LAMPIRAN

Lampiran 1. Surat Kesepakatan Pembimbing

SURAT KESEPAKATAN BIMBINGAN SKRIPSI

Kami yang bertanda tangan dibawah ini:

Pihak Pertama

Nama	: Muhammad Faqih Maulana
NIM	: 21090056
Program Studi	: Sarjana Terapan Teknik Informatika

Pihak Kedua

Nama	: Ir. Ginanjar Wiro Sasmito, S.Kom., M.Kom.
Status	: Dosen
NIDN	: 0613028601
Jabatan Fungsional	: Lektor Kepala
Pangkat/Golongan	: Penata Tk.I / III-D

Pada hari ini Kamis tanggal 13 Maret 2025 telah terjadi sebuah kesepakatan bahwa Pihak Kedua bersedia menjadi Pembimbing I Skripsi Pihak Pertama dengan syarat Pihak Pertama wajib melakukan bimbingan Skripsi minimal 8 kali kepada Pihak Kedua. Adapun waktu dan tempat pelaksanaan disepakati antar pihak.

Demikian kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Skripsi

Tegal, 13 Maret 2025

Pihak Pertama	Pihak Kedua
Muhammad Faqih Maulana	Ir. Ginanjar Wiro Sasmito, S.Kom., M.Kom.

Mengetahui
Ketua Program Studi Sarjana Terapan Teknik Informatika

Dyah Apriliani, S.T., M.Kom.
NIPY. 09.015.225

SURAT KESEPAKATAN BIMBINGAN SKRIPSI

Kami yang bertanda tangan dibawah ini:

Pihak Pertama

Nama	: Muhammad Faqih Maulana
NIM	: 21090056
Program Studi	: Sarjana Terapan Teknik Informatika

Pihak Kedua

Nama	: M. Nishom, S.Kom., M.Kom.
Status	: Dosen
NIDN	: 0619048701
Jabatan Fungsional	: Lektor
Pangkat/Golongan	: Penata/ III/C

Pada hari ini Kamis tanggal 13 Maret 2025 telah terjadi sebuah kesepakatan bahwa Pihak Kedua bersedia menjadi Pembimbing II Skripsi Pihak Pertama dengan syarat Pihak melakukan bimbingan satu kali dalam seminggu atau setidak tidaknya tiga bimbingan dalam
satu bulan (dengan progress), apabila saya tidak memenuhi syarat tersebut maka saya tidak berhak meminta surat rekomendasi mengikuti sidang skripsi dan saya juga berjanji memenuhi persyaratan tersebut dan menyelesaikan penelitian berupa produk dan laporan sesuai jadwal penelitian (tepat waktu).
Demikian kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Skripsi

Tegal, 13 Maret 2025

Pihak Pertama	Pihak Kedua
Muhammad Faqih Maulana	M. Nishom, S.Kom., M.Kom.

Mengetahui
Ketua Program Studi Sarjana Terapan Teknik Informatika

Dyah Apriliani, S.T., M.Kom.
NIPY. 09.015.225

Lampiran 2. Surat Pernyataan Pengajuan HKI

SURAT PERNYATAAN

Yang bertanda tangan di bawah ini, pemegang hak cipta:

1. Nama : Muhammad Faqih Maulana
   Kewarganegaraan : Indonesia
   Alamat : Jl. Mbah Bregas Bo.40 RT.04/RW.04 Desa Sidakaton ,Kecamatan Dukuhturi, Kabupaten Tegal,Provinsi Jawa Tengah, 52192

2. Nama : Ir. Ginanjar Wiro Sasmito, S.Kom., M.Kom.
   Kewarganegaraan : Indonesia
   Alamat : Jln. Raya Kluwut Timur No. 24, Kecamatan Bulakamba, Kabupaten Brebes.

3. Nama : M. Nishom, S.Kom., M.Kom.
   Kewarganegaraan : Indonesia
   Alamat : Jl. Jepara, Perum Griya Putri Land Blok A6, RT 03 RW 04, Margadana, Tegal, 52143

Dengan ini menyatakan bahwa:

1. Karya Cipta yang saya mohonkan:
   Berupa : Program Komputer
   Berjudul : Rancang Bangun Aplikasi Booking Booth Tenant pada Cresindo Event Organizer Menggunakan Algoritma Greedy
   - Tidak meniru dan tidak sama secara esensial dengan Karya Cipta milik pihak lain atau obyek kekayaan intelektual lainnya sebagaimana dimaksud dalam Pasal 68 ayat (2);
   - Bukan merupakan Ekspresi Budaya Tradisional sebagaimana dimaksud dalam Pasal 38;
   - Bukan merupakan Ciptaan yang tidak diketahui penciptanya sebagaimana dimaksud dalam Pasal 39;
   - Bukan merupakan hasil karya yang tidak dilindungi Hak Cipta sebagaimana dimaksud dalam Pasal 41 dan 42;
   - Bukan merupakan Ciptaan seni lukis yang berupa logo atau tanda pembeda yang digunakan sebagai merek dalam perdagangan barang/jasa atau digunakan sebagai lambang organisasi, badan usaha, atau badan hukum sebagaimana dimaksud dalam Pasal 65 dan;
   - Bukan merupakan Ciptaan yang melanggar norma agama, norma susila, ketertiban umum, pertahanan dan keamanan negara atau melanggar peraturan perundang-undangan sebagaimana dimaksud dalam Pasal 74 ayat (1) huruf d Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

2. Sebagai pemohon mempunyai kewajiban untuk menyimpan asli contoh ciptaan yang dimohonkan dan harus memberikan apabila dibutuhkan untuk kepentingan penyelesaian sengketa perdata maupun pidana sesuai dengan ketentuan perundang-undangan.

3. Karya Cipta yang saya mohonkan pada Angka 1 tersebut di atas tidak pernah dan tidak sedang dalam sengketa pidana dan/atau perdata di Pengadilan.

