *article*	Menampilkan detail *article* yang dipilih oleh *user*
Detail Organizer	*User* dapat melihat informasi dari *organizer*	Menampilkan informasi dari *organizer*, termasuk *event-event* yang diadakan yang telah dibagi berdasarkan *event* yang telah lewat dan belum
Login	*User* dapat melakukan *login*	*Login* menggunakan *username* dan *password*
Register	*User* dapat melakukan registrasi akun	Registrasi akun dengan menyertakan informasi seperti *username*, *password*, *email* dan tanggal lahir
Account Detail	*User* dapat melihat detail akunnya dan merubahnya jika diperlukan	Melihat detail akun seperti *email*, nama lengkap, nomor telefon, *gender* dan juga foto profil, dan dapat diubah
Change Password	*User* dapat melakukan perubahan *password*	Merubah *password* dengan menyertakan *password* terkini, dan *password* baru

Selain mengembangkan halaman antarmuka pengguna (front end), mahasiswa magang juga bertanggung jawab dalam merancang dan membangun halaman untuk sisi backend. Halaman backend ini ditujukan khusus bagi pihak admin sebagai pusat kendali untuk mengelola berbagai data dan konten dalam sistem. Melalui halaman ini, admin dapat melakukan penyuntingan terhadap data seperti tiket, events, organizer, artikel, dan berbagai entitas lainnya yang berkaitan dengan kegiatan atau layanan yang disediakan. Tidak hanya itu, halaman backend juga dilengkapi dengan fitur manajemen pengguna, di mana admin memiliki wewenang untuk melakukan blacklist terhadap pengguna pada sisi frontend apabila ditemukan pelanggaran kebijakan atau aktivitas yang mencurigakan. Dengan demikian, halaman backend berfungsi sebagai alat utama

untuk menjamin kelancaran operasional dan keamanan sistem secara keseluruhan.

Tabel 3.3. Halaman pada *backend*

Halaman	**Fungsi**	**Fitur**
Login	Agar *user* dapat masuk ke dalam *page* dan menggunakan menu-menu sesuai dengan hak aksesnya	*User* dapat melakukan *login* dengan *username* dan *password*, sesuai dengan *role* yang telah diberikan oleh *super admin*, dan juga *user* hanya dapat melihat menu sesuai dengan hak aksesnya
Master User List	*User* dapat membuat, melihat, mengubah, dan menghapus *user-user* yang telah ada dan detailnya	*User* dapat melihat *user-user* dan informasinya (*name*, *username*, *role*), *user* juga dapat mengubah *password* dengan memasukkan *password* sekarang dan *password* baru. *User* juga dapat melakukan CRUD (*Create*, *Read*, *Update* dan *Delete*)
Master Organizer List	*User* dapat membuat, melihat, mengubah, dan menghapus *organizer* yang telah ada	Menunjukkan *organizer* yang ada dan informasi-informasinya, fitur CRUD untuk mengubah data dari *organizer* dan *searching organizer*
Master Article List	*User* dapat membuat, melihat, mengubah, dan menghapus *article* yang telah ada dan detailnya	Menunjukkan *article* yang ada dan informasi detailnya, fitur CRUD untuk mengubah data dari *article* dan *searching article*
Master User Group	*User* dapat membuat, melihat, mengubah, dan menghapus hak akses suatu *user group* dan detailnya	*User* dapat CRUD *user group* untuk mengatur hak akses bagi sekelompok *user* yang berada di *user group* tertentu. Hak akses yang dapat diatur adalah hak akses terhadap suatu menu, hak aksesnya berupa *view*, *edit*, *delete*, dan *create*

Berlanjut ke halaman berikutnya

Tabel 3.3 Halaman pada *backend* (Lanjutan)

Halaman	Fungsi	Fitur
Master Article Category	*User* dapat membuat, melihat, mengubah, dan menghapus *article category* dan detail yang telah ada	Menunjukkan *article category* yang ada dan informasi detailnya, fitur CRUD untuk mengubah data dari *article category* dan *searching article category*
Master Event	*User* dapat membuat, melihat, mengubah, dan menghapus *event* dan detail yang telah ada	Menunjukkan *event-event* yang ada dan detailnya, fitur CRUD untuk mengubah data dari *event*, termasuk untuk CRUD *ticket-ticket* yang ada pada masing-masing *event* dan juga *searching event*
Master Customer	*User* dapat melihat nama *customer* dan status *blacklist*nya. *User* juga dapat mengubah status *blacklist*nya	*User* dapat melakukan *blacklist* ini agar *customer* tidak dapat *login* dengan alasan tertentu

Untuk hasil dari perancangan, beserta skema *database* berupa ERD, akan dilampirkan pada halaman selanjutnya pada bagian *project development* dan halaman implementasi

B Project Printing Services

Pada minggu ke-13 hingga ke-22, mahasiswa magang dipindahkan ke *project printing services* yang bersifat *urgent* dan memiliki prioritas tinggi. *Project* ini merupakan sistem yang memudahkan proses *file services* seperti *printing*, *scanning*, dan *copying*, dimana proses transaksinya dilakukan secara *online*. *Customer* hanya tinggal datang ke tempat dan memberikan kode unik yang di-*generate* berbeda untuk setiap transaksinya, dan dapat mengambil *file* tersebut. Teknologi yang digunakan dalam *project* ini adalah Node.js, Express.js, dan Handlebars untuk *template engine*. Namun mahasiswa magang hanya akan berfokus pada rancang bangun *web ticketing* karena *project printing* ini merupakan *project* yang bersifat *confidential*. Namun ketika *project* ini sedang senggang, mahasiswa magang tetap akan melanjutkan pengembangan pada *website ticketing*.

3.3 Detail Pelaksanaan Magang

Pada minggu pertama hingga minggu terakhir, mahasiswa magang diposisikan sebagai *junior developer*, dimana bertanggung jawab atas pengembangan *website* secara keseluruhan, mulai dari *frontend*, *backend*, dan juga API. Total jam kerja yang diselesaikan selama magang mencapai lebih dari 640 jam, melebihi batas minimal yang ditetapkan oleh kampus.

3.3.1 Project Requirement dan Rancangan awal

Website ticketing ini dikembangkan untuk memfasilitasi proses pencarian dan pembelian tiket berbagai jenis acara secara *online*, mulai dari konser, seminar, *event* olahraga, hingga *event* hiburan lainnya. Sistem ini dirancang untuk mengotomasi dan memperluas jangkauan proses penjualan tiket yang sebelumnya dilakukan secara manual dan terbatas pada area tertentu. Dengan adanya sistem ini, proses pembelian tiket menjadi lebih praktis, cepat, dan dapat diakses oleh *target* pasar yang lebih luas.

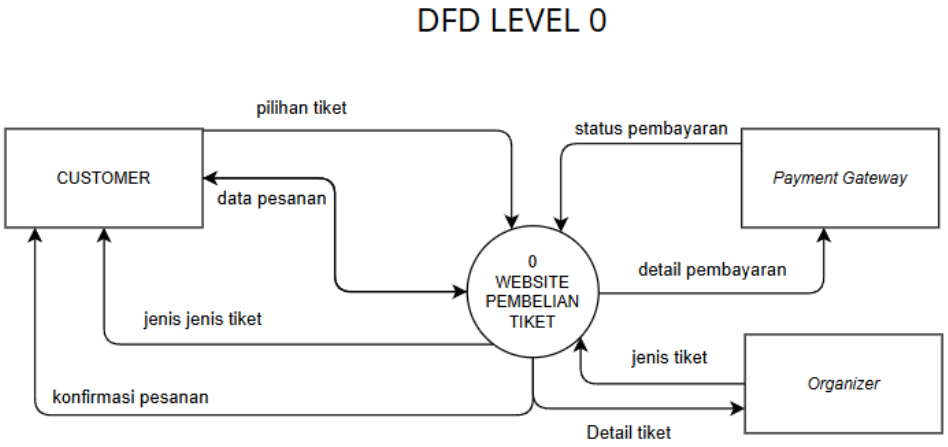
Website ini dibangun dengan arsitektur sistem yang terdiri dari tiga bagian utama, yaitu *Front End* untuk pelanggan (*customer*), *Back End* berupa API *service* menggunakan Golang, serta *Admin Service* untuk pengelolaan konten dan data oleh penyelenggara acara. Namun, dalam proyek ini fokus pengembangan difokuskan pada sisi *Front End* yang langsung berinteraksi dengan pengguna (*end-user*).

Fitur-fitur utama yang disediakan dalam *Front End* ini meliputi sistem *login* dan registrasi, pencarian tiket dengan berbagai *filter*, proses pembelian tiket terintegrasi dengan *payment gateway* Xendit, serta penyediaan artikel-artikel informatif yang berkaitan dengan *event* yang sedang atau akan berlangsung. Pengguna dapat langsung mengakses halaman utama *website* untuk mencari *event* atau membaca artikel tanpa harus *login*. Namun, untuk melakukan pembelian tiket atau *checkout*, pengguna diwajibkan untuk melakukan *login* terlebih dahulu.

Website ini juga memungkinkan pengguna untuk menerima *invoice* dan tiket secara digital melalui *email* setelah proses pembayaran selesai. Dengan integrasi ke Xendit sebagai *payment gateway*, sistem pembayaran telah berjalan dengan lancar. Di sisi lain, fitur *generate e-ticket* masih dalam tahap pengembangan lanjutan dan direncanakan akan segera diselesaikan untuk melengkapi proses pembelian dari awal hingga akhir.

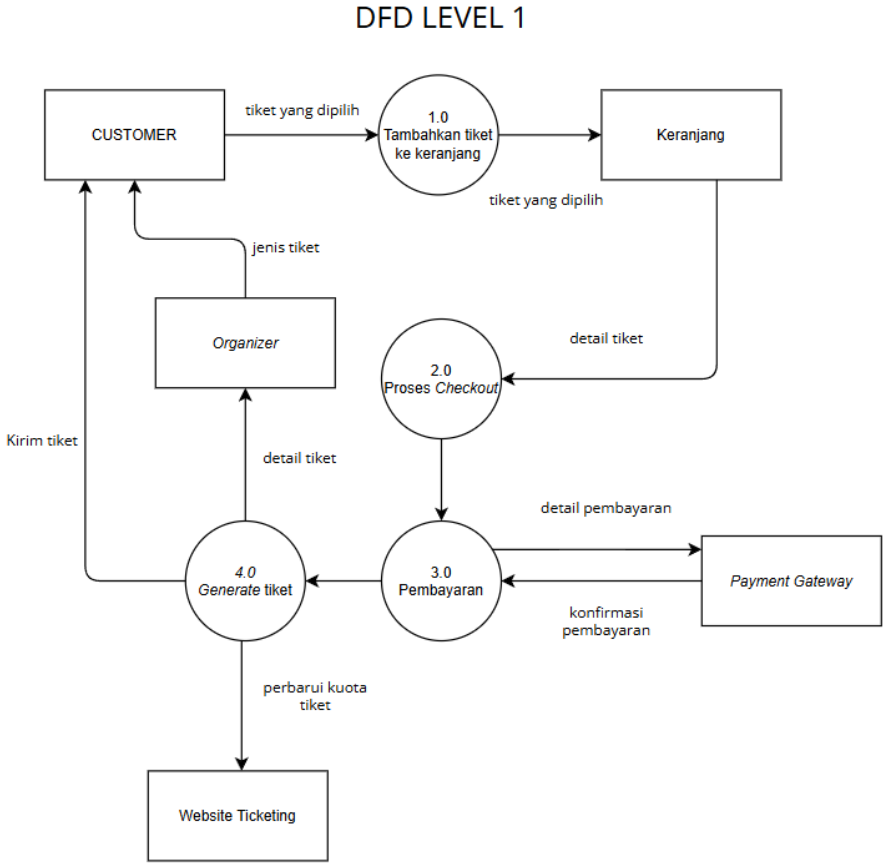
3.3.2 Data Diagram

Data Flow Diagram (DFD) adalah teknik grafis yang menggambarkan alur informasi dan transformasi data dalam sistem. DFD menunjukkan bagaimana data bergerak melalui sistem, dari input hingga output, serta proses-proses yang mengubah data tersebut. Diagram ini terdiri dari empat komponen utama: *external entity* (entitas luar), proses, *data store* (penyimpanan data), dan alur data. DFD digunakan dalam analisis dan desain sistem untuk memahami kebutuhan sistem secara visual dan memudahkan komunikasi antara analis sistem dengan pengguna.



Gambar 3.1. *Data Flow Diagram Level 0*

DFD Level 0 atau context diagram menggambarkan sistem website pembelian tiket sebagai satu proses utama bernomor "0" yang menunjukkan batas sistem serta interaksi dengan dua entitas eksternal, yaitu *customer* dan *payment gateway*. *Customer* mengirimkan data berupa pilihan tiket dan pesanan ke dalam sistem, kemudian sistem merespons dengan menampilkan daftar tiket yang tersedia serta mengirimkan konfirmasi pesanan. Di sisi lain, sistem juga berkomunikasi dengan *payment gateway* untuk memproses pembayaran, dengan mengirimkan detail transaksi dan menerima status pembayaran yang menunjukkan keberhasilan atau kegagalan. Seluruh proses ini mencerminkan peran sistem sebagai perantara yang mengintegrasikan aktivitas pemesanan dan penyelesaian pembayaran secara menyeluruh dan efisien.



Gambar 3.2. *Data Flow Diagram Level 1*

DFD Level 1 merupakan hasil dekomposisi dari proses ”0” pada *context diagram* yang dipecah menjadi empat proses utama dengan satu *data store* yang mendukung operasional sistem *e-commerce*. Setiap proses memiliki fungsi spesifik dan saling berinteraksi melalui aliran data untuk mencapai tujuan sistem secara keseluruhan.

A Proses-Proses Utama:

Proses 1.0 - Tambahkan *Event* ke keranjang berfungsi untuk menangani aktivitas penambahan produk atau *event* yang dipilih *customer* ke dalam keranjang belanja. Proses ini menerima *input* berupa tiket yang dipilih dari *customer*, kemudian memproses dan mengirimkan tiket pada keranjang ke entitas Keranjang. Proses ini merupakan tahap awal dari transaksi pembelian dalam sistem.

Proses 2.0 - Proses *Checkout* menangani tahap pemrosesan *checkout* dari

keranjang belanja. Proses ini menerima detail tiket dari entitas Keranjang dan memproses informasi tersebut untuk diteruskan ke tahap pembayaran. Proses ini berfungsi sebagai penghubung antara pemilihan produk dengan proses pembayaran.

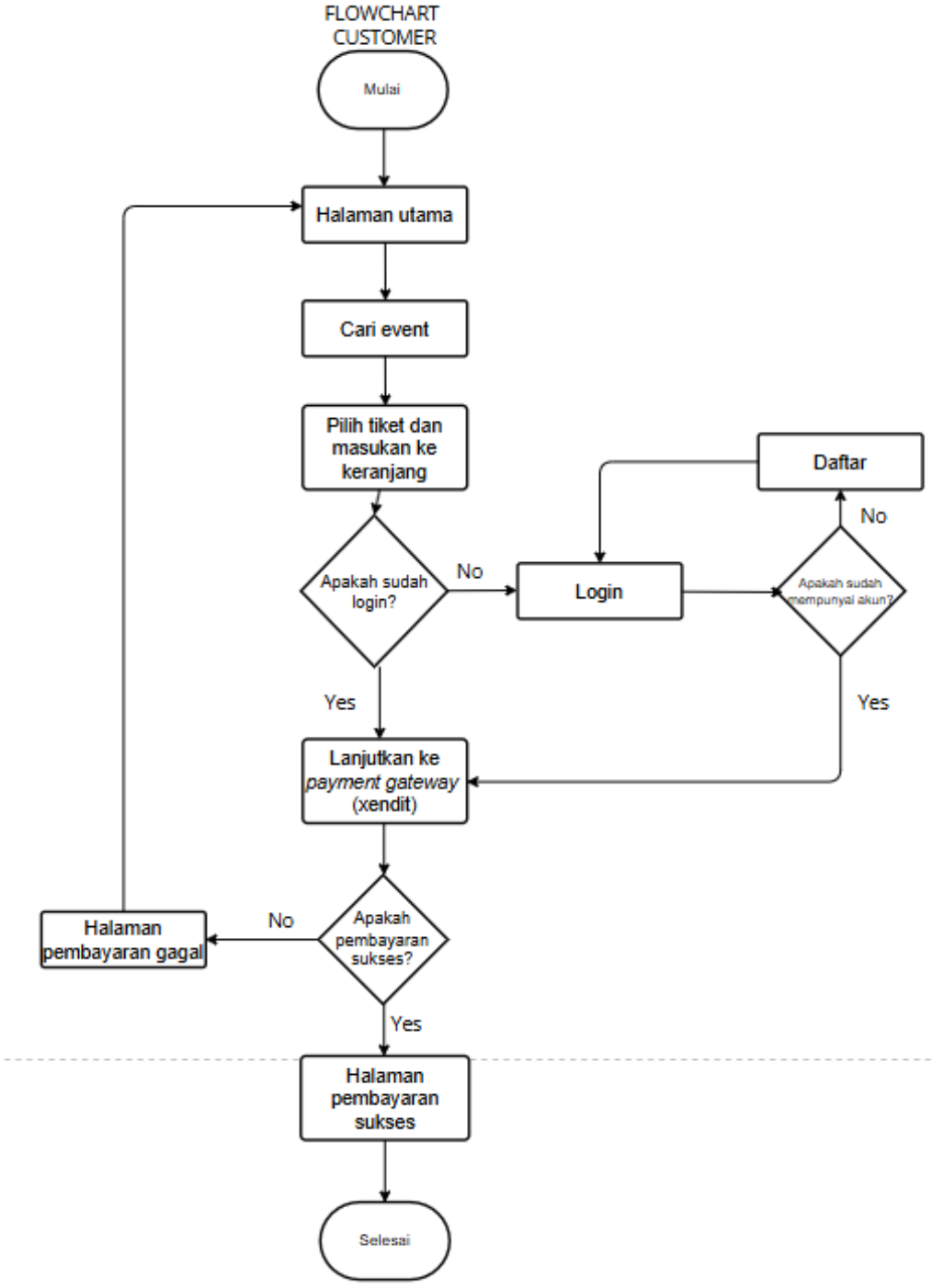
Proses 3.0 – Pembayaran merupakan tahapan penting yang menangani seluruh aktivitas transaksi pembayaran dengan melibatkan integrasi langsung dengan sistem *payment gateway*. Proses ini menerima data dari proses 2.0 berupa detail pembelian dan informasi pelanggan, lalu mengirimkan detail pembayaran ke *payment gateway* untuk diproses secara aman. Setelah menerima status konfirmasi pembayaran (berhasil atau gagal), sistem akan menindaklanjuti status tersebut. Jika pembayaran berhasil, proses ini tidak hanya mengirimkan konfirmasi ke proses berikutnya untuk menghasilkan tiket, tetapi juga mengirimkan informasi detail tiket yang telah dibeli kepada pihak event organizer sebagai notifikasi penyelenggaraan. Selain itu, proses ini bertanggung jawab untuk memperbarui kuota tiket secara real-time di dalam sistem website ticketing, agar ketersediaan tiket tetap akurat dan up to date. Dengan demikian, proses 3.0 berperan sebagai penghubung krusial antara sistem pembayaran, penyelenggara acara, dan manajemen inventori tiket dalam sistem.