4. Dalam hal ketentuan sebagaimana dimaksud dalam Angka 1 dan Angka 3 tersebut di atas saya / kami langgar, maka saya / kami bersedia secara sukarela bahwa:
   a. permohonan karya cipta yang saya ajukan dianggap ditarik kembali; atau
   b. Karya Cipta yang telah terdaftar dalam Daftar Umum Ciptaan Direktorat Hak Cipta, Direktorat Jenderal Hak Kekayaan Intelektual, Kementerian Hukum Dan Hak Asasi Manusia R.I dihapuskan sesuai dengan ketentuan perundang-undangan yang berlaku.
   c. Dalam hal kepemilikan Hak Cipta yang dimohonkan secara elektronik sedang dalam berperkara dan/atau sedang dalam gugatan di Pengadilan maka status kepemilikan surat pencatatan elektronik tersebut ditangguhkan menunggu putusan Pengadilan yang berkekuatan hukum tetap.

Demikian Surat pernyataan ini saya/kami buat dengan sebenarnya dan untuk dipergunakan sebagimana mestinya.

Tegal, Juni 2025

(Muhammad Faqih Maulana)
Pemegang Hak Cipta *

(Ir. Ginanjar Wiro Sasmito, S.Kom., M.Kom.)
Pemegang Hak Cipta *

(M. Nishom, S.Kom., M.Kom.)
Pemegang Hak Cipta *

Lampiran 3. Surat Pengalihan Hak Cipta

SURAT PENGALIHAN HAK CIPTA

Yang bertanda tangan di bawah ini :

1. Nama : Muhammad Faqih Maulana
   Kewarganegaraan : Indonesia
   Alamat : Jl. Mbah Bregas Bo.40 RT.04/RW.04 Desa Sidakaton ,Kecamatan Dukuhturi, Kabupaten Tegal,Provinsi Jawa Tengah, 52192

2. Nama : Ir. Ginanjar Wiro Sasmito, S.Kom., M.Kom.
   Kewarganegaraan : Indonesia
   Alamat : Jln. Raya Kluwut Timur No. 24, Kecamatan Bulakamba, Kabupaten Brebes.

3. Nama : M. Nishom, S.Kom., M.Kom.
   Kewarganegaraan : Indonesia
   Alamat : Jl. Jepara, Perum Griya Putri Land Blok A6, RT 03 RW 04, Margadana, Tegal, 52143

Adalah **Pihak I** selaku pencipta, dengan ini menyerahkan karya ciptaan saya kepada :

Nama : Pusat Penelitian dan Pengabdian Masyarakat (P3M)
Alamat : Jl. Mataram No.9, Pesurungan Lor, Kec. Margadana, Kota Tegal, Provinsi Jawa Tengah, 52147

Adalah **Pihak II** selaku Pemegang Hak Cipta berupa Program Komputer dengan judul "Rancang Bangun Aplikasi Booking Booth Tenant pada Cresindo Event Organizer menggunakan Algoritma Greedy ". untuk didaftarkan di Direktorat Hak Cipta dan Desain Industri, Direktorat Jenderal Kekayaan Intelektual, Kementerian Hukum dan Hak Asasi Manusia Republik Indonesia.

Demikianlah surat pengalihan hak ini kami buat, agar dapat dipergunakan sebagaimana mestinya.

Tegal, Juni 2025

Pemegang Hak Cipta
Kepala P3M

(Muhammad Fikri Hidayattulah, S.T., M.Kom.)

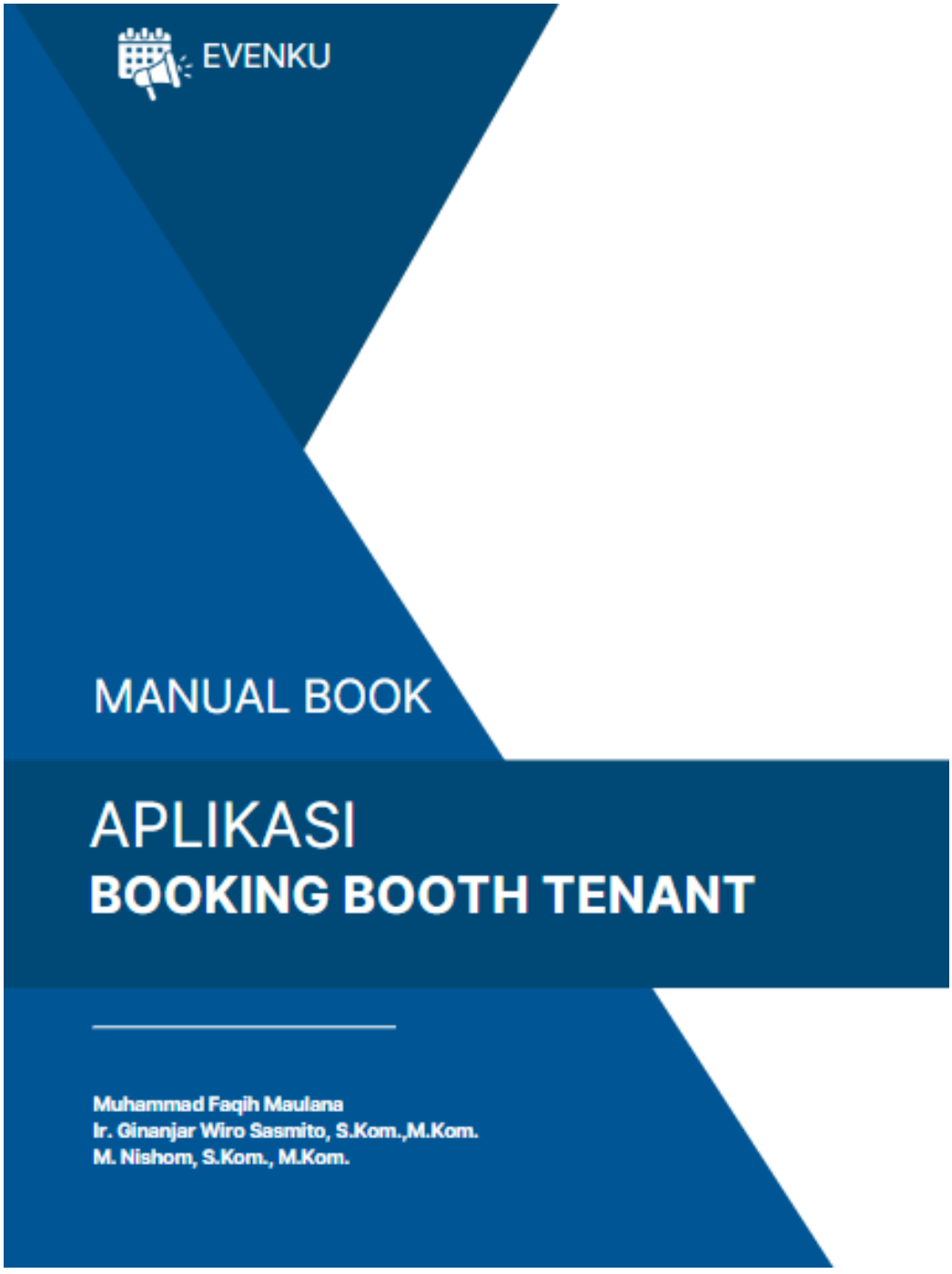
Pencipta

(Muhammad Faqih Maulana)

(Ir. Ginanjar Wiro Sasmito, S.Kom., M.Kom)

(M. Nishom, S.Kom., M.Kom.)