Proses 4.0 - *Generate* Tiket berfungsi untuk menghasilkan tiket sebagai bukti transaksi yang akan diberikan kepada *customer*. Proses ini menerima konfirmasi pembayaran dari proses 3.0 dan menghasilkan tiket yang dikirimkan kepada *customer* sebagai *output* akhir dari sistem. Tiket ini berfungsi sebagai bukti pembelian yang sah.

3.3.3 Flowchart Diagram

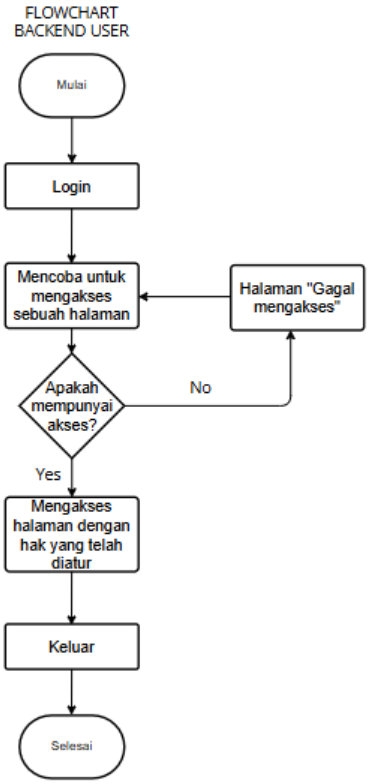
Flowchart atau bagan alur adalah suatu diagram yang menampilkan langkah-langkah dan keputusan yang diperlukan untuk melakukan sebuah proses dari suatu program secara sistematis dan terstruktur. Diagram ini menggunakan simbol-simbol standar yang telah ditetapkan untuk merepresentasikan berbagai jenis operasi, keputusan, dan alur data dalam sistem.

Berikut adalah *flowchart diagram* dari proyek *website* Ticket yang menggambarkan alur kerja sistem dari awal hingga akhir proses.



Gambar 3.3. Flowchart Front end

Setelah memahami bagan alur kerja pada *front end* pada Gambar 3.12 , berikut adalah bagan alur yang menggambarkan proses *backend* sistem yang digunakan untuk mengelola konten



Gambar 3.4. Flowchart Backend

Dalam *website ticketing* ini, terdapat dua alur utama berdasarkan peran pengguna, yaitu ***Customer*** dan ***Backend User***. Untuk ***Customer***, alur dimulai dari halaman *homepage*, di mana pengguna dapat mencari *event*, memilih tiket, dan menambahkannya ke keranjang. Jika pengguna belum *login*, sistem akan mengarahkan ke halaman *login* terlebih dahulu. Jika belum memiliki akun, pengguna dapat melakukan registrasi terlebih dahulu sebelum melanjutkan *login*. Setelah *login* berhasil, pengguna akan diarahkan ke halaman pembayaran melalui *payment gateway* (Xendit). Jika pembayaran berhasil, maka akan ditampilkan halaman sukses; sebaliknya, jika gagal, pengguna akan diarahkan ke halaman gagal pembayaran.

Sementara itu, ***Backend User*** memulai alur dengan *login* ke sistem. Setelah *login*, pengguna akan mencoba mengakses halaman tertentu. Jika memiliki akses, pengguna dapat mengakses halaman tersebut dengan kontrol akses yang sesuai. Jika tidak memiliki akses, sistem akan membatasi akses ke halaman tersebut. Setelah selesai, pengguna dapat melakukan *logout* untuk mengakhiri sesi.

3.3.4 Proses Development

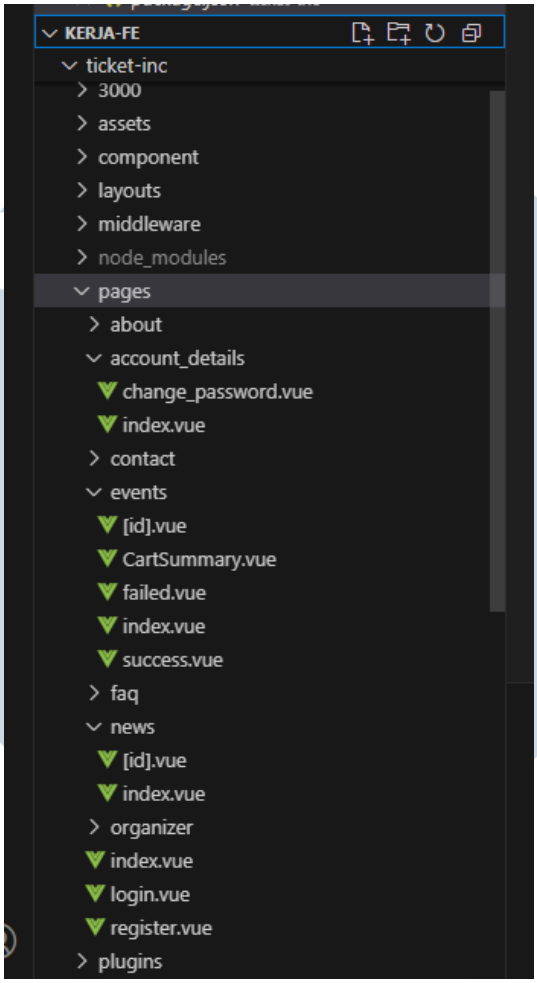
Proses pengembangan awal dari proyek *website ticketing* ini dilaksanakan selama kurang lebih 6 minggu, dimulai dari minggu ke-2 hingga minggu ke-8 masa magang. Fokus utama pada tahap awal ini adalah membangun halaman-halaman antarmuka pengguna (*front end*) serta mengembangkan API menggunakan Golang sebagai *backend service* untuk mendukung proses-proses pada sisi *Front End*. Selama proses tersebut, pengembangan dilakukan secara lokal menggunakan *localhost* sebagai media *preview* dan pengujian fitur-fitur dasar.

Tahap awal pengembangan meliputi pembuatan berbagai halaman penting seperti *login*, *register*, *event list*, dan *form* pembelian, serta pengembangan *endpoint* API yang mendukung fungsi *login*, *register*, *change password*, dan pengelolaan data *event*. Integrasi sistem pembayaran dengan Xendit sebagai *payment gateway* berhasil diselesaikan di minggu ke-8, yang sekaligus menjadi titik awal proses *deployment* ke *server* publik menggunakan konfigurasi Nginx sebagai *reverse proxy* dan PM2 untuk manajemen proses Node.js/Golang.

Setelah fitur-fitur dasar selesai dan dapat berjalan secara stabil, dilakukan proses *website development* dan *testing* secara internal, di mana setiap *task* yang telah dikerjakan harus didemonstrasikan kepada atasan dan di-*push* ke *repository* Git untuk dilakukan pengecekan. Pengujian dilakukan secara rutin oleh atasan langsung, yang berperan sebagai penguji dan *quality control* selama proses magang. Proses ini memungkinkan *feedback* langsung diberikan secara cepat dan dilakukan perbaikan secara iteratif.

Meskipun sistem ini masih dalam tahap pengembangan dan belum memasuki fase produksi secara penuh, terdapat kemungkinan besar bahwa sistem ini akan dilanjutkan dan digunakan sebagai solusi nyata untuk penjualan tiket oleh perusahaan. Saat ini, belum dilakukan tahapan lanjut seperti *security testing*, *penetration testing*, atau *sanity testing*, namun tahapan tersebut kemungkinan akan menjadi bagian dari proses validasi sebelum sistem diputuskan untuk *go-public*.

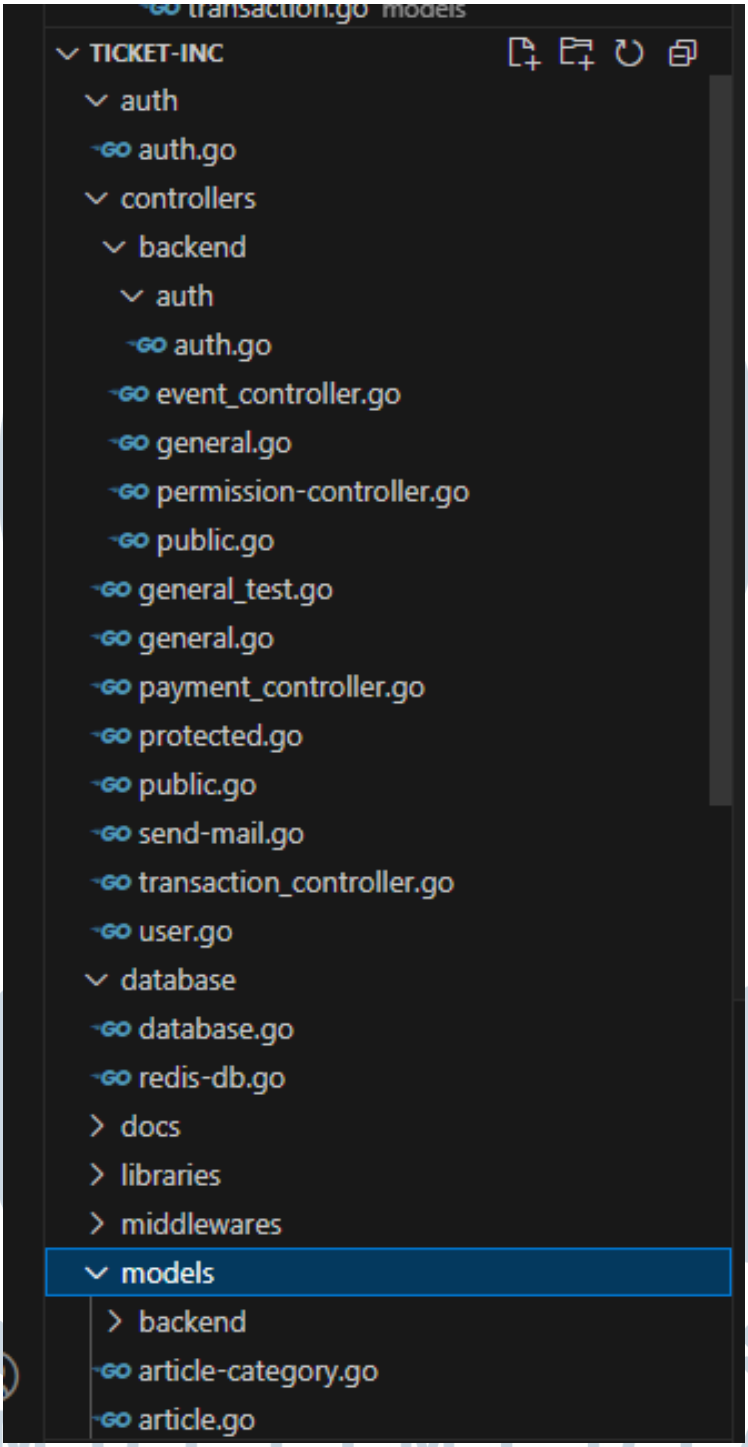
Hingga saat laporan kerja magang ini dikerjakan, tahap *development* proyek *Ticketing* ini masih berlangsung. Sejauh ini, hasil dari proyek *Ticketing* memiliki 18 halaman yang dapat diakses, dengan detail 10 halaman pada *frontend* dan 8 halaman pada *backend*. Namun akses dari setiap halaman pada *backend* berbeda sesuai dengan *role* dan hak akses yang diberikan kepada pengguna itu sendiri. Berikut adalah *asset-asset* yang digunakan dalam *website Ticketing* ini



Gambar 3.5. *Project data asset* untuk *front end*

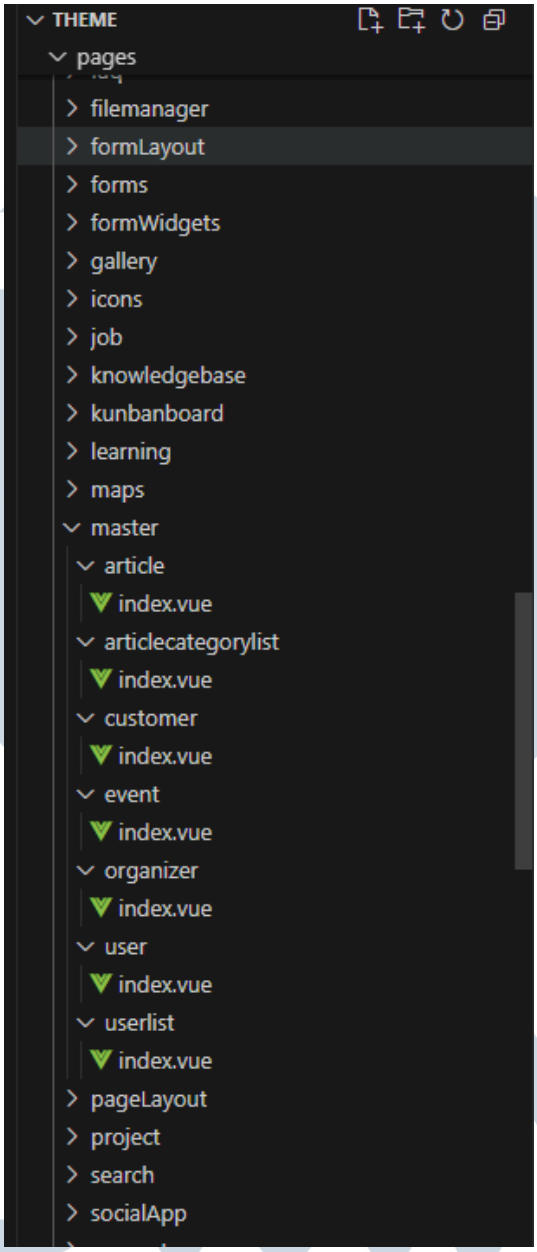
Gambar di atas menampilkan berbagai aset yang digunakan dalam pembangunan *website ticketing*, khususnya pada bagian *front-end*. Aset-aset ini mencakup elemen visual seperti ikon, serta komponen-komponen antarmuka pengguna (UI) yang mendukung tampilan dan interaksi dalam *website*. Penggunaan aset yang konsisten dan terstruktur berperan penting dalam menciptakan pengalaman pengguna yang intuitif, menarik, dan responsif.

Setelah membahas aset *front-end*, selanjutnya adalah penjelasan mengenai aset yang digunakan dalam pembangunan API untuk *website ticketing* ini. Aset-aset tersebut mencakup struktur *endpoint*, dokumentasi API, konfigurasi *middleware*, hingga dependensi dan *library* yang digunakan dalam proses pengembangan *back-end*. API ini berperan sebagai jembatan komunikasi antara *front-end* dan *server*, sehingga memungkinkan pertukaran data yang aman, cepat, dan efisien antar komponen sistem.