Lampiran 4. Manual Book



Pendahuluan

Eventku Adalah Aplikasi *Booking Booth Tenant* Dengan Algoritma *Greedy* yang dirancang khusus untuk memfasilitasi proses penyewaan booth bagi para tenant (pedagang atau pemilik usaha) dalam suatu *event*, seperti bazar, pameran, atau festival.

Tujuan Utama Dari Pembuatan Aplikasi ini Adalah untuk :

1. Memudahkan tenant dalam memesan *booth* secara online tanpa harus datang langsung atau melalui proses manual.
2. Menggunakan algoritma *greedy* untuk memilih *booth* terbaik secara cepat berdasarkan preferensi *tenant* seperti lokasi, ukuran, dan jenis *booth*.
3. Memastikan seluruh *booth* terisi secara efisien, meminimalkan ruang kosong, dan menyeimbangkan penempatan *tenant* di area strategis.
4. Memberikan pengalaman pemesanan yang cepat, transparan, dan nyaman, sehingga tenant merasa puas dan lebih loyal terhadap penyelenggara *event*.
5. Memungkinkan admin *event* untuk memantau proses *booking* secara *real-time*, mengatur layout *booth*, serta melihat data pemesanan secara sistematis.
6. Tenant dapat langsung melakukan pembayaran *booth* secara online melalui aplikasi menggunakan berbagai metode transfer bank sehingga proses menjadi lebih praktis, aman, dan *real-time*.

Dengan antarmuka yang interaktif dan ramah pengguna, Eventku memudahkan proses pemesanan booth bagi *tenant*, sekaligus membantu penyelenggara *event* dalam mengelola data penyewaan secara efisien.

Penerapan algoritma *Greedy* memungkinkan sistem secara cerdas memilih *booth* terbaik sesuai preferensi *tenant* secara otomatis dan cepat. Selain itu, fitur pembayaran langsung dalam aplikasi membuat seluruh proses dari pemilihan hingga pembayaran menjadi lebih praktis.

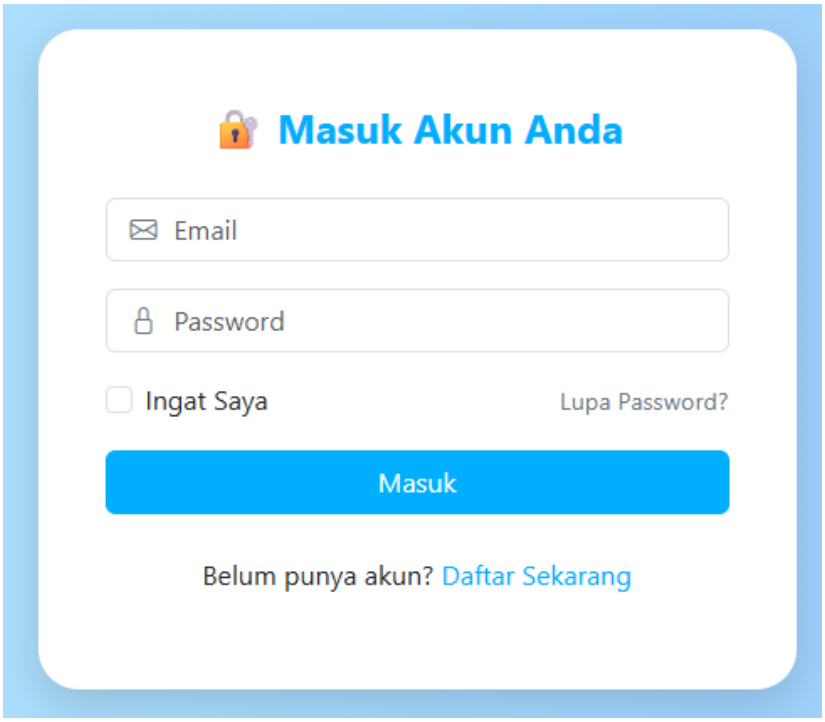
Penggunaan Aplikasi EventKu

1. Admin
    a. *Login*
    b. Mengelola Banner dengan Menentukan Judul, Tanggal, Serta Lokasi Gambar *Banner*, Serta Galeri.
    c. Mengelola *Event* Dengan Menentukan Acara berupa judul acara, Tanggal Acara, Tanggal Selesai,Deskripsi Acara Dan Gambar.
    d. *Categori Booth* dengan menentuka tipe *booth*, Harga, fasilitas.
    e. *Maps Booth* mengelola layout denah dengan menentukan *section*, dan mengatur lokasi *booth* yang ingin di tentukan.
    f. *Manage User* untuk Mengelola Data Akun *Tenant* dan Admin.
    g. *Inbox* untuk *Chat* Antara Tenant dengan Admin.
    h. *Report* Penjualan untuk melihat Detail *Booking* Serta Transaksi.

2. Tenant
    a. Login
    b. Melihat Informasi Tentang Event dan Informasi Booth Yang tersedia.
    c. Melakukan Booking Dengan Mengisi Formulir tentang data Perusahaan.
    d. Melakukan Metode Pembayaran.
    e. Melihat Pesanan Yang Telah di Booking serta Mencetak Ke dalam Pdf.
    f. Melakukan Chat dengan Admin Apabila ingin konfirmasi lebih lanjut.

Tata Cara Pengoprasian Aplikasi EventKu

1. **Login Admin**



Gambar 1. Login Admin

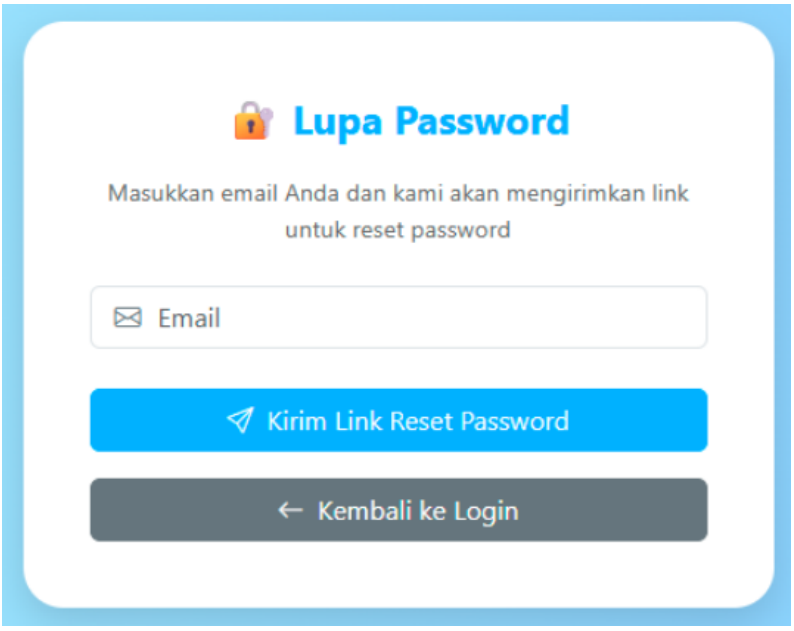
Setelah membuka aplikasi di browser, klik tombol Login. Jika login berhasil, Anda akan langsung diarahkan ke halaman Dashboard.

Di halaman Dashboard, tersedia menu Sidebar di sebelah kiri yang berisi:

- Dashboard: Menampilkan ringkasan data utama.
- Manage Banner: Mengelola banner halaman utama.
- Manage Event: Membuat dan mengatur acara.
- Kategori Booth: Mengelompokkan jenis booth yang tersedia.
- Maps Booth: Menampilkan denah lokasi booth.
- Manage User: Mengelola akun pengguna.
- Inbox: Melihat pesan atau pertanyaan dari tenant.
- Report: Menampilkan laporan dan statistik event.
- Logout: Keluar dari akun Anda.

Menu sidebar ini memudahkan admin untuk mengelola seluruh data dan aktivitas dalam platform.

Forgot-Password Admin



Gambar 2. Forgot-Password Admin

Klik tombol "Kirim Link Reset Password" untuk mengirimkan tautan reset ke email Anda. Di halaman ini, Anda dapat melakukan pemulihan akses akun apabila lupa kata sandi.

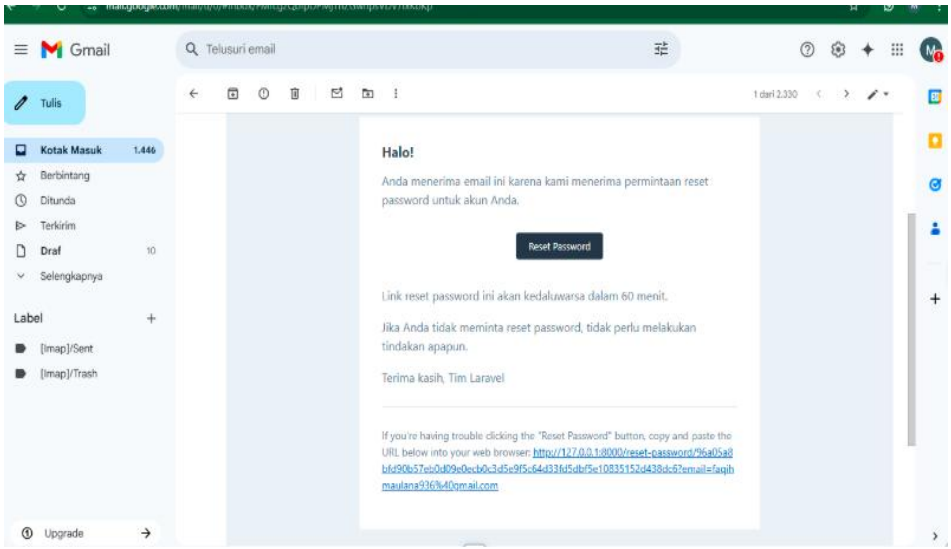
Cukup masukkan alamat email yang terdaftar, kemudian sistem akan mengirimkan link untuk mengganti password melalui email.

Fitur ini berguna untuk:

1. Mengamankan akun jika Anda lupa kata sandi,
2. Mendapatkan akses kembali dengan cepat,
3. Melakukan reset password tanpa harus menghubungi admin.

Jika Anda ingin kembali ke halaman login, klik tombol "Kembali ke Login" di bawahnya.

Email Reset Password



Gambar 3. Email Reset Password

Email ini dikirim karena pengguna melakukan permintaan untuk me-reset password akun mereka.

Tombol “Reset Password”:

Klik tombol ini untuk menuju halaman pengaturan ulang password. Link hanya berlaku selama 6 menit.

Catatan Tambahan:

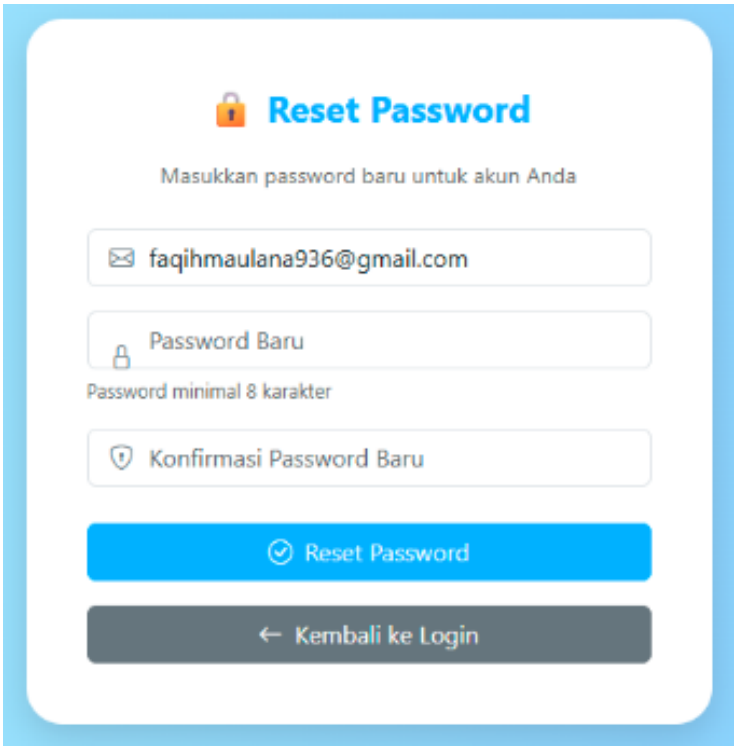
Jika Anda tidak merasa meminta reset password, abaikan email ini. Tidak ada tindakan lanjutan yang perlu dilakukan.