Gambar 3.6. *Project data asset* untuk API

Gambar 3.2 merupakan *asset-asset* yang digunakan untuk membangun API untuk *website ticketing* ini. Selanjutnya adalah *asset-asset* yang digunakan untuk membangun *website ticketing*, terutama pada bagian *backend*



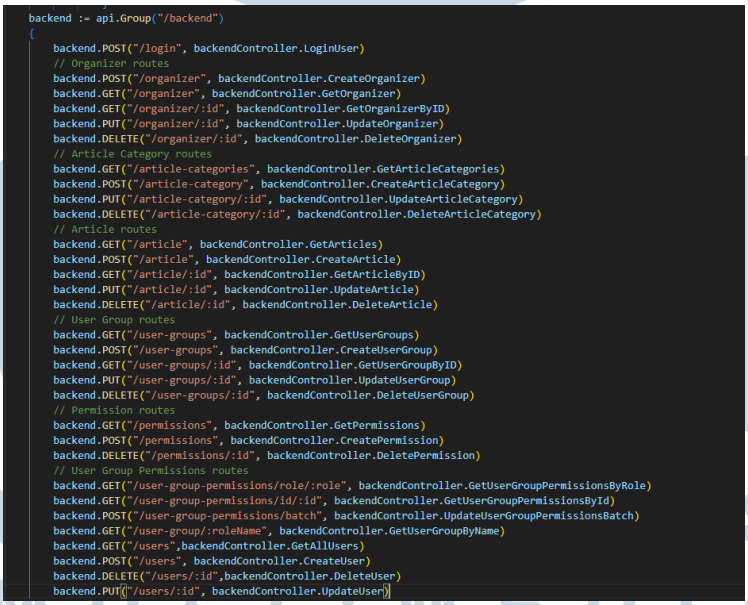
Gambar 3.7. *Project data asset* untuk *backend*

Gambar 3.3 merupakan *asset-asset* yang digunakan untuk membangun *website ticketing* pada sisi *backend*. Selain itu, terdapat total 76 *endpoint* dari API Golang untuk menunjang berjalannya *website* ini, dengan dengan 22 *public endpoint*, 46 *endpoint* untuk *backend* dan ada 7 *endpoint* untuk *protected endpoint*. Berikut adalah gambar dari *list endpoint* yang digunakan dalam *project ticketing* ini

```
// Create a new group for the API
api := r.Group("/api")
{
    // Create a new group for the public routes
    public := api.Group("/public")
    {
        public.GET("/ping", controllers.PingPong)
        public.POST("/login", controllers.Login)
        public.POST("/signup", controllers.Signup)
        public.GET("/locations", controllers.GetLocation)
        public.GET("/banners", controllers.GetBanners)
        public.GET("/categories", controllers.GetCategory)
        public.GET("/organizers", controllers.GetOrganizer)
        public.GET("/events", controllers.GetEvents)
        public.GET("/event/:eventUrl", controllers.GetEventInfo)
        public.GET("/organizers/:id", controllers.GetOrganizerDetail)
        public.GET("/article" , controllers.GetArticles)
        public.GET("/article/:id" , controllers.GetArticleDetail)
        public.GET("/article/:id/related" , controllers.GetRelatedArticles)
        public.POST("/payment/create-invoice", controllers.CreateXenditInvoice)
        public.POST("/payment/webhook", controllers.HandleXenditWebhook)
        public.POST("/transactions", controllers.CreateTransaction)
        public.GET("/transactions/:id", controllers.GetTransaction)
        public.POST("/transactions/:id/update-payment", controllers.UpdateTransactionPayment)
        public.GET("/customer-transactions", controllers.GetCustomerTransactions)
        public.POST("/tickets", controllers.CreateTickets)
        public.GET("/tickets/customer/:customer_id", controllers.GetTicketsByCustomer)
        public.GET("/tickets/event/:event_id", controllers.GetTicketsByEvent)
            }
```

Gambar 3.8. *List API Endpoint Public*

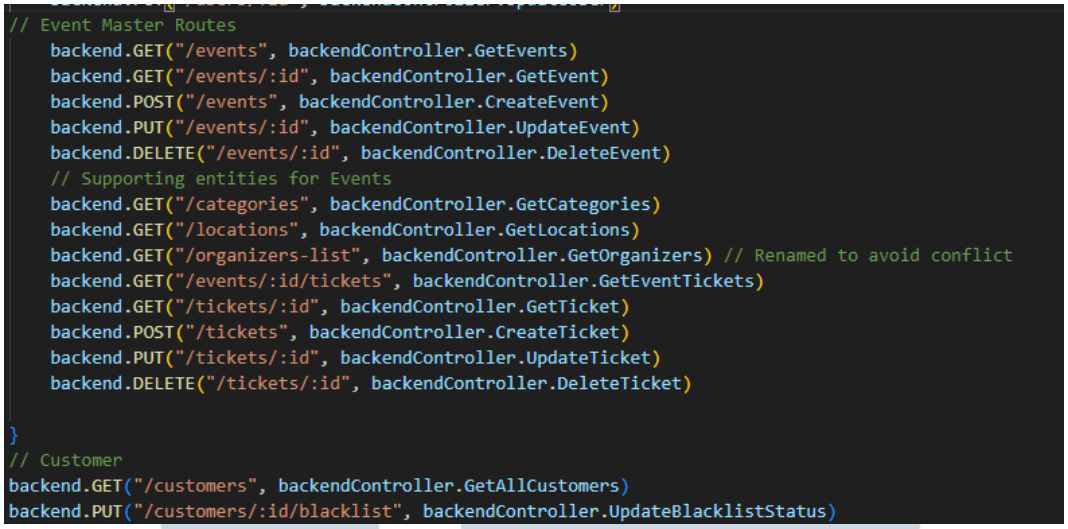
Gambar 3.4 merupakan *list public endpoint*. *Endpoint-endpoint* tersebut sebagian besar menangani permintaan-permintaan yang datang dari sisi *front end*. Berikutnya adalah *list api endpoint* pada *backend*



```
backend := api.Group("/backend")
{
    backend.POST("/login", backendController.LoginUser)
    // Organizer routes
    backend.POST("/organizer", backendController.CreateOrganizer)
    backend.GET("/organizer", backendController.GetOrganizer)
    backend.GET("/organizer/:id", backendController.GetOrganizerByID)
    backend.PUT("/organizer/:id", backendController.UpdateOrganizer)
    backend.DELETE("/organizer/:id", backendController.DeleteOrganizer)
    // Article Category routes
    backend.GET("/article-categories", backendController.GetArticleCategories)
    backend.POST("/article-category", backendController.CreateArticleCategory)
    backend.PUT("/article-category/:id", backendController.UpdateArticleCategory)
    backend.DELETE("/article-category/:id", backendController.DeleteArticleCategory)
    // Article routes
    backend.GET("/article", backendController.GetArticles)
    backend.POST("/article", backendController.CreateArticle)
    backend.GET("/article/:id", backendController.GetArticleByID)
    backend.PUT("/article/:id", backendController.UpdateArticle)
    backend.DELETE("/article/:id", backendController.DeleteArticle)
    // User Group routes
    backend.GET("/user-groups", backendController.GetUserGroups)
    backend.POST("/user-groups", backendController.CreateUserGroup)
    backend.GET("/user-groups/:id", backendController.GetUserGroupByID)
    backend.PUT("/user-groups/:id", backendController.UpdateUserGroup)
    backend.DELETE("/user-groups/:id", backendController.DeleteUserGroup)
    // Permission routes
    backend.GET("/permissions", backendController.GetPermissions)
    backend.POST("/permissions", backendController.CreatePermission)
    backend.DELETE("/permissions/:id", backendController.DeletePermission)
    // User Group Permissions routes
    backend.GET("/user-group-permissions/role/:role", backendController.GetUserGroupPermissionsByRole)
    backend.GET("/user-group-permissions/id/:id", backendController.GetUserGroupPermissionsById)
    backend.POST("/user-group-permissions/batch", backendController.UpdateUserGroupPermissionsBatch)
    backend.GET("/user-group/:roleName", backendController.GetUserGroupByName)
    backend.GET("/users",backendController.GetAllUsers)
    backend.POST("/users", backendController.CreateUser)
    backend.DELETE("/users/:id",backendController.DeleteUser)
    backend.PUT("/users/:id", backendController.UpdateUser)
```

Gambar 3.9. *List API Endpoint Backend*

Gambar 3.5 merupakan *list* dari API *endpoint backend*. *List* dari *endpoint* API ini digunakan untuk menerima permintaan yang datang dari sisi *backend*, seperti permintaan menambah, menyunting, menghapus ataupun mengakses data dari basis data.

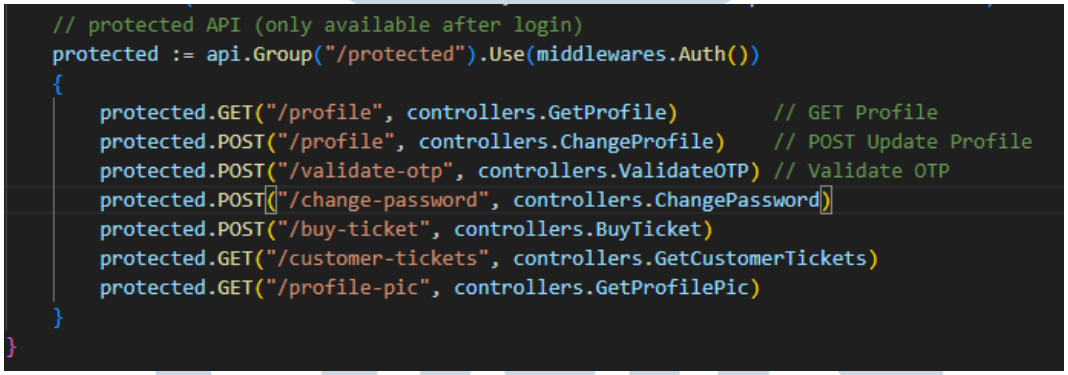


```
// Event Master Routes
    backend.GET("/events", backendController.GetEvents)
    backend.GET("/events/:id", backendController.GetEvent)
    backend.POST("/events", backendController.CreateEvent)
    backend.PUT("/events/:id", backendController.UpdateEvent)
    backend.DELETE("/events/:id", backendController.DeleteEvent)
    // Supporting entities for Events
    backend.GET("/categories", backendController.GetCategories)
    backend.GET("/locations", backendController.GetLocations)
    backend.GET("/organizers-list", backendController.GetOrganizers) // Renamed to avoid conflict
    backend.GET("/events/:id/tickets", backendController.GetEventTickets)
    backend.GET("/tickets/:id", backendController.GetTicket)
    backend.POST("/tickets", backendController.CreateTicket)
    backend.PUT("/tickets/:id", backendController.UpdateTicket)
    backend.DELETE("/tickets/:id", backendController.DeleteTicket)

}
// Customer
backend.GET("/customers", backendController.GetAllCustomers)
backend.PUT("/customers/:id/blacklist", backendController.UpdateBlacklistStatus)
```

Gambar 3.10. Lanjutan *List API Endpoint Backend*

Gambar 3.6 merupakan lanjutan dari *list* API *endpoint* pada *backend*. Terdapat beberapa *endpoint* API yang menangani perubahan data pada *events*, *organizers*, dan juga *ticket*.



```
    // protected API (only available after login)
    protected := api.Group("/protected").Use(middlewares.Auth())
    {
        protected.GET("/profile", controllers.GetProfile)        // GET Profile
        protected.POST("/profile", controllers.ChangeProfile)    // POST Update Profile
        protected.POST("/validate-otp", controllers.ValidateOTP) // Validate OTP
        protected.POST("/change-password", controllers.ChangePassword)
        protected.POST("/buy-ticket", controllers.BuyTicket)
        protected.GET("/customer-tickets", controllers.GetCustomerTickets)
        protected.GET("/profile-pic", controllers.GetProfilePic)
    }
}
```

Gambar 3.11. *List Api Endpoint Protected*

Gambar 3.7 merupakan *list endpoint* yang *protected*. Artinya, hanya dapat diakses ketika *user* sudah terautentikasi atau telah melakukan *login* sebelumnya.

Dalam pengerjaan proyek ini, digunakan berbagai *framework* dan *library* tambahan untuk menunjang pengembangan aplikasi baik pada sisi *frontend* maupun *backend*. Pada sisi *frontend*, digunakan Nuxt.js sebagai *framework* utama yang dibangun di atas Vue.js untuk mempermudah pengembangan aplikasi *web* modern dengan fitur seperti *server-side rendering* (SSR), *routing* otomatis, dan optimisasi performa. Axios digunakan sebagai *library* untuk melakukan komunikasi HTTP dengan *backend* API, seperti pengambilan data pengguna, autentikasi, dan

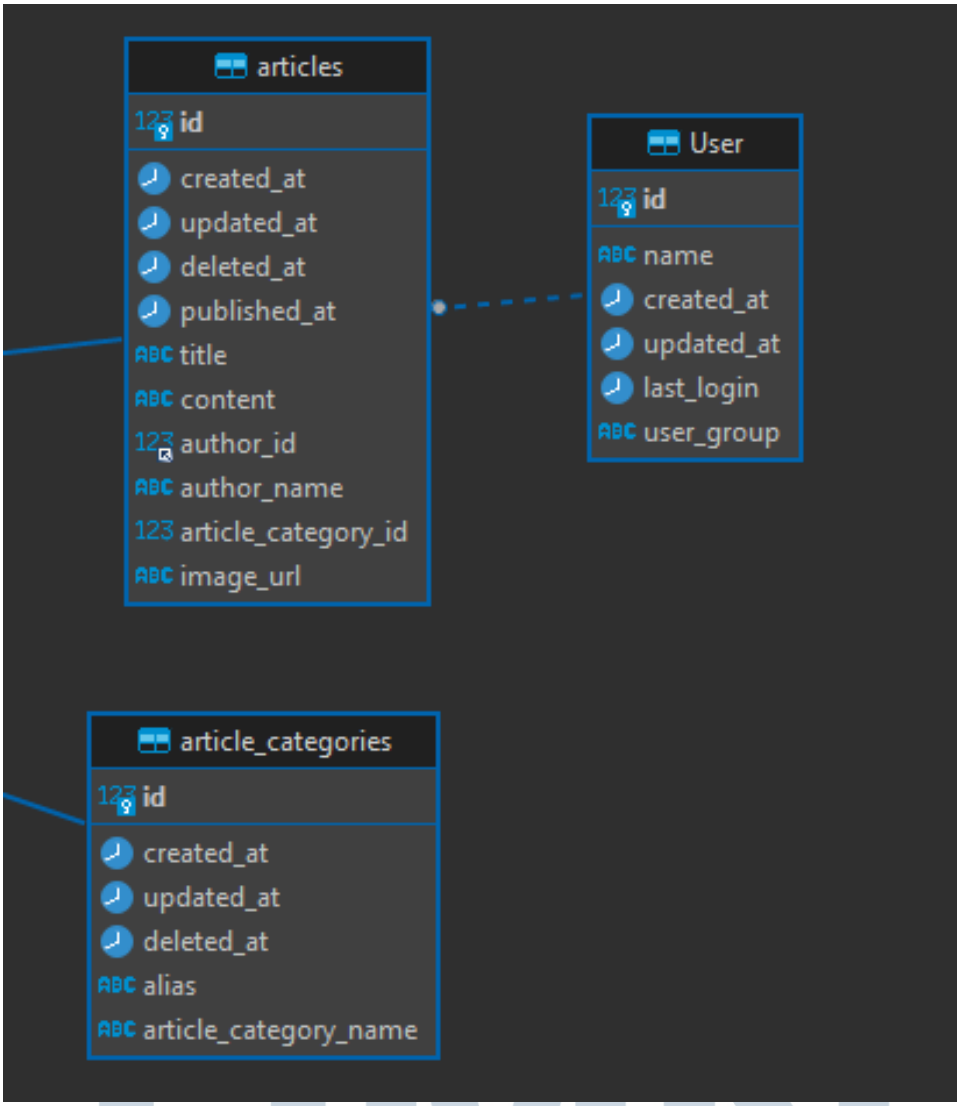
pemrosesan transaksi. Untuk pengelolaan *state*, digunakan Pinia sebagai *state management library* resmi dari Vue, yang diimplementasikan dalam berbagai komponen untuk menyimpan dan mengatur data aplikasi seperti *user session* dan data produk.

Selain itu, digunakan juga Moment.js untuk memformat dan memanipulasi tanggal/waktu, terutama pada komponen tampilan jadwal kegiatan atau transaksi. Berbagai *library* antarmuka pengguna lainnya juga digunakan, seperti @vueform/multiselect untuk *dropdown* yang interaktif, *swiper* untuk pembuatan *slider*, serta vue3-easy-data-table untuk menampilkan data dalam bentuk tabel responsif. Beberapa *plugin* tambahan seperti vue-toastification, vue-sweetalert2, dan vue3-toastify digunakan untuk menampilkan notifikasi dan umpan balik kepada pengguna guna meningkatkan pengalaman pengguna secara keseluruhan.

Pada sisi *backend*, proyek ini dibangun menggunakan bahasa pemrograman Go (Golang) dan memanfaatkan sejumlah *package* bawaan serta *library* eksternal. *Package* bawaan seperti net/http, time, errors, dan log digunakan untuk membangun *server*, menangani permintaan HTTP, memproses waktu, serta mencatat *log* sistem. Untuk mempermudah pembuatan API RESTful, digunakan *framework* Gin (github.com/gin-gonic/gin) yang dikenal ringan dan cepat. Autentikasi berbasis *token* ditangani menggunakan *library* jwt-go (github.com/dgrijalva/jwt-go) untuk proses *login* dan validasi akses. Selain itu, digunakan pula *package* lokal seperti *database* dan models/backend yang berfungsi untuk mengelola koneksi ke *database* serta mendefinisikan struktur data dan logika bisnis di sisi *backend*.

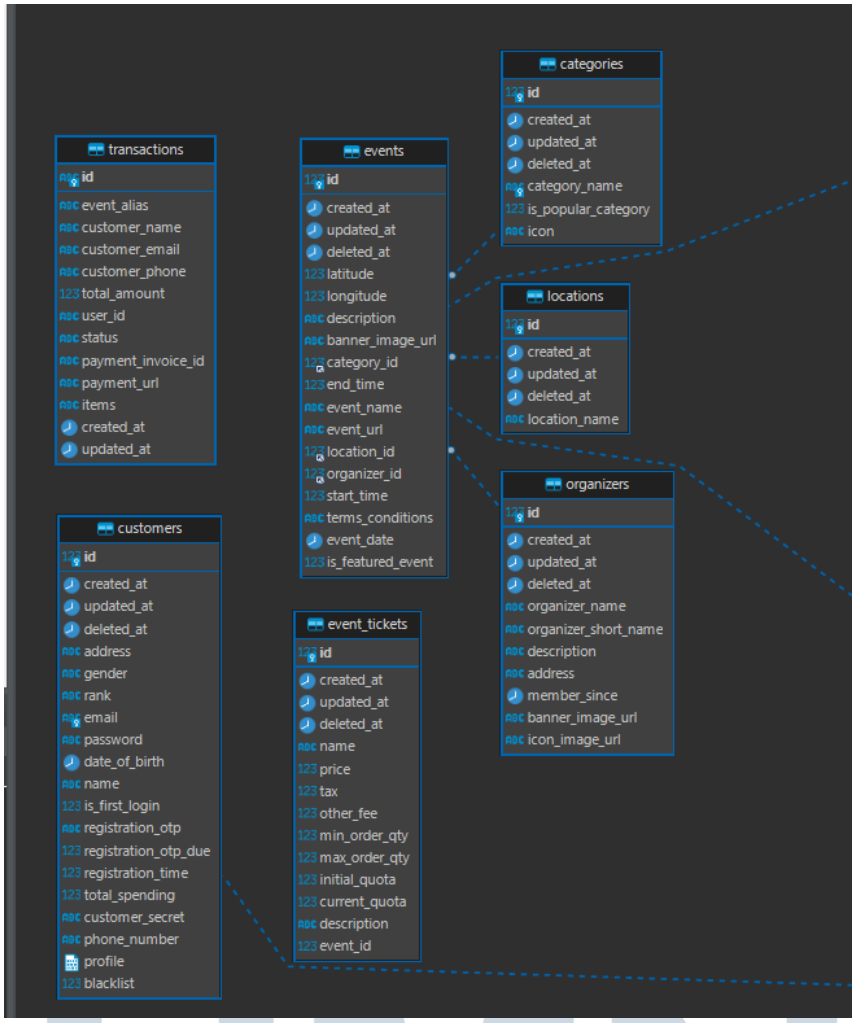
3.3.5 Skema Basis Data

Database schema merupakan representasi arsitektur visual dan logis dari sistem basis data yang digunakan dalam pengembangan *website ticketing* ini. Skema ini mencakup struktur tabel, relasi antar entitas, serta atribut-atribut penting yang digunakan untuk menyimpan dan mengelola data secara efisien. Dengan adanya skema basis data ini, pengelolaan informasi seperti data pengguna, tiket, *event*, artikel, dan lain-lain dapat dilakukan secara terstruktur dan terintegrasi



Gambar 3.12. Skema *database* untuk Artikel

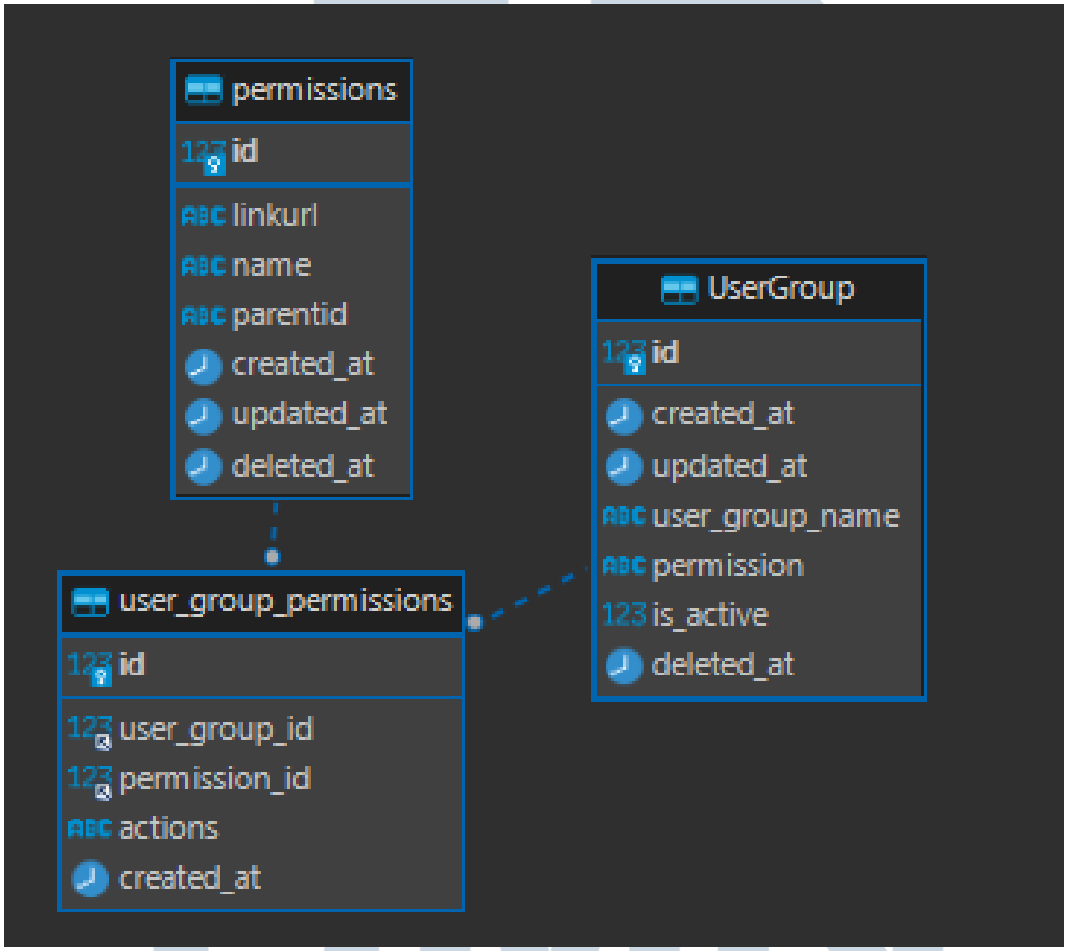
Pada skema basis data ini, *article* menjadi tabel untuk menyimpan data utama pada artikel, yang dimana juga ada tabel *user* sebagai tabel untuk menyimpan informasi dari sang mahasiswa magang, dan juga tabel *article_categories* yang akan menyimpan informasi berupa kategori-kategori pada artikel. Pada tabel utama *articles*, terdapat *author_id* yang akan menjadi parameter untuk mensinkronisasi data *user* dengan tabel *User*. Kemudian, pada tabel utama *articles* juga memiliki *article_category_id*, yang akan menjadi parameter untuk mendapatkan informasi kategori *article* dengan mensinkronisasi *article_category_id* pada tabel *articles* dengan *id* pada *article_categories*.



Gambar 3.13. Skema *database* untuk *tickets*

Pada skema basis data ini, *events* menjadi tabel utama untuk menyimpan data *tickets*. *category_id* akan menjadi parameter untuk mendapatkan/mensinkronkan data dari tabel *categories*, dengan cara mencocokkan *category_id* pada tabel *events* dengan *id* pada tabel *categories*. *Location_id* berfungsi sebagai parameter untuk mendapatkan detail lokasi dari tabel *locations*, dengan cara mencocokkan *location_id* pada tabel *events* dengan *id* pada tabel *locations*. *Organizer_id* pada tabel *events* berfungsi untuk mendapatkan detail dari *organizer* dengan cara mencocokkan *organizer_id* pada tabel *events* dengan *id* pada tabel *organizer*. Tabel *event_tickets* adalah tabel untuk menyimpan jenis-jenis *ticket* yang ada pada suatu *event*, *event_id* pada tabel *event_tickets* merupakan parameter yang digunakan untuk mensinkronisasi antara tabel *event_ticket* dengan tabel *events*. Tabel *transactions* adalah tabel yang digunakan untuk menyimpan

data-data transaksi yang sudah dibeli, tabel ini menggabungkan antara informasi dari *event* dan informasi dari *customer* yang telah membeli *ticket*. Informasi dari *customer* disinkronkan melalui *user_id*, yang dapat dicocokkan dengan *id* pada tabel *customer*



Gambar 3.14. Skema *database* untuk *access control* pada *backend*

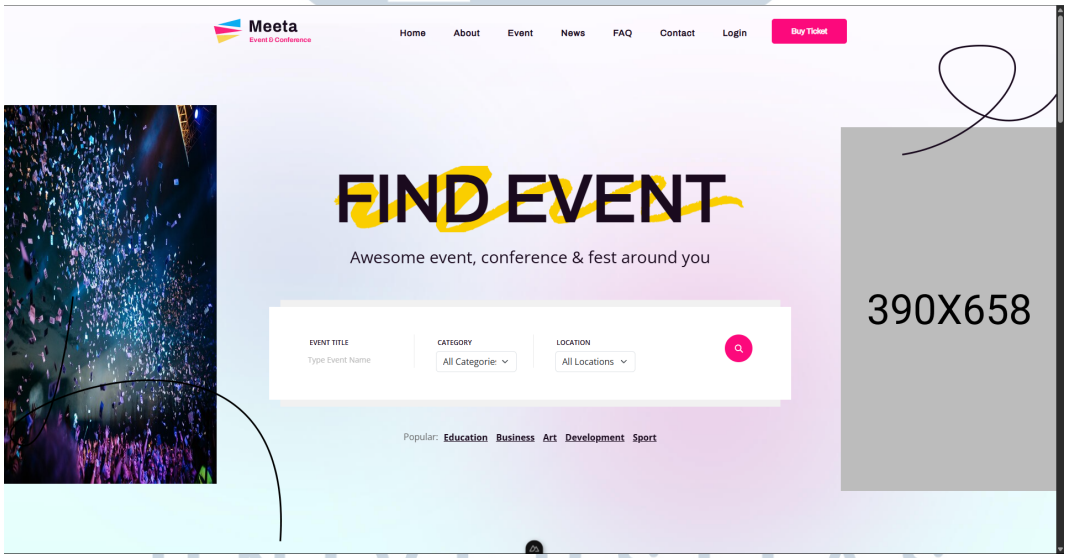
Pada skema basis data ini, tabel *permissions* adalah tabel yang menyimpan jenis-jenis *permission* yang ada. Tabel *UserGroup* adalah tabel yang menyimpan data berupa jenis-jenis grup pengguna, seperti misalnya *admin*, *editor*, ataupun yang lain. Tabel *user_group_permissions* adalah tabel dimana tabel *permissions* dan tabel *UserGroup* saling bersinkron untuk mengatur hak akses, *user_group_id* dicocokkan dengan *id* pada tabel *UserGroup*, dan *permission_id* akan dicocokkan dengan *id* pada tabel *permissions*. Kolom *actions* akan berisi akses-akses yang dapat diakses oleh *id* pada *user group* tertentu dan terhadap *permission*. Kolom *actions* berisi informasi seperti *view*, *add*, *edit*, *delete*

3.3.6 Halaman implementasi Front end

Gambar-gambar berikut menampilkan beberapa halaman antarmuka yang terdapat pada website ticketing yang dirancang selama pelaksanaan magang. Setiap halaman dirancang dengan mempertimbangkan aspek user experience (pengalaman pengguna) dan user interface (antarmuka pengguna), agar mudah digunakan dan dipahami oleh pengguna dari berbagai latar belakang.

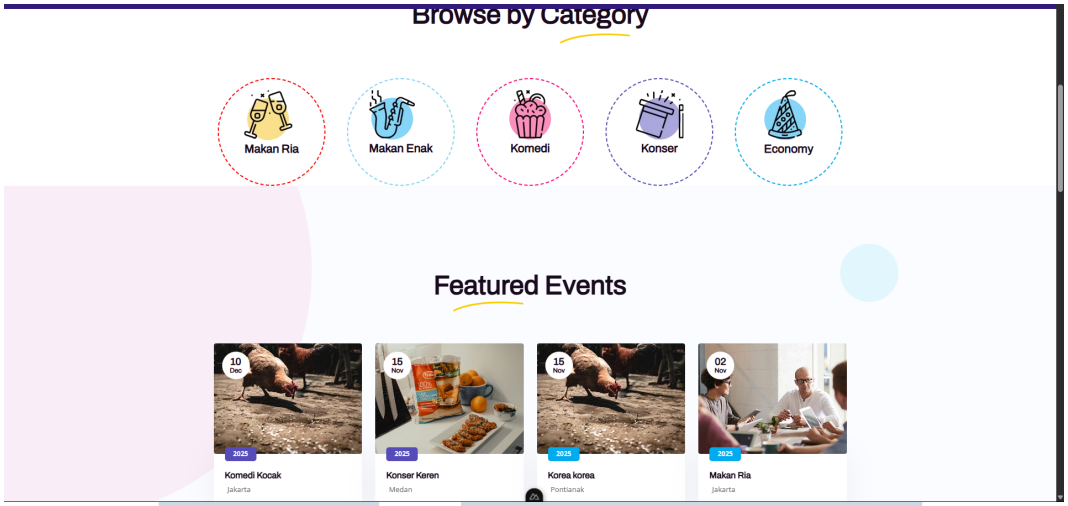
Desain antarmuka difokuskan pada kemudahan navigasi, kejelasan informasi, serta konsistensi elemen visual untuk menciptakan pengalaman yang intuitif dan menyenangkan. Selain itu, tampilan visual pada halaman-halaman ini juga dikembangkan dengan estetika yang menarik dan responsif, sehingga pengguna merasa nyaman saat mengakses layanan, baik melalui perangkat desktop maupun mobile.

Diharapkan melalui pendekatan desain ini, website ticketing dapat meningkatkan kepuasan pengguna dan mempermudah proses pemesanan tiket secara digital.



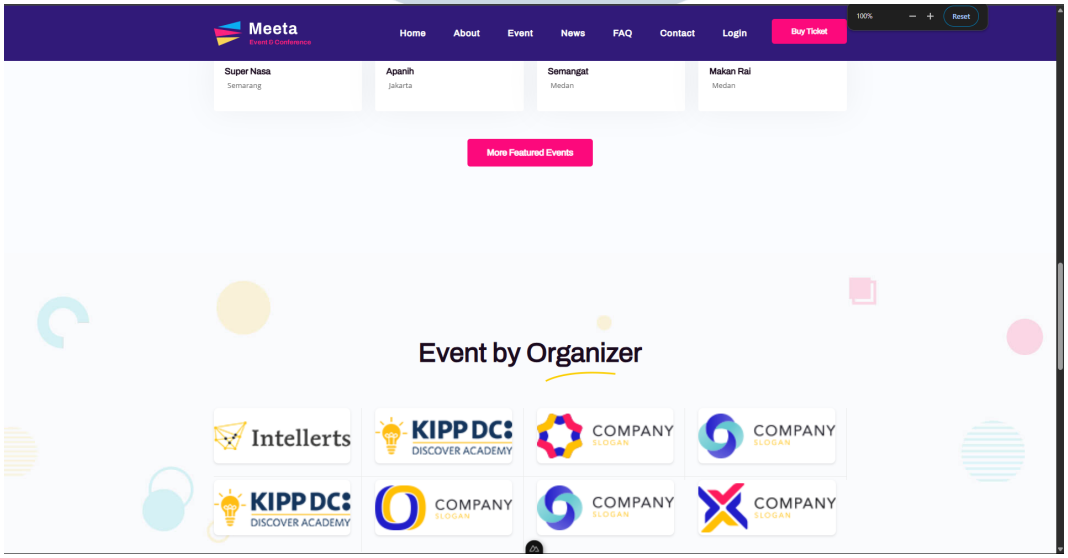
Gambar 3.15. Halaman *Home page*

Selain halaman diatas, jika *user* melakukan penguliran, maka halaman dibawah ini akan terlihat, halaman ini berfungsi untuk langsung mencari *ticket* berdasarkan kategori tertentu, juga ada bagian *featured event*, yang dimana merupakan *event-event* tertentu yang dikhususkan untuk dimunculkan pada bagian *homepage*



Gambar 3.16. Lanjutan halaman *homepage* (Pencarian & *Featured Event*)

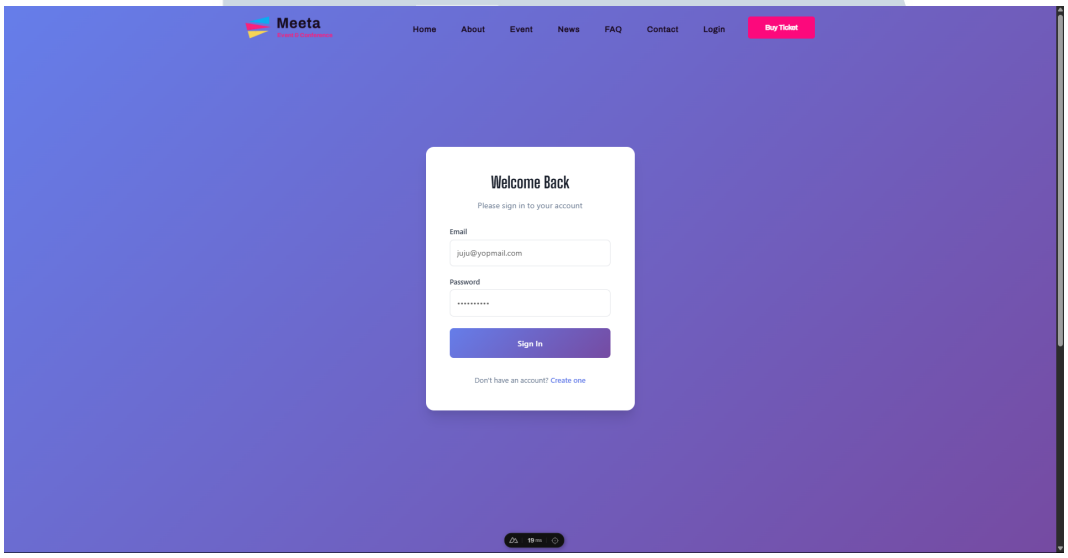
Jika pengguna menggulir halaman ini ke bawah, maka akan muncul halaman berikut. Halaman berisi kumpulan beberapa *organizer*, ini berfungsi agar *user* dapat melihat informasi lebih lanjut dari sebuah *organizer*. Selain itu, juga ada *navigation bar* yang