Alternatif Link:

- Jika tombol tidak dapat diklik, tersedia juga tautan langsung yang bisa disalin ke browser.

Email ini merupakan bagian dari sistem keamanan untuk membantu pengguna memulihkan akses ke akun mereka secara mandiri.

Reset-Password Admin

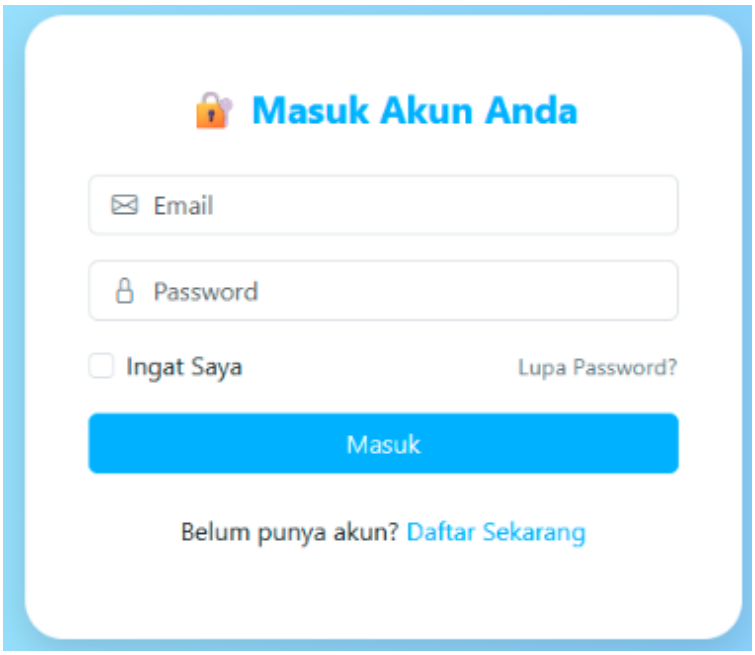


Gambar 4. Reset Password Admin

Halaman ini muncul setelah Anda mengklik link reset password dari email. Di sini, Anda dapat memasukkan password baru dan konfirmasi password untuk mengganti kata sandi akun Anda. Setelah diisi, klik tombol "Reset Password" untuk menyimpan perubahan.

Jika ingin membatalkan, Anda bisa klik tombol "Kembali ke Login".

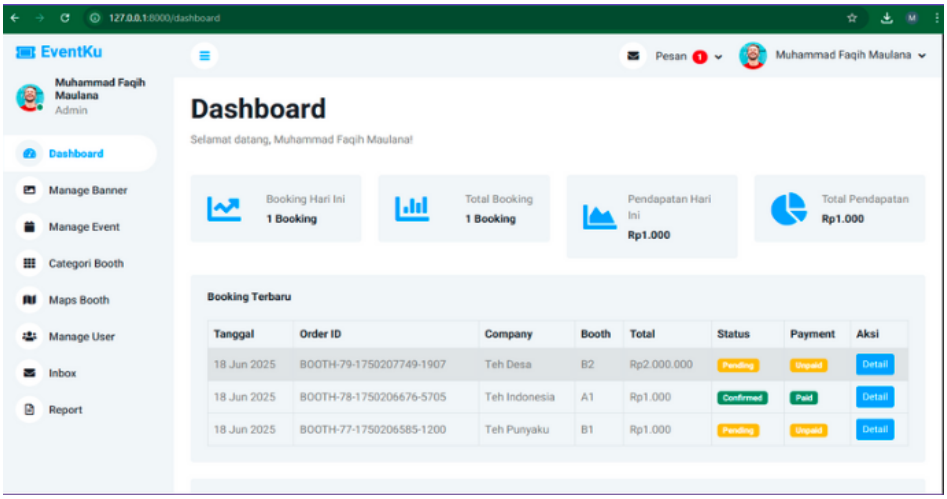
Login Admin Kembali



Gambar 5. Login Admin

Setelah Anda mengklik tombol "Reset Password", sistem akan menyimpan password baru Anda dan secara otomatis mengarahkan Anda ke halaman login. Di halaman login, Anda dapat langsung masuk kembali menggunakan Email Anda dan Password baru yang baru saja Anda atur. Proses ini memastikan akun Anda dapat diakses kembali dengan aman menggunakan kata sandi yang baru.

2. Dashboard



Gambar 6. Dashboard

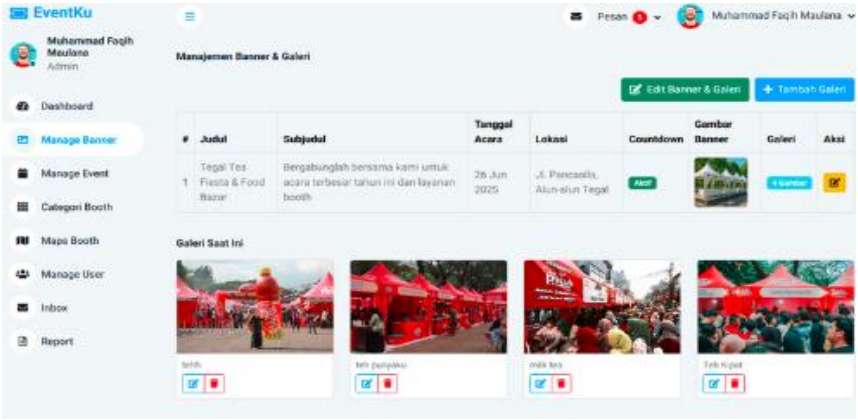
Gambar tersebut menampilkan Dashboard Admin dari aplikasi EventKu, tempat admin dapat memantau aktivitas booking booth secara menyeluruh. Di bagian atas, terdapat ringkasan statistik berupa:

1. Booking Hari Ini: Menampilkan jumlah pemesanan booth yang dilakukan pada hari ini.
2. Total Booking: Menunjukkan total seluruh pemesanan booth yang telah dilakukan.
3. Pendapatan Hari Ini dan Total Pendapatan: Menginformasikan pemasukan dari booking booth, baik untuk hari ini maupun secara keseluruhan.