Gambar 3.17. Lanjutan halaman *homepage* (Daftar *Organizer*)

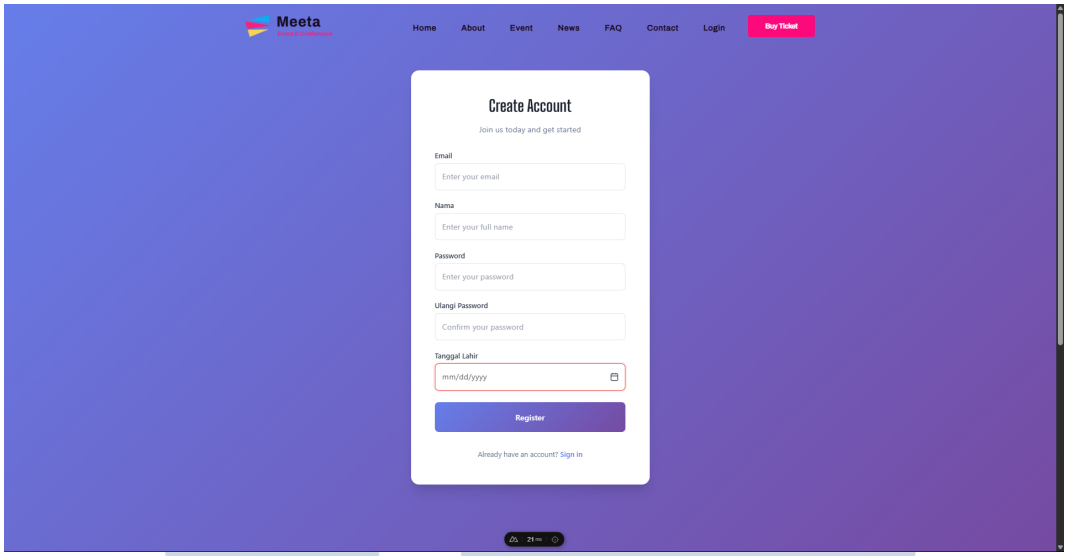
Halaman *homepage* merupakan tampilan pertama yang akan dilihat oleh pengguna saat mereka pertama kali mengakses *website*. Oleh karena itu, halaman ini memiliki peran penting sebagai wajah utama dari keseluruhan sistem, yang mencerminkan identitas, fungsi utama, serta kualitas dari *website* secara keseluruhan. Pada halaman ini, pengguna akan diperkenalkan dengan fitur-

fitur inti, navigasi utama, dan elemen visual yang dirancang untuk memberikan kesan pertama yang positif serta memudahkan pengguna dalam memahami alur penggunaan situs. Desain yang menarik, informatif, dan mudah digunakan sangat diperlukan agar pengguna merasa nyaman dan tertarik untuk menjelajahi halaman-halaman lainnya. Dengan demikian, *homepage* berfungsi tidak hanya sebagai pintu masuk, tetapi juga sebagai titik awal untuk membangun pengalaman pengguna yang baik dan interaktif.



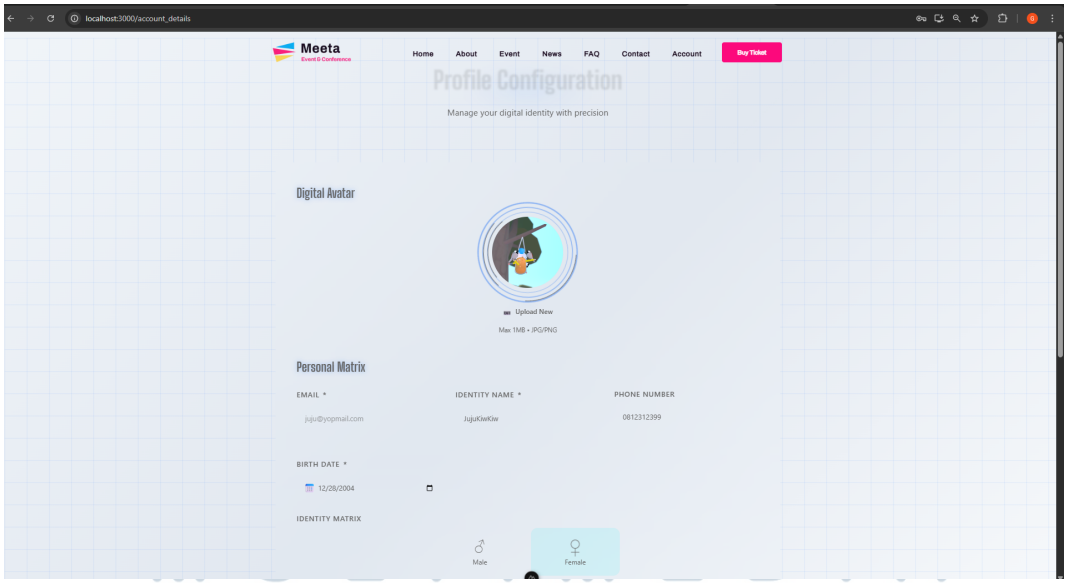
Gambar 3.18. *Login page*

Halaman *Login* merupakan halaman yang digunakan oleh pengguna untuk masuk ke dalam akun mereka sebelum dapat mengakses fitur-fitur utama yang bersifat personal, seperti melakukan *checkout* atau pembelian tiket. Melalui halaman ini, pengguna dapat memasukkan kredensial berupa alamat *email* dan kata sandi yang telah didaftarkan sebelumnya. Autentikasi yang dilakukan pada halaman *login* berfungsi sebagai langkah pengamanan agar hanya pengguna yang terdaftar yang dapat melakukan transaksi dan mengakses data pribadi mereka. Selain itu, halaman ini juga dapat menyediakan opsi tambahan seperti ”Lupa Kata Sandi” atau tautan menuju halaman registrasi bagi pengguna baru. Dengan adanya sistem *login*, *website* dapat memastikan pengalaman pengguna yang lebih aman, personal, dan terintegrasi dengan histori transaksi atau aktivitas lainnya di dalam platform.



Gambar 3.19. *Register page*

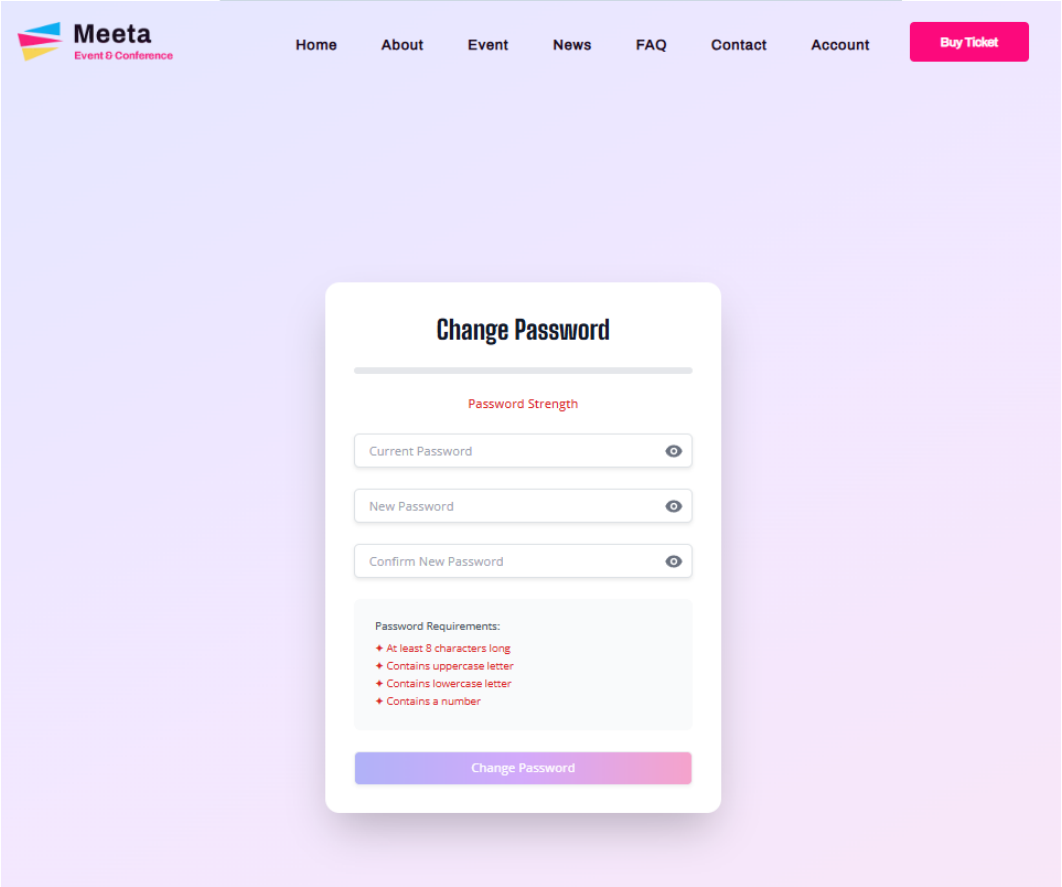
Halaman *Register* adalah halaman yang disediakan bagi pengguna baru untuk membuat akun di dalam sistem. Pada halaman ini, pengguna diminta untuk mengisi data diri mereka, seperti nama lengkap, alamat *email*, dan kata sandi, sebagai syarat untuk melakukan pendaftaran.



Gambar 3.20. *Account Detail Page*

Halaman *Account Detail* merupakan halaman yang memungkinkan pengguna untuk melihat serta memperbarui informasi pribadi mereka yang telah terdaftar dalam sistem. Melalui halaman ini, pengguna dapat mengelola data

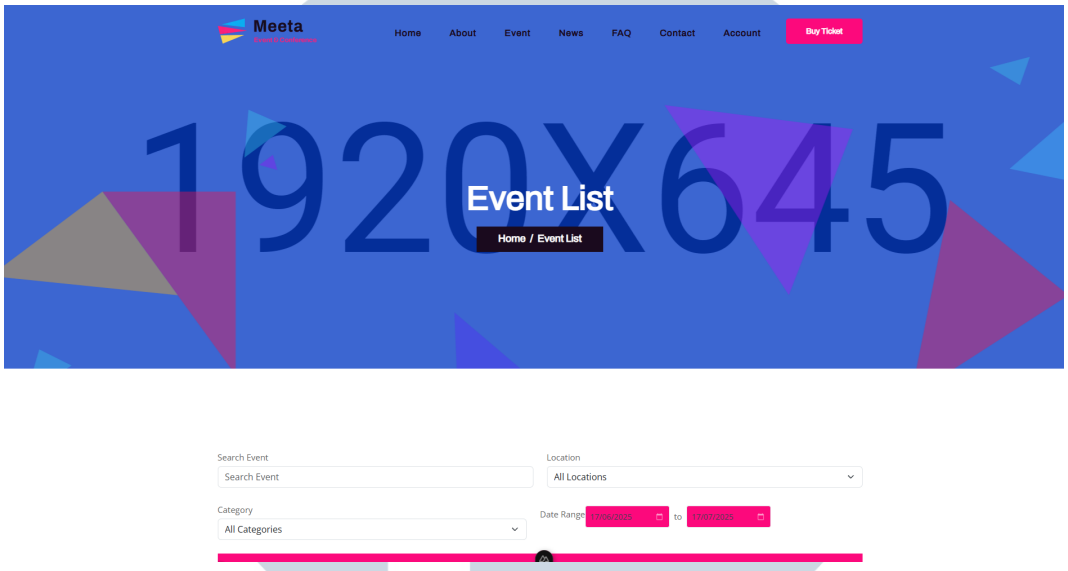
diri secara mandiri, termasuk mengganti foto profil, memperbarui nama lengkap, nomor telepon, tanggal lahir, serta memilih atau mengubah informasi terkait jenis kelamin. Fitur ini dirancang untuk memberikan fleksibilitas dan kenyamanan kepada pengguna dalam menjaga data mereka tetap akurat dan terkini. Dengan adanya halaman ini, sistem dapat memastikan bahwa informasi pengguna yang tersimpan selalu relevan dan sesuai dengan kebutuhan personalisasi layanan yang tersedia di dalam platform.



Gambar 3.21. *Change Password Page*

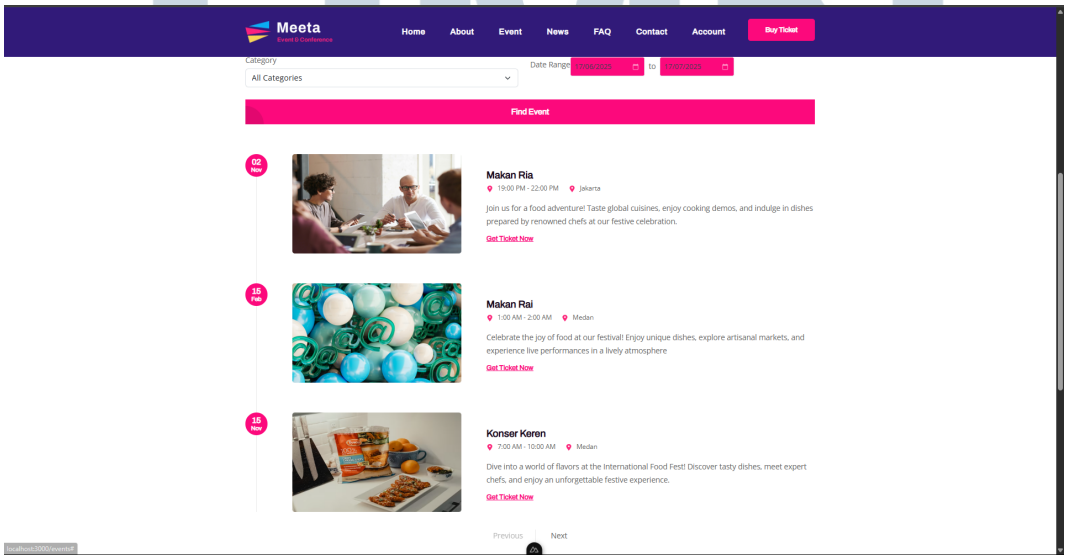
Halaman *Change Password* berfungsi sebagai sarana bagi pengguna untuk mengubah kata sandi akun mereka secara mandiri. Untuk menjaga keamanan akun, pengguna diwajibkan untuk memasukkan kata sandi saat ini sebagai bentuk verifikasi identitas, kemudian menyertakan kata sandi baru yang diinginkan. Sistem akan memvalidasi kata sandi baru tersebut agar memenuhi standar keamanan tertentu, yaitu harus terdiri dari minimal 8 karakter, mengandung huruf kapital, huruf kecil, serta angka. Dengan ketentuan tersebut, halaman ini bertujuan untuk memastikan bahwa perubahan kata sandi dilakukan secara sah dan menghasilkan

kombinasi yang kuat, sehingga dapat melindungi akun pengguna dari potensi akses tidak sah atau penyalahgunaan. Fitur ini menjadi bagian penting dalam menjaga privasi dan integritas data pengguna di dalam sistem.



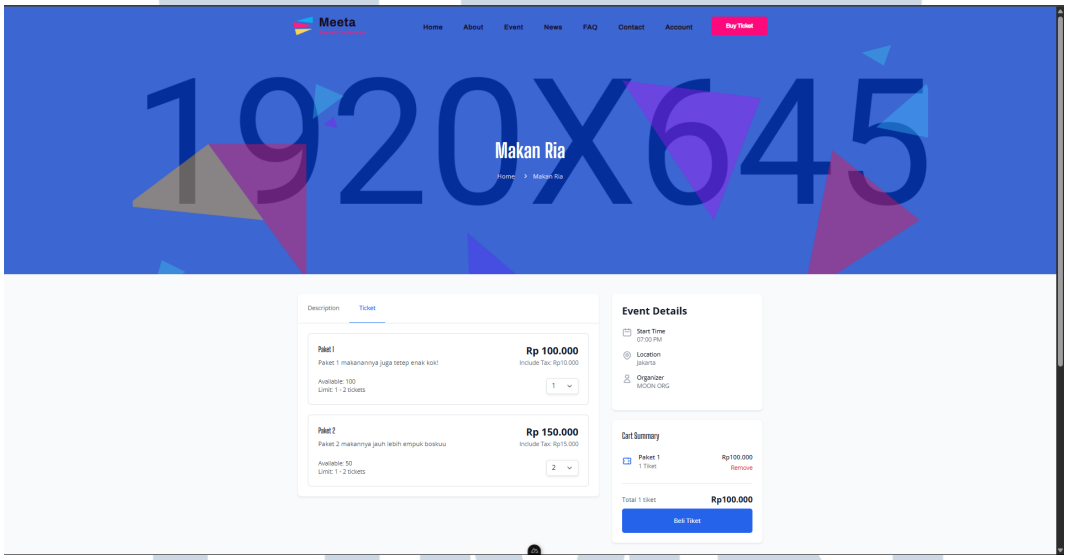
Gambar 3.22. *Event List Page*

Jika pengguna menggulir halaman ini ke bawah, maka halaman berikut akan muncul. Halaman ini merupakan halaman dimana *event* yang telah dicari dengan parameter diatas muncul. Terdapat informasi singkat seperti judul *event*, deskripsi, dan juga dapat ditekan dan masuk ke *detail event page*



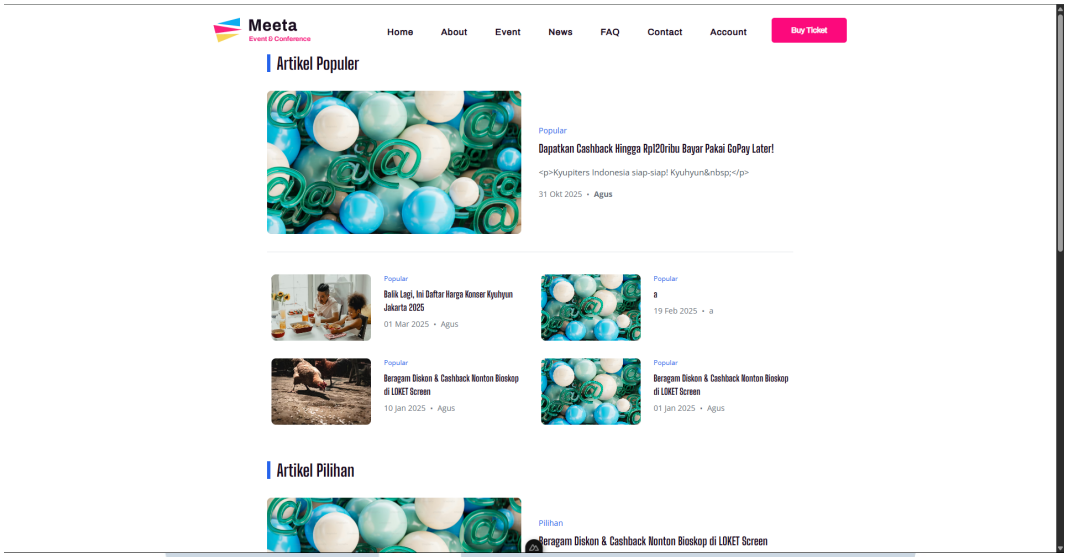
Gambar 3.23. Lanjutan *Event List Page* (Hasil Pencarian)

Halaman *Event List* merupakan halaman yang menyajikan daftar *event* yang tersedia dan dapat diakses oleh seluruh pengguna. Di halaman ini, pengguna dapat dengan mudah menelusuri berbagai *event* melalui fitur pencarian dan *filter* yang disediakan. *Filter* yang tersedia mencakup pencarian berdasarkan nama *event*, lokasi, rentang tanggal pelaksanaan, hingga kategori *event* tertentu. Fitur ini memudahkan pengguna untuk menemukan *event* yang sesuai dengan minat dan kebutuhan mereka. Setiap *event* yang ditampilkan pada daftar dapat diklik untuk diarahkan ke halaman *event detail*, di mana informasi lengkap mengenai *event* tersebut akan ditampilkan. Dengan demikian, halaman *event list* berperan penting dalam memfasilitasi eksplorasi dan pemilihan *event* secara efisien dan interaktif.



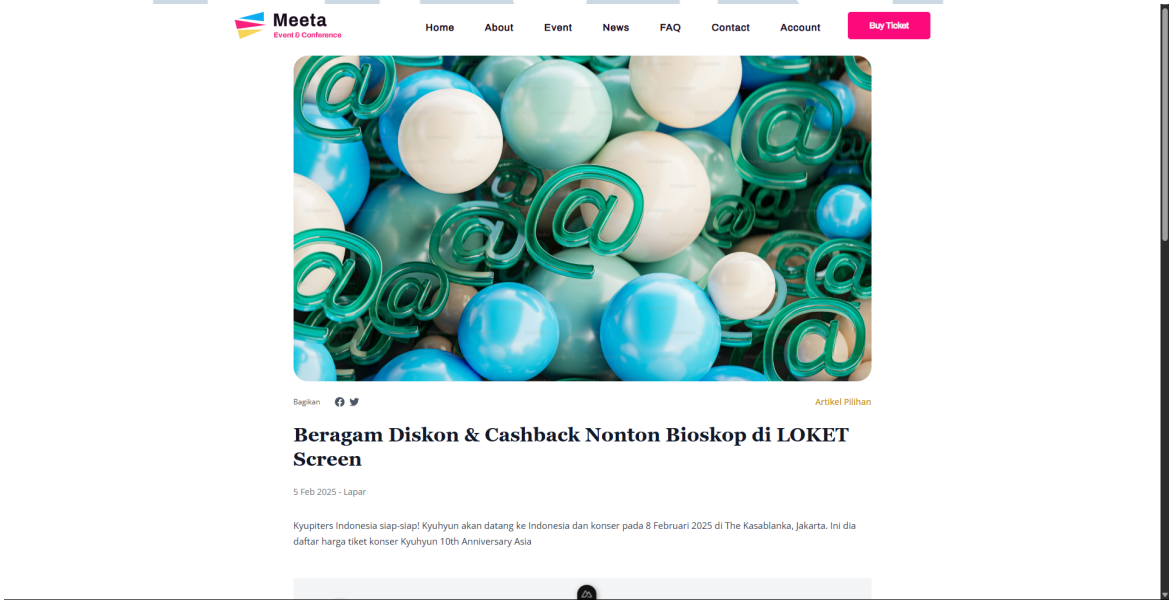
Gambar 3.24. *Event Detail Page*

Halaman *event detail page* merupakan halaman yang menampilkan isi lengkap dari artikel yang telah dipilih oleh pengguna. Di halaman ini, pengguna dapat membaca secara menyeluruh konten artikel yang mencakup informasi, berita, atau panduan yang relevan dengan tema *website*. Tampilan halaman dirancang agar nyaman dibaca, baik di perangkat *desktop* maupun *mobile*. Selain itu, tersedia fitur artikel terkait yang secara otomatis menampilkan daftar artikel lain dengan kategori yang sama dan waktu publikasi terbaru. Fitur ini bertujuan untuk meningkatkan keterlibatan pengguna serta memudahkan mereka dalam menemukan konten lain yang sejenis dan bermanfaat, sehingga memperpanjang waktu interaksi pengguna dengan platform secara keseluruhan.



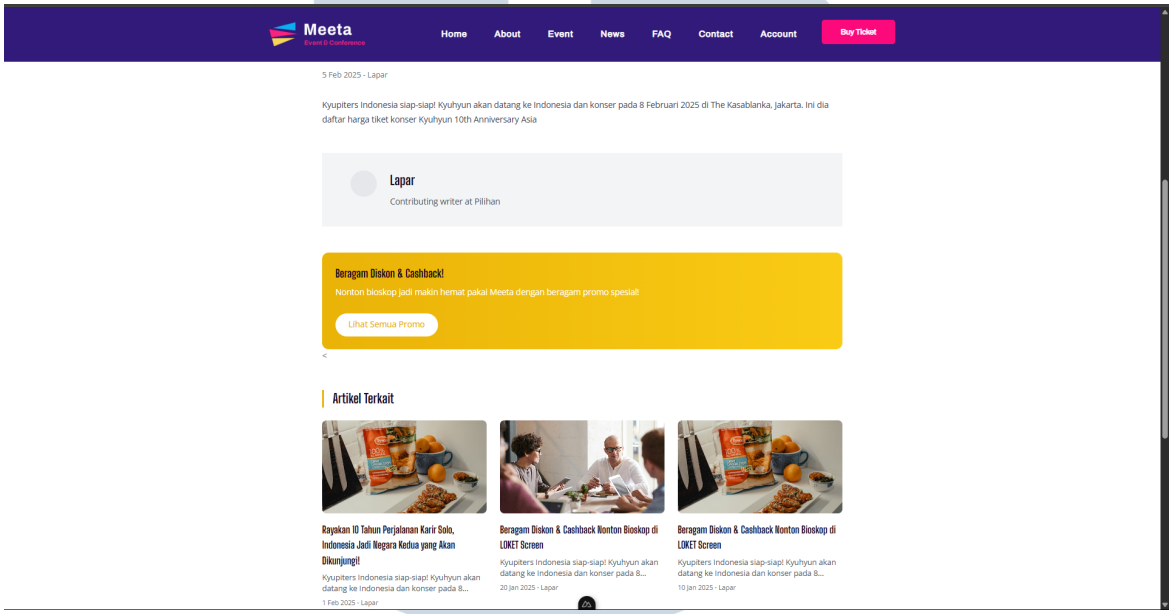
Gambar 3.25. *Article List Page*

Halaman *list artikel* merupakan halaman yang menampilkan kumpulan artikel yang tersedia di dalam platform. Pada halaman ini, pengguna dapat menelusuri berbagai artikel yang telah dipublikasikan dan memilih salah satu untuk dibaca lebih lanjut. Setiap artikel biasanya ditampilkan dalam bentuk ringkasan atau cuplikan singkat, disertai dengan judul, gambar, tanggal publikasi, serta kategori agar pengguna lebih mudah mengenali topik yang dibahas. Secara garis besar, artikel dibagi menjadi 2, yaitu artikel yang populer dan artikel pilihan



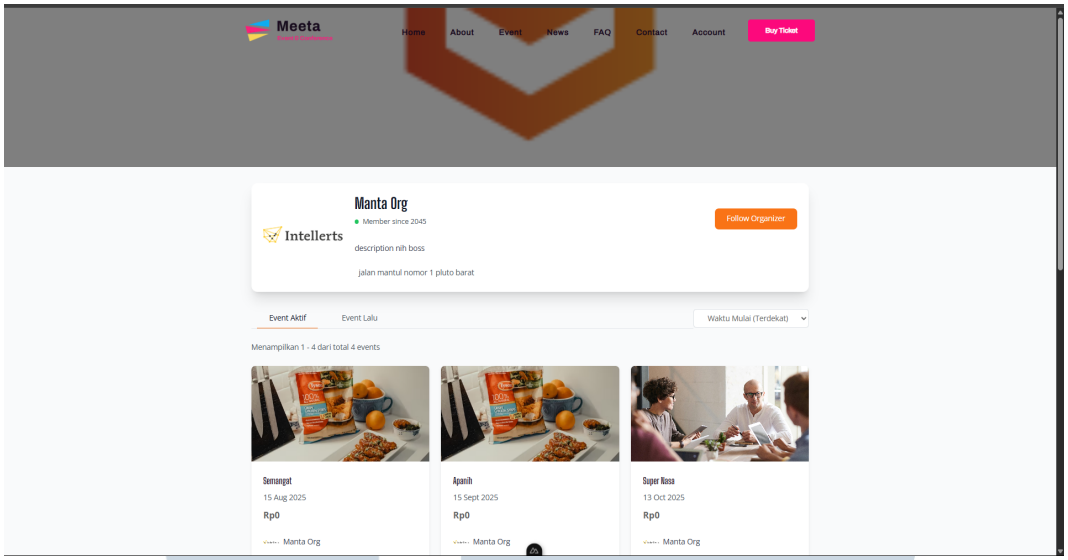
Gambar 3.26. *Article Detail Page*

Jika *user* melakukan penguliran, maka bagian Artikel Terkait akan terlihat. Bagian artikel terkait merupakan rekomendasi artikel yang dapat dibaca oleh *user*. Artikel yang muncul merupakan artikel dengan kategori yang sama dengan artikel yang sedang dibaca oleh *user*.



Gambar 3.27. Lanjutan *Article Detail Page* (Artikel Terkait)

Halaman *article detail page* adalah halaman yang menampilkan isi lengkap dari artikel yang telah dipilih oleh pengguna. Di halaman ini, pengguna dapat membaca seluruh konten artikel secara nyaman, baik melalui perangkat *desktop* maupun *mobile*. Artikel ditampilkan secara terstruktur dengan judul, tanggal publikasi, serta isi yang disajikan dalam format yang mudah dibaca. Selain itu, halaman ini juga dilengkapi dengan fitur artikel terkait, yang secara otomatis menampilkan artikel lain dengan kategori yang sama dan waktu publikasi terbaru. Fitur ini dirancang untuk membantu pengguna menemukan konten sejenis yang relevan dengan minat mereka, sehingga meningkatkan keterlibatan pengguna dan memperkaya pengalaman membaca dalam platform. Dengan adanya fitur ini, pengguna diharapkan dapat membaca artikel sesuai dengan minatnya. Kemudian, bagian artikel terkait ini tidak akan memunculkan artikel yang sedang dibaca untuk menghindari *loop* yang tidak diinginkan terjadi.

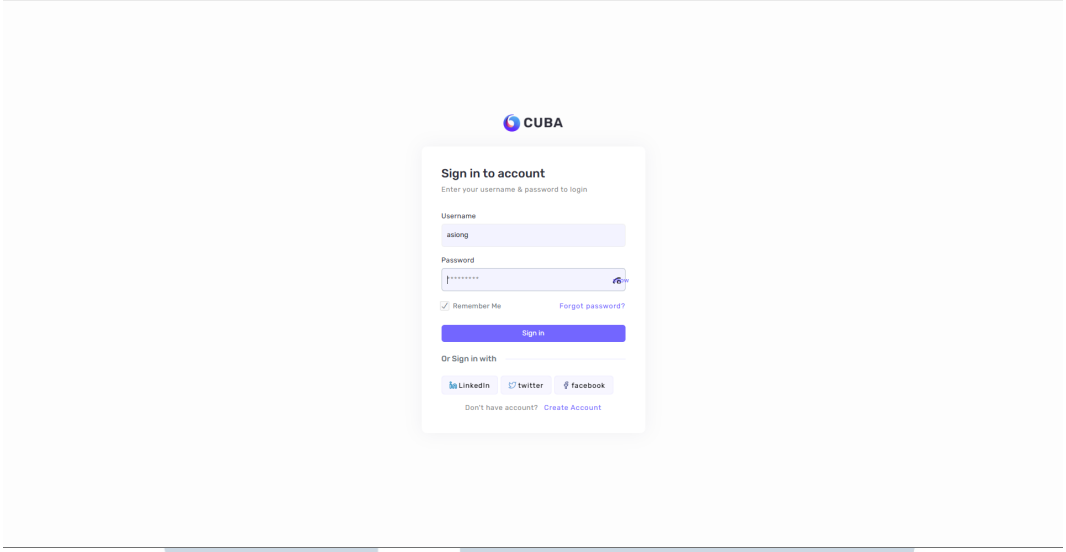


Gambar 3.28. *Detail Organizer Page*

Halaman *detail organizer* merupakan halaman yang menampilkan informasi lengkap mengenai suatu penyelenggara (*organizer*) *event*. Di halaman ini, pengguna dapat melihat berbagai detail penting seperti nama *organizer*, alamat, deskripsi singkat mengenai profil atau latar belakang *organizer*, serta daftar *event* yang diselenggarakan. Daftar *event* tersebut dikelompokkan menjadi dua kategori, yaitu *event* yang masih aktif (akan datang atau sedang berlangsung) dan *event* yang telah berlalu. Dengan adanya pembagian ini, pengguna dapat dengan mudah menelusuri riwayat maupun agenda *event* dari *organizer* yang bersangkutan. Halaman ini dirancang untuk memberikan gambaran menyeluruh mengenai kredibilitas dan aktivitas *organizer*, serta memudahkan pengguna dalam menemukan *event-event* menarik dari penyelenggara tertentu.

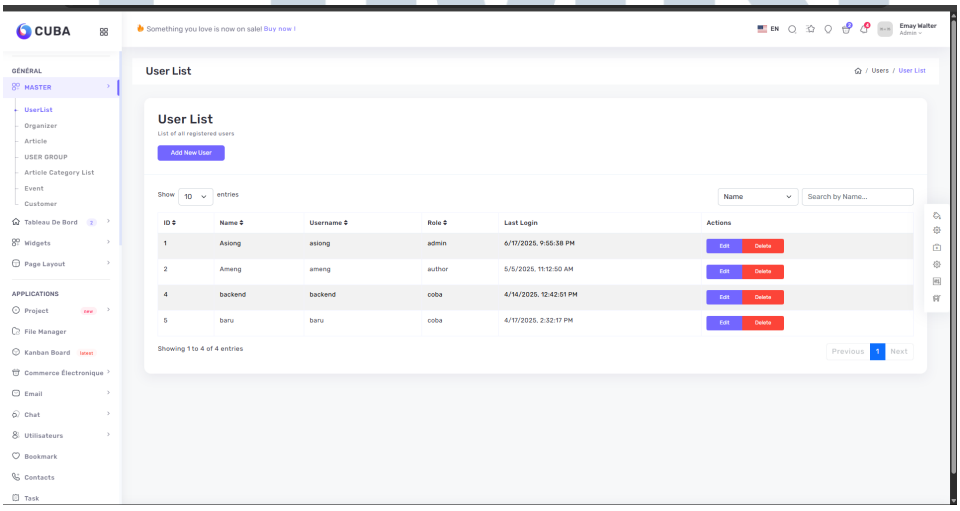
3.3.7 Halaman Implementasi Backend

Gambar-gambar dibawah ini adalah halaman yang terdapat pada *website ticketing backend*. Halaman-halaman ini berfungsi untuk mengubah konten, maupun data untuk mempermudah jalannya *website ticketing* ini. Halaman-halaman ini dibuat sesederhana mungkin, agar pengguna dapat memahami fitur-fitur dari halaman ini dengan cepat dan baik.