Di bagian bawah terdapat tabel Booking daftar pesanan secara detail, seperti:

1. Tanggal Pemesanan.
2. Order ID.
3. Nama Perusahaan/Brand.
4. Nomor Booth.
5. Total Harga.
6. Status Pemesanan (Pending/Confirmed).
7. Status Pembayaran (Unpaid/Paid).
8. Aksi berupa tombol Detail untuk melihat informasi pemesanan lebih lengkap.

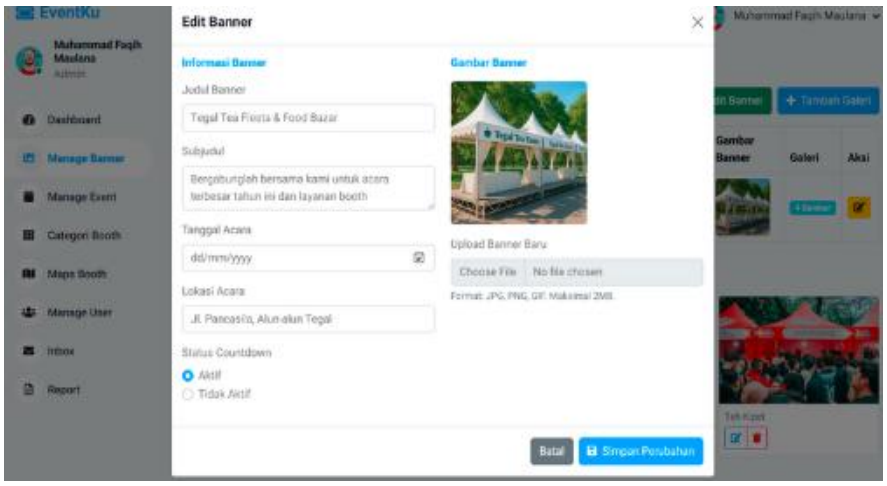
3. Manage Banner



Gambar 7. Manage Banner

pada Tampilan Manage Banner brfungsi untuk mengedit konten banner tampilan antar muka di halaman utama berupa judul acara, tanggal acara, serta gambar dengan countdown hitung mundur, serta menambahkan galeri dengan deskripsi.

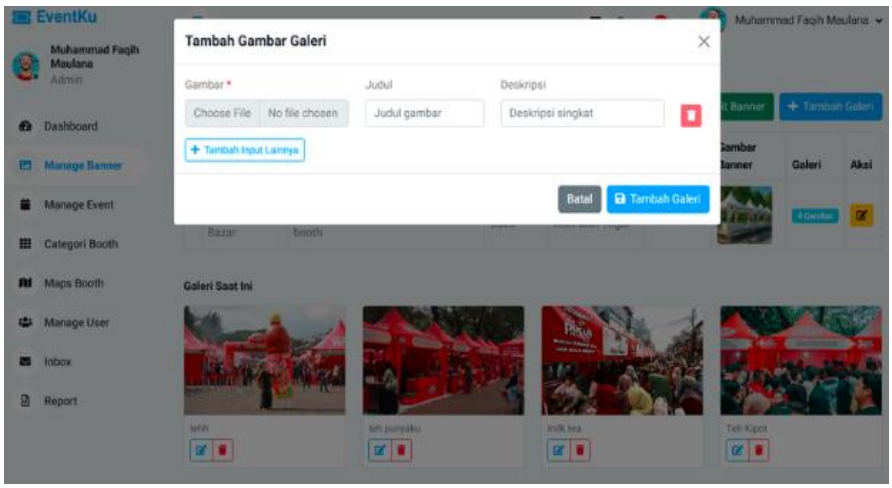
Edit Banner



Gambar 8. Edit Banner

Untuk Edit Banner Sendiri Berfungsi untuk mengubah informasi mulai dari judul banner, subjudul Tanggal acara, Lokasi Acara Status Countdown, Gambar Banner, yang bertujuan untuk memudahkan admin dalam mengedit sesuai jadwal acara yang ingin di tentukan, selanjutnya **simpan perubahan** dan banner akan tampil di halaman utama.

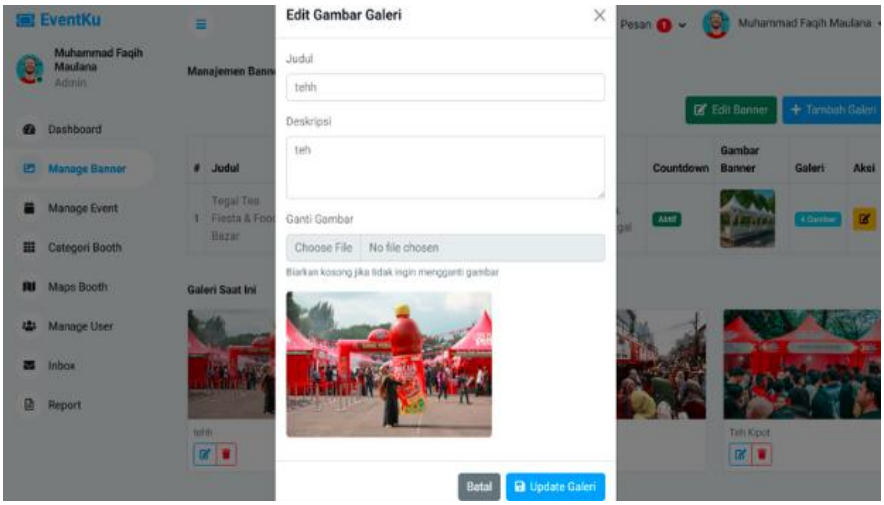
Tambah Galeri



Gambar 9. Tambah Galeri

Admin dapat Menambah atau menghapus Galeri Dengan gambar yang ingin di tampilkan di halaman utama untuk menampilkan histori tentang event yang sebelumnya di adakan. dengan menambahkan judul dan deskripsi yang di tentukan lalu klik **tambah galeri**, maka otomatis akan kesimpan dan muncul di halaman utama.