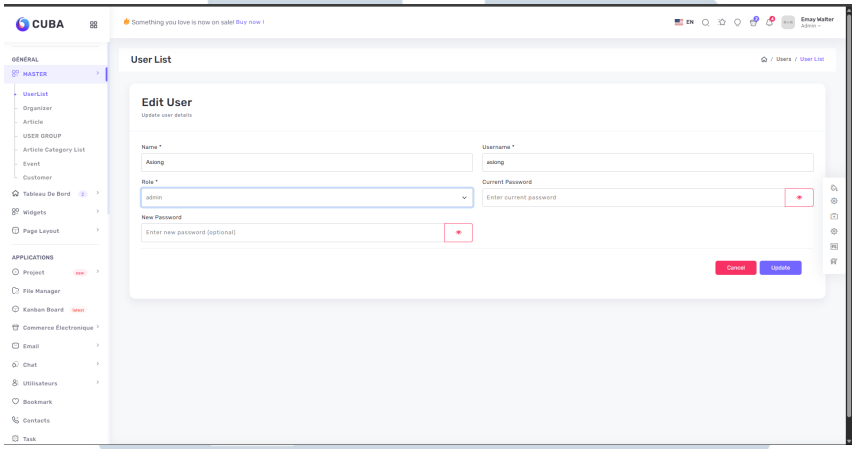
Gambar 3.29. *Login Page Backend*

Halaman *login* pada sisi *backend* berfungsi sebagai titik masuk bagi pengguna untuk mengakses sistem. Pada halaman ini, pengguna diminta untuk memasukkan *username* dan *password* yang telah terdaftar. Setelah proses autentikasi berhasil dilakukan, sistem akan mengenali identitas pengguna dan menentukan hak akses yang dimilikinya berdasarkan *user group*. Pengguna kemudian akan diarahkan ke halaman utama dan hanya dapat mengakses fitur-fitur yang sesuai dengan peran mereka di dalam sistem. Mekanisme ini bertujuan untuk menjaga keamanan dan memastikan bahwa setiap pengguna hanya dapat menggunakan fungsi yang relevan dengan tanggung jawabnya.



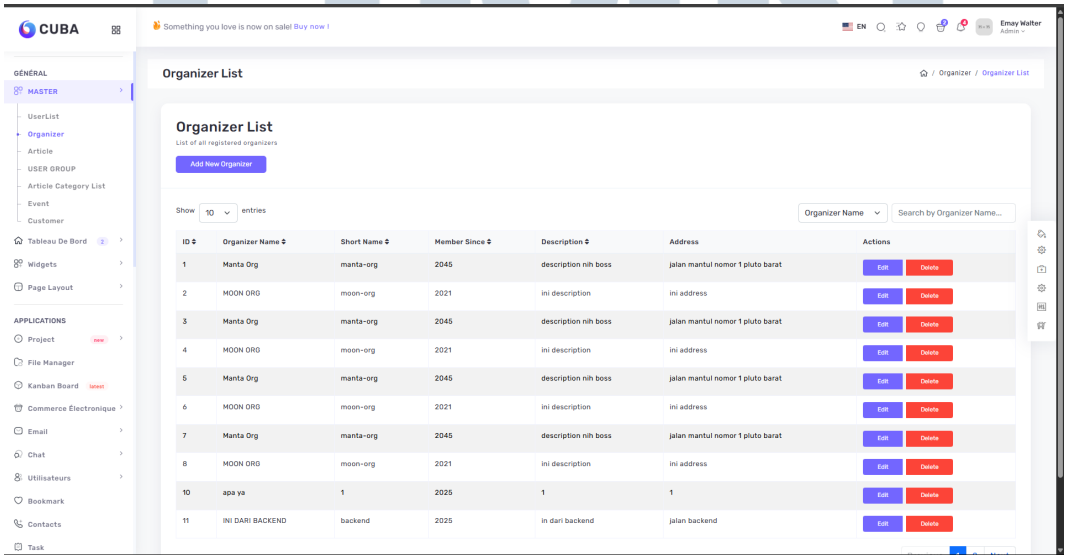
Gambar 3.30. *User List Page*

Jika pengguna menekan tombol *edit*, maka akan muncul tampilan seperti berikut. Halaman ini berfungsi agar pengguna dapat menyunting data dari *user* yang ada. Hanya *user* dengan hak akses *edit* pada *edit user page* yang dapat melakukan aksi ini.



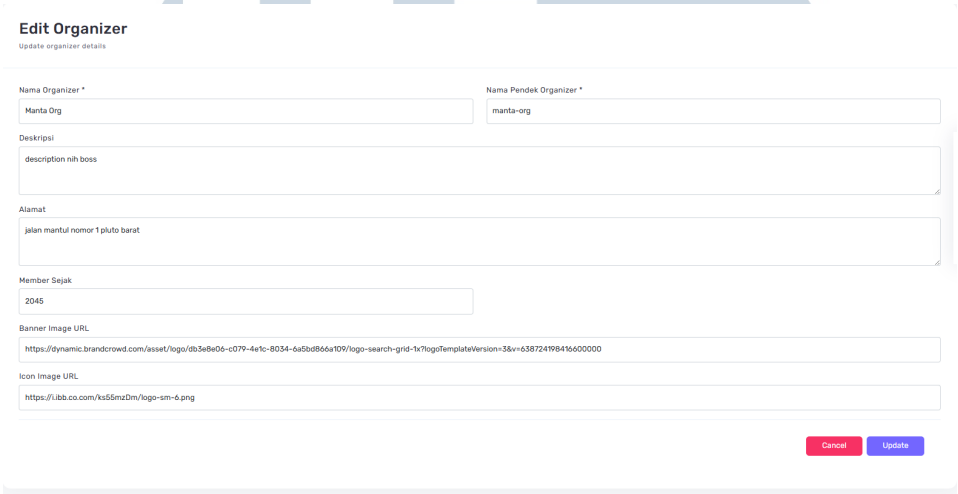
Gambar 3.31. *Edit User List Page*

Halaman *user list* merupakan halaman yang menampilkan daftar seluruh pengguna *backend* yang terdaftar dalam sistem. Setiap entri pengguna ditampilkan dalam bentuk tabel yang berisi informasi seperti nama, *username*, dan peran (*role*) masing-masing. Melalui halaman ini, *admin* atau pengguna dengan hak akses tertentu dapat melakukan pengelolaan data pengguna secara langsung. Kolom-kolom yang dapat diedit mencakup nama, *username*, *role*, serta kata sandi pengguna



Gambar 3.32. *Organizer List Page*

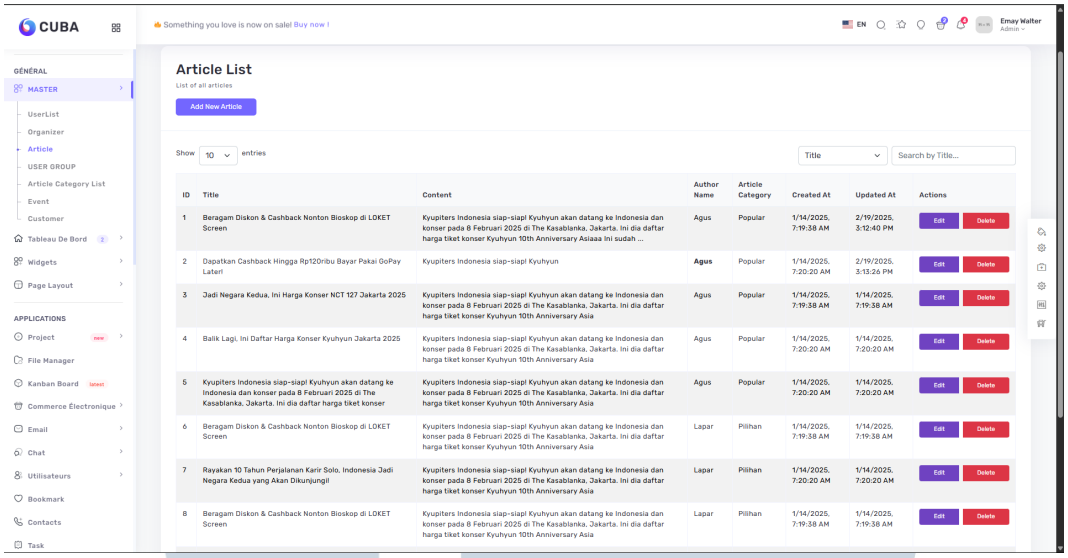
Apabila pengguna menekan tombol *edit* pada salah satu entri *organizer*, sistem akan menampilkan sebuah halaman *form edit* yang dirancang khusus untuk memungkinkan proses pembaruan data. Halaman ini berfungsi sebagai antarmuka bagi pengguna—dalam hal ini *admin*—untuk melakukan perubahan terhadap informasi yang terkait dengan penyelenggara (*organizer*) yang dipilih.



Gambar 3.33. *Edit Organizer Page*

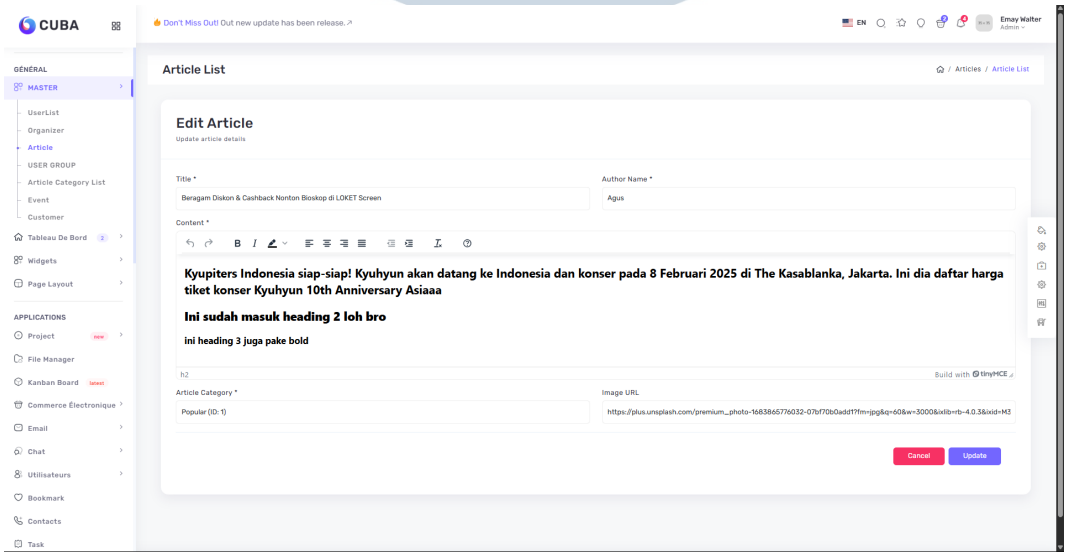
Halaman *organizer list* merupakan fitur penting dalam sistem *website ticketing* yang berfungsi untuk menampilkan daftar lengkap seluruh penyelenggara (*organizer*) yang telah terdaftar di dalam sistem. Halaman ini dirancang dengan tampilan yang terstruktur agar memudahkan pengguna dengan hak akses tertentu, seperti *admin*, dalam melakukan monitoring dan manajemen data *organizer*. Setiap entri dalam daftar ini menyajikan informasi dasar penyelenggara, seperti nama, kontak, dan status keaktifan.

Selain menampilkan data, halaman ini juga terintegrasi dengan fitur *edit organizer* yang memungkinkan *admin* untuk menyunting atau memperbarui informasi penyelenggara secara langsung. Melalui fitur ini, *admin* dapat melakukan perubahan pada data penting seperti nama *organizer*, alamat, deskripsi, maupun data administratif lainnya sesuai kebutuhan operasional. Kehadiran fitur ini bertujuan untuk meningkatkan efisiensi dalam pengelolaan data serta memastikan bahwa informasi yang tersaji kepada pengguna akhir selalu akurat, terkini, dan relevan dengan kondisi sebenarnya.



Gambar 3.34. *Article List Page Backend*

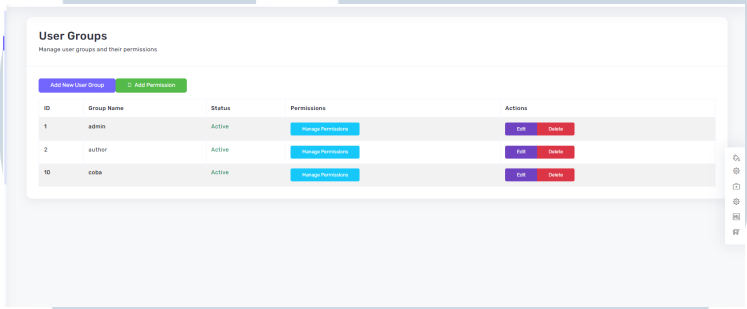
Jika pengguna menekan tombol *edit*, maka akan muncul tampilan seperti berikut. Halaman ini adalah halaman dimana *user* dapat mengubah informasi mendetail dari sebuah *article*.



Gambar 3.35. *Edit Article Page*

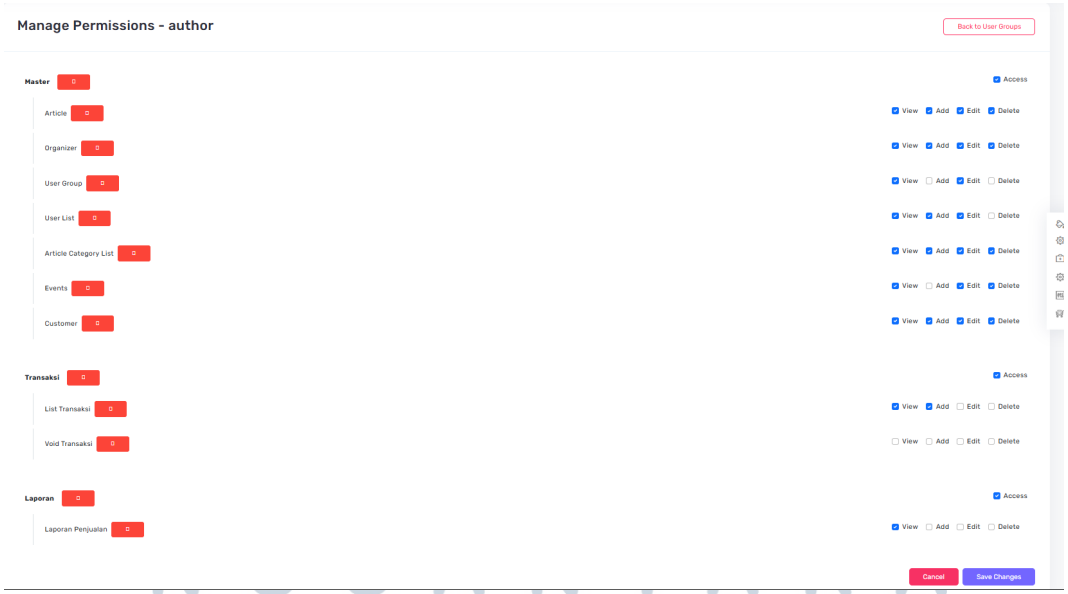
Halaman *article list* adalah halaman yang digunakan untuk melihat seluruh artikel yang telah diterbitkan di dalam sistem *website ticketing*. Artikel-artikel tersebut ditampilkan dalam bentuk daftar yang memudahkan pengguna *backend*, seperti *admin* atau *editor*, untuk mengelola konten secara terpusat. Setiap artikel dapat disunting melalui halaman *Edit Article Page*, di mana pengguna dapat

memperbarui berbagai data artikel seperti judul, kategori, maupun kontennya. Untuk mempermudah proses penyuntingan konten, halaman ini telah dilengkapi dengan fitur WYSIWYG (*What You See Is What You Get*), yaitu *editor* teks visual yang memungkinkan pengguna menyunting konten artikel secara langsung dengan berbagai opsi pemformatan seperti *bold*, *italic*, pembuatan daftar, penyisipan tautan, gambar, dan lainnya, tanpa perlu menulis kode HTML secara manual. Dengan adanya fitur ini, proses pengelolaan konten menjadi lebih intuitif dan efisien.



Gambar 3.36. *User Group List Page*

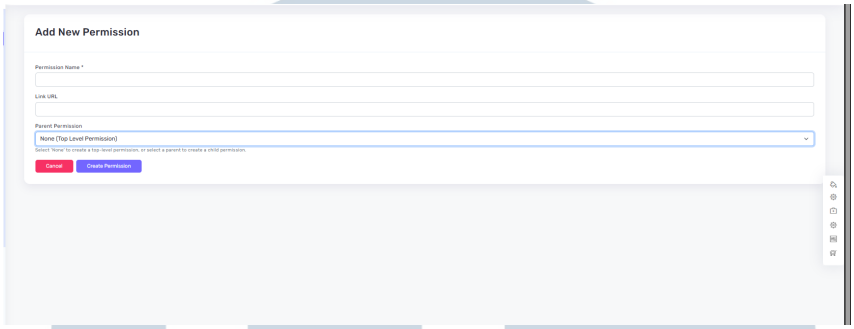
Jika pengguna menekan tombol *manage permission*, maka akan muncul tampilan seperti berikut. Halaman ini merupakan halaman dimana *user* dapat mengubah hak akses dari suatu grup pengguna.



Gambar 3.37. *Manage Permission Page*

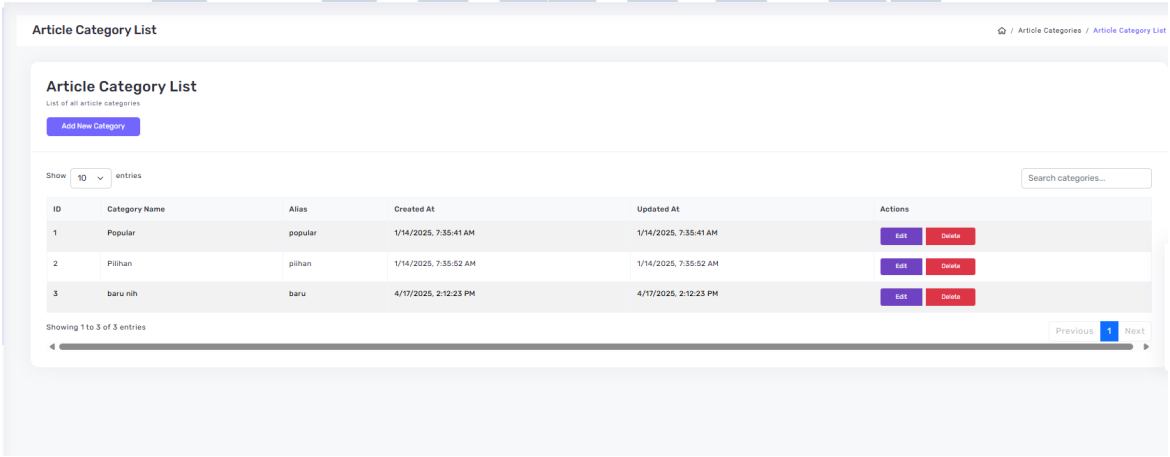
Ketika pengguna menekan tombol ”*Add New Permission*”, sistem akan

menampilkan sebuah halaman *input* khusus yang memungkinkan pengguna untuk menambahkan *permission* (izin akses) baru ke dalam sistem.



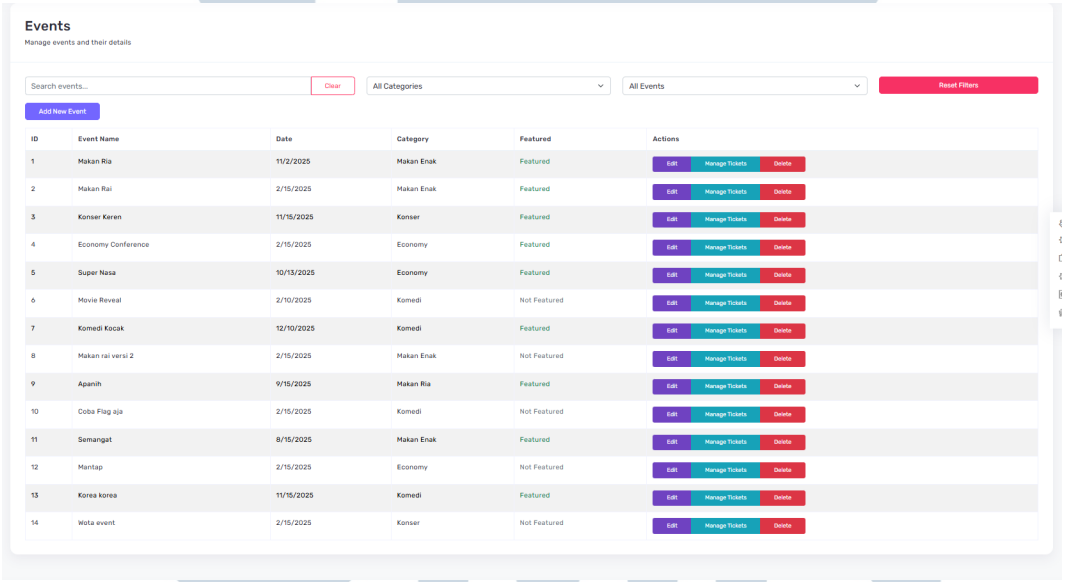
Gambar 3.38. *Add New Permission Page*

Halaman *user group list* adalah halaman yang menampilkan daftar seluruh kelompok pengguna (*user group*) yang terdaftar di sistem *backend*. Setiap *user group* merepresentasikan peran tertentu dalam sistem dan memiliki hak akses yang berbeda-beda. Untuk mengelola hak akses tersebut, tersedia fitur *manage permission* yang memungkinkan *admin* untuk mengatur tingkat akses dari masing-masing *user group*. Hak akses yang dapat dikonfigurasi mencakup empat jenis utama, yaitu *view*, *add*, *edit*, dan *delete*, yang dapat diterapkan pada berbagai modul atau fitur dalam sistem. Selain itu, terdapat halaman *add new permission* yang digunakan untuk menambahkan jenis akses baru ke dalam sistem. Pada halaman ini, *admin* juga dapat menentukan struktur hierarki akses dengan menetapkan *parent* atau akar dari *permission* yang dibuat. Dengan sistem ini, pengelolaan hak akses menjadi fleksibel dan dapat disesuaikan dengan kebutuhan organisasi.



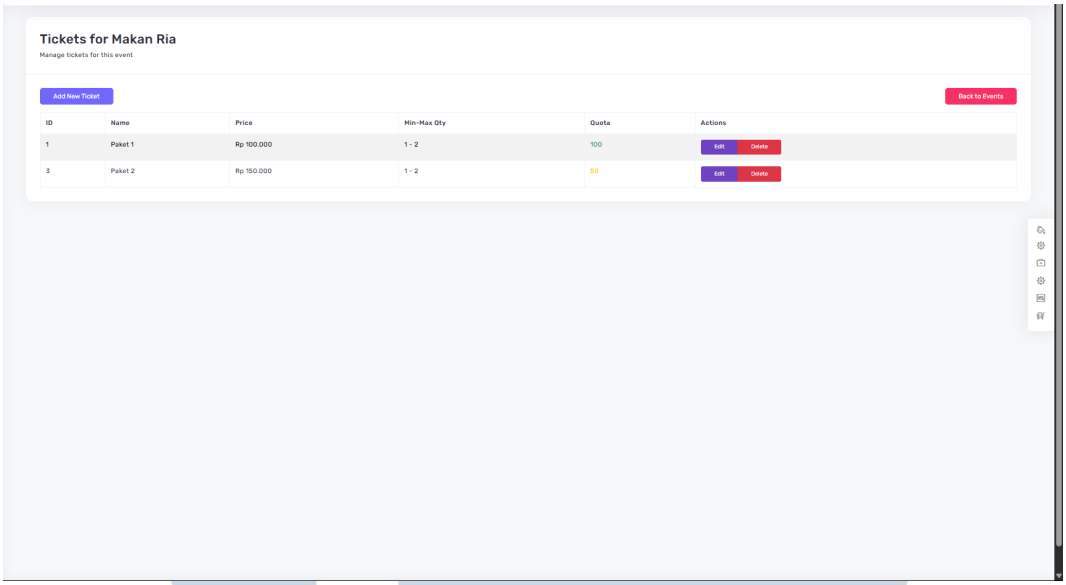
Gambar 3.39. *Article Category List*

Halaman *article category list* merupakan halaman yang menampilkan seluruh kategori artikel yang tersedia dalam sistem. Melalui halaman ini, pengguna yang memiliki hak akses yang sesuai dapat melakukan berbagai tindakan pengelolaan kategori, seperti menambahkan kategori artikel baru, menyunting kategori yang sudah ada, maupun menghapusnya. Setiap tindakan tersebut dibatasi berdasarkan hak akses masing-masing pengguna, yang diatur melalui sistem *user group* dan *permission*. Dengan adanya halaman ini, pengelompokan artikel menjadi lebih terstruktur sehingga memudahkan proses manajemen konten serta membantu pengguna dalam menelusuri artikel berdasarkan kategori yang relevan. Halaman ini juga berhubungan dengan halaman *article list*, dimana *article* akan mengaitkan jenis kategori artikelnya dengan artikel kategori yang ada disini. Selain itu, juga terdapat fitur pencarian, fitur pencarian ini dapat dilakukan dengan menuliskan nama kategori



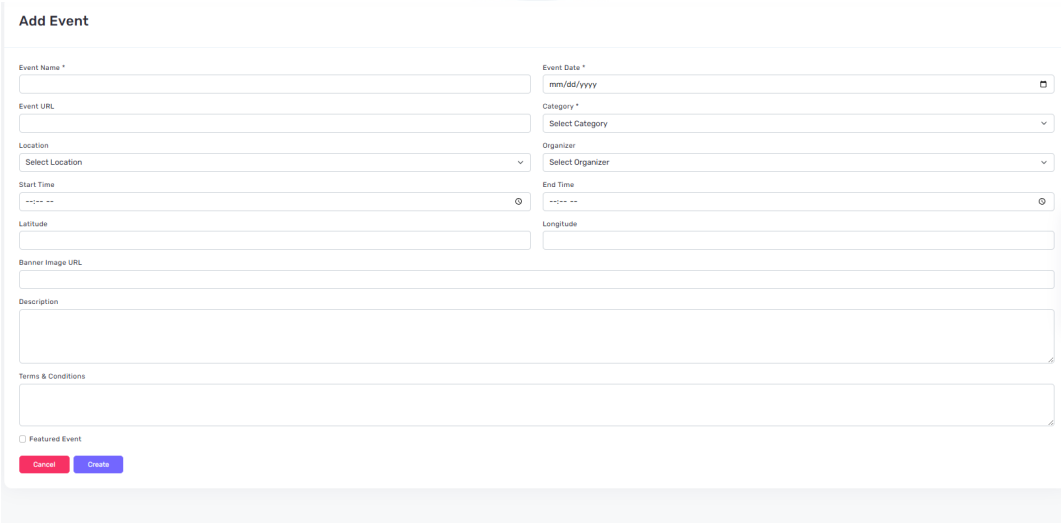
Gambar 3.40. *Event List Page Backend*

Jika pengguna menekan tombol *manage ticket*, maka akan muncul tampilan seperti berikut. Halaman *manage tiket* dapat digunakan *user* untuk mengelola jenis-jenis tiket yang ada pada suatu *event*. Setiap jenis tiket dapat diatur secara detail, seperti nama tiket, harga, kuota, batas waktu penjualan, serta fitur tambahan lainnya



Gambar 3.41. *Manage Ticket Page*

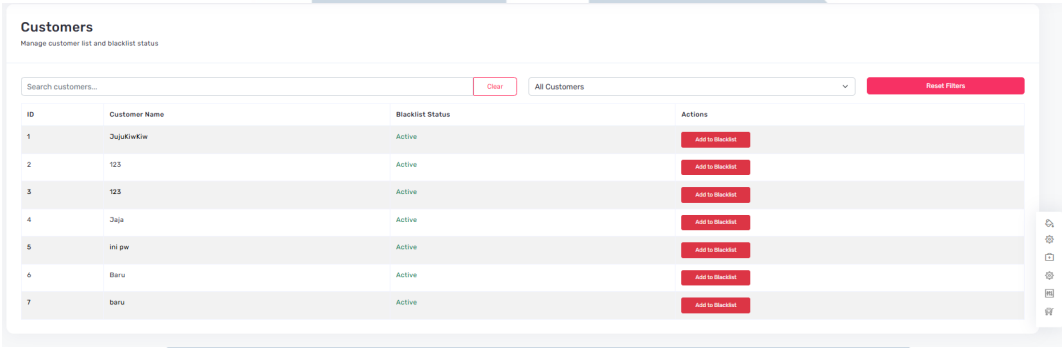
Jika pengguna menekan tombol *add new event*, maka akan muncul tampilan seperti berikut. Halaman ini berfungsi sebagai wadah bagi *user* untuk menambahkan *event* baru.



Gambar 3.42. *Add New Event Page*

Halaman *event list page* adalah halaman yang menampilkan seluruh *event* yang tersedia di dalam sistem. Pada halaman ini, pengguna dapat melihat daftar *event* beserta informasi dasarnya secara terstruktur. Bagi pengguna yang memiliki hak akses yang sesuai, tersedia fitur *manage ticket* yang memungkinkan pengelolaan jenis-jenis tiket untuk masing-masing *event*, seperti mengubah nama

tiket, harga, kuota, dan ketentuan lainnya. Selain itu, pengguna juga dapat menambahkan *event* baru ke dalam sistem melalui halaman ini, selama mereka memiliki izin untuk melakukannya. Dengan fitur-fitur tersebut, halaman ini menjadi pusat utama dalam pengelolaan *event*.



Gambar 3.43. *Customer List Page*

Halaman *customer list* merupakan halaman yang menampilkan seluruh *customer* yang telah terdaftar melalui sisi *front-end website ticketing*. Informasi setiap *customer* ditampilkan dalam bentuk daftar yang terstruktur, sehingga memudahkan proses pemantauan dan pengelolaan data pengguna. Melalui halaman ini, pengguna *backend* dengan hak akses yang sesuai juga dapat melakukan tindakan khusus seperti melakukan *blacklist* terhadap *customer* tertentu apabila diperlukan, misalnya karena pelanggaran ketentuan atau aktivitas mencurigakan. Fitur ini dirancang untuk menjaga keamanan sistem serta memastikan kualitas interaksi antara platform dan penggunanya tetap terjaga.

—

3.3.8 Tools dan Teknologi yang Digunakan

Selama proses pengembangan *website ticketing* di PT Mitra Integrasi Digital, mahasiswa magang menggunakan berbagai *tools* untuk mendukung efisiensi kerja dan memastikan kualitas hasil yang optimal. *Tools* yang digunakan antara lain:

1. **Visual Studio Code (VS Code)**
   Visual Studio Code adalah *editor* kode sumber yang ringan dan fleksibel, digunakan sebagai lingkungan utama untuk menulis dan mengelola kode pengembangan *website*. VS Code mendukung berbagai ekstensi seperti Git

integration, *auto-completion*, dan *debugging*, yang sangat membantu dalam proses pengembangan perangkat lunak.

2. **DBeaver**
   DBeaver merupakan aplikasi manajemen basis data yang digunakan untuk mengakses, memvisualisasikan, dan mengelola struktur *database* proyek. DBeaver menyediakan antarmuka grafis yang mempermudah eksekusi *query* SQL serta pemantauan relasi antar tabel, sehingga mempercepat proses *debugging* dan pengembangan berbasis data.

3. **Git**
   Git adalah sistem kontrol versi yang digunakan untuk melacak perubahan kode dan mengelola versi proyek secara efisien. Penggunaan Git memungkinkan mahasiswa magang melakukan *commit*, *push*, dan sinkronisasi perubahan kode secara teratur. Selain itu, Git juga membantu dalam dokumentasi proses pengembangan dan *rollback* jika terjadi kesalahan.

4. **WireGuard**
   WireGuard adalah protokol VPN yang digunakan untuk mengamankan koneksi ke jaringan internal perusahaan. Dalam proyek ini, WireGuard memungkinkan mahasiswa magang untuk mengakses *repository* Git yang berada di dalam jaringan privat perusahaan dengan koneksi yang terenkripsi dan aman.

5. **Nginx**
   Nginx adalah *web server* sekaligus *reverse proxy* yang digunakan dalam proses *deployment* aplikasi ke *server*. Dalam proyek ini, Nginx berfungsi sebagai *reverse proxy* yang mengarahkan permintaan dari pengguna ke aplikasi yang berjalan di *backend*. Penggunaan Nginx memungkinkan pengelolaan trafik yang efisien serta mendukung konfigurasi SSL dan *load balancing*, sehingga meningkatkan performa dan keamanan aplikasi yang di-*deploy*.

6. **PM2**
   PM2 (*Process Manager* 2) adalah manajer proses untuk aplikasi Node.js yang digunakan untuk menjalankan, memonitor, dan menjaga agar aplikasi tetap aktif secara otomatis di *server*. Dalam pengembangan *website ticketing* ini, PM2 digunakan untuk memastikan aplikasi *backend* tetap berjalan setelah

server restart serta mempermudah monitoring *log* dan manajemen proses aplikasi.

3.4 Pengujian

Pengujian terhadap sistem dilakukan secara internal untuk memastikan bahwa setiap fitur dapat berjalan sesuai dengan fungsinya. Pengujian ini dilakukan dalam dua tahap. Pertama, pengujian dilakukan secara mandiri oleh penulis dengan mencoba seluruh alur sistem secara menyeluruh, termasuk validasi data, tampilan antarmuka, dan proses *backend*. Kedua, pengujian dilakukan oleh dua *supervisor* dari perusahaan tempat magang, yaitu Bapak Boby Hartanto dan Bapak Boby Harmoko, untuk mengevaluasi sistem dari sudut pandang pengguna dan memberikan masukan terkait fungsionalitas maupun kenyamanan penggunaan.

Masukan dari kedua *supervisor* digunakan untuk melakukan beberapa perbaikan kecil, seperti penyesuaian tampilan dan pengoptimalan alur proses agar lebih efisien dan sesuai kebutuhan pengguna.

3.5 Kendala dan Solusi yang Ditemukan

3.5.1 Tantangan Adaptasi Teknologi

Kendala: Pada awal magang, mahasiswa magang menghadapi kesulitan dalam memahami dan beradaptasi dengan teknologi yang digunakan perusahaan, khususnya Vue.js dengan Nuxt.js dan Golang yang belum pernah digunakan sebelumnya.

Solusi: Melakukan pembelajaran mandiri melalui dokumentasi resmi, tutorial *online*, dan praktik langsung (*learning by doing*). *Supervisor* juga memberikan *guidance* dan *resources* yang membantu proses adaptasi dengan memberikan contoh-contoh implementasi yang relevan.

3.5.2 Integrasi Payment Gateway

Kendala: Integrasi dengan *payment gateway* Xendit memerlukan pemahaman yang mendalam tentang *webhook handling*, *signature validation*, dan *flow* pembayaran yang kompleks.

Solusi: Mempelajari dokumentasi Xendit secara mendetail, melakukan *testing* menggunakan *sandbox environment* untuk simulasi berbagai skenario

pembayaran, dan berkonsultasi dengan *supervisor* untuk *best practices* dalam implementasi *payment gateway* yang aman dan *reliable*.

3.5.3 Deployment dan Configuration

Kendala: Proses *deployment* dengan Nginx sebagai *reverse proxy* dan konfigurasi PM2 untuk *process management* memerlukan pemahaman tentang *server administration* yang sebelumnya belum dikuasai.

Solusi: Belajar fundamental Linux *server administration*, mempelajari konfigurasi Nginx untuk *routing* dan *load balancing*, dan memahami *process management* dengan PM2 melalui praktik langsung dan *guidance* dari tim *senior developer*.