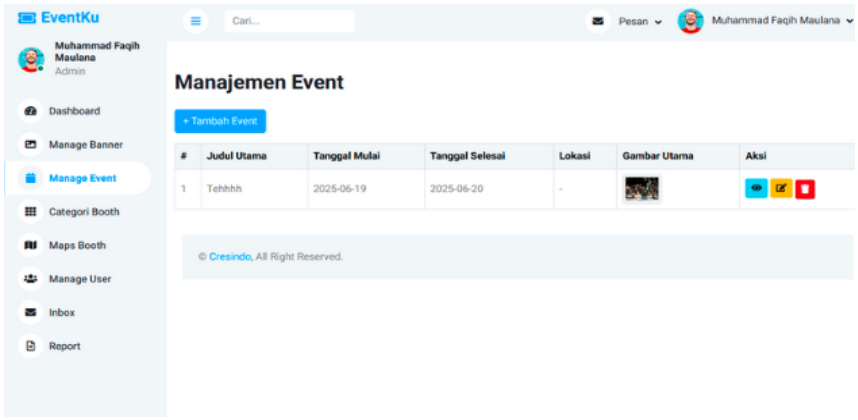
Edit Galeri



Gambar 10. Edit Galeri

Admin dapat Mengedit Galeri Dengan gambar yang ingin di rubah apabila ada kesalahan dalam menambah gambar atau deskripsi sebelumnya untuk mempermudah merubah kesalahan sebelumnya baik di dalam judul, deskripsi atau gambar yang sebelumnya di tambahkan. setelah melakukan perubahan maka klik button **update galeri**, maka otomatis akan update.

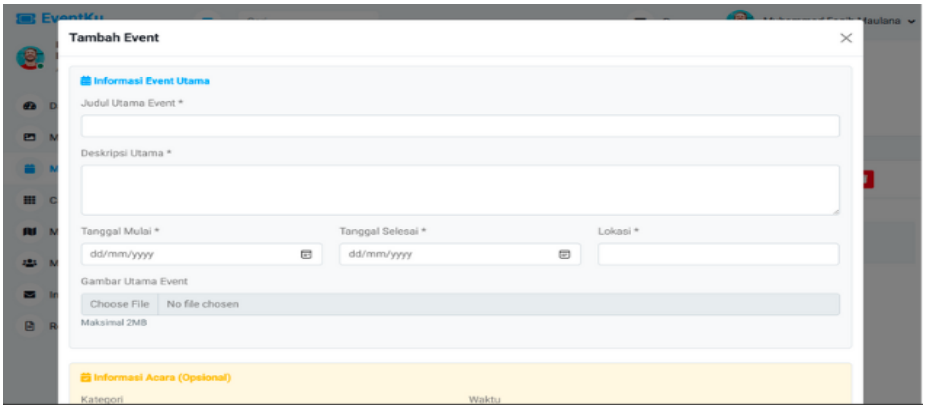
4. Manage Event



Gambar 11. Manage Event

Fitur Manage Event Bertujuan untuk Menambah Sebuah Acara di dalam suatu event yang di selenggarakan mulai dari judul acara, Tanggal Mulai dan tanggal Selesai Serta gambar yang di tentukan,waktu acara, deskripsi, Narasumber, posisi (gitaris).

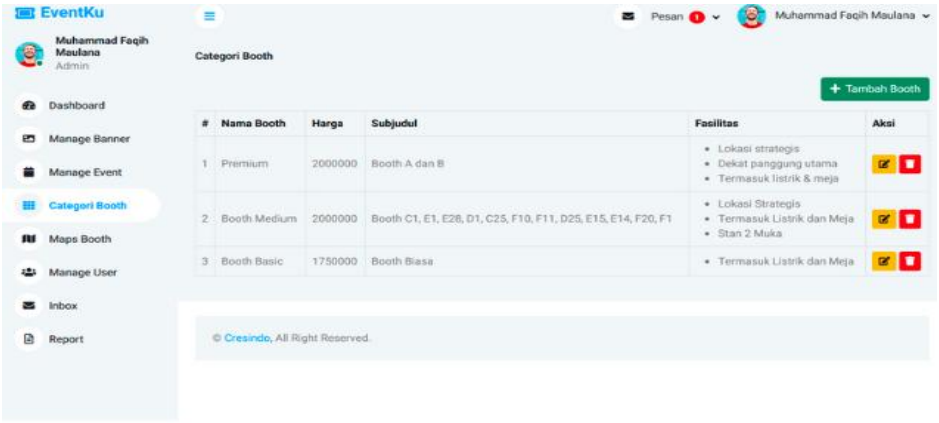
Tambah Event



Gambar 12. Tambah Event

Admin dapat Menambah, Mengedit Event Sesuai yang di inginkan berdasarkan judul dan deskripsi serta tanggal acara lalu klik **simpan**. setelah simpan lihat halaman utama dan cek apakah informasinya sudah sesuai.

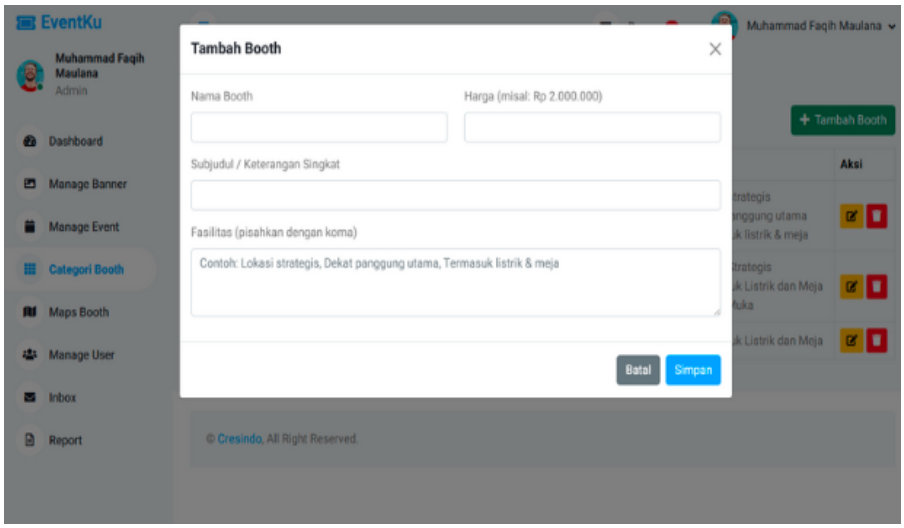
5. Categori Booth



Gambar 13. Categori Booth

Pada Menu Categori Booth Bertujuan untuk Menambahkan informasi terkait tentang Nama Booth, Harga, Subjudul, Fasilitas yang berfungsi untuk memberikan informasi terkait dengan booth, harga booth, serta fasilitas apa saja yang di dapat.

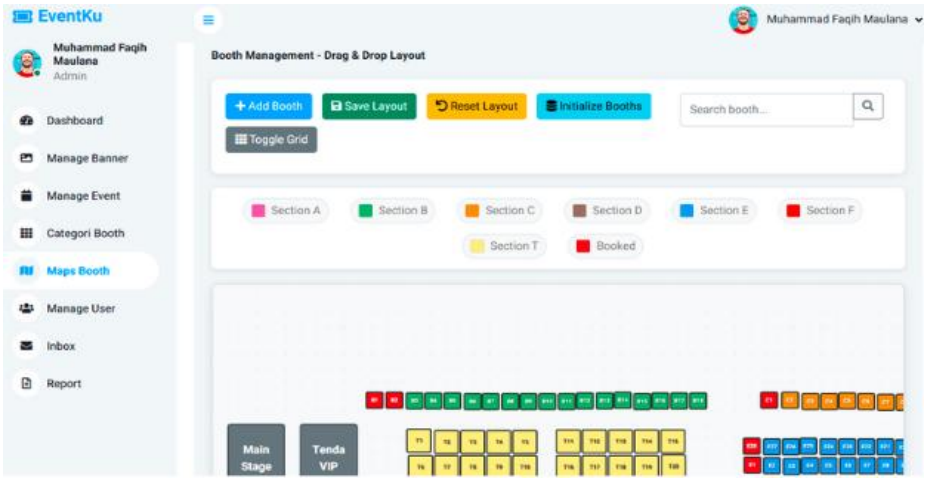
Tambah Categori



Gambar 14. Tambah Kategori

Pada Menu Categori Booth apabila admin ingin menambahkan informasi terkait harga booth, Fasilitas, serta deskripsi lainnya admin dapat mengklik button t**ambah booth** serta isi informasi yang ingin di tampilkan di halaman utama. setelah sudah menginputkan terkait informasi lalu klik **simpan**. maka tampilan di halaman utama akan muncul.

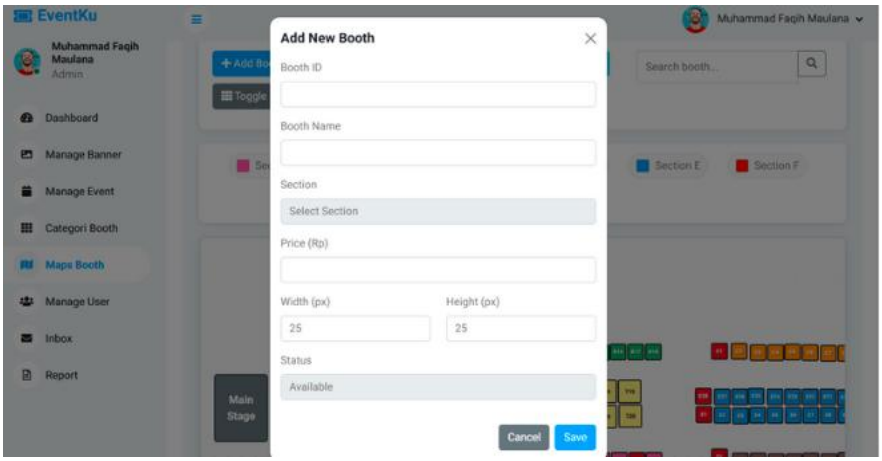
6. Maps Booth



Gambar 15. Maps Booth

Pada Tampilan di atas adalah menu maps booth dengan Fitur Drag & Drop Layout yang berguna untuk menambah booth(stan) dengan menentukan lokasi yang ingin di tentukan, serta Memberikan Informasi Terkait Harga, Nama Booth, Booth ID dengan Section yang di tentukan.

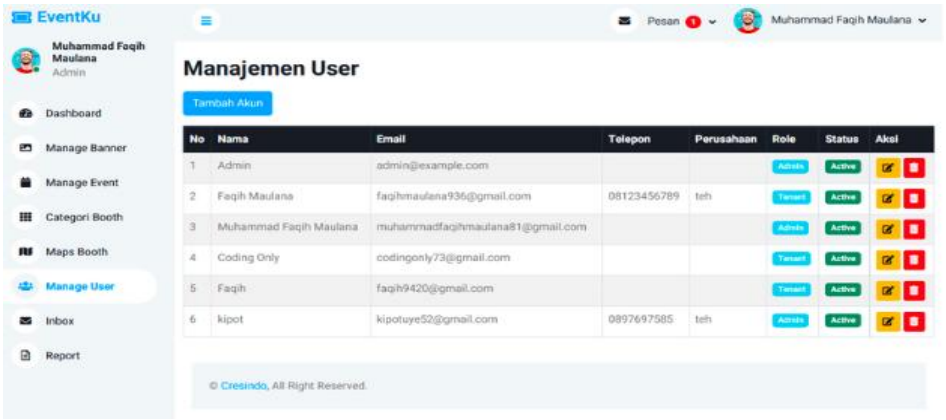
Tambah Booth



Gambar 16. Tambah Booth

Pada Menu Maps Booth admin dapat Menambahkan Booth dengan Kriteria tertentu yang di inginkan dengan ketentuan Booth ID A1, Nama Booth (contoh : Makanan), Pilih Section Sesuai Pada Menu Tersebut, Harga, Serta ukuran yang ingin di tentukan, dan Status Berupa (Available) apabila tersedia, (Booked) Apabila Sudah TerBooking, (Maintenance) Apabila Booth/Stan Dalam masa perbaikan. Setelah Semuanya Sudah Di isi, Maka Klik **Save** lalu geser sesuai lokasi yang di tentukan. apabila ada kesalahan atau ingin update terkait harga Klik dua kali Booth dengan ID yang di tentukan Sebelumnya, Lalu Update serta Klik Button **Save** lagi.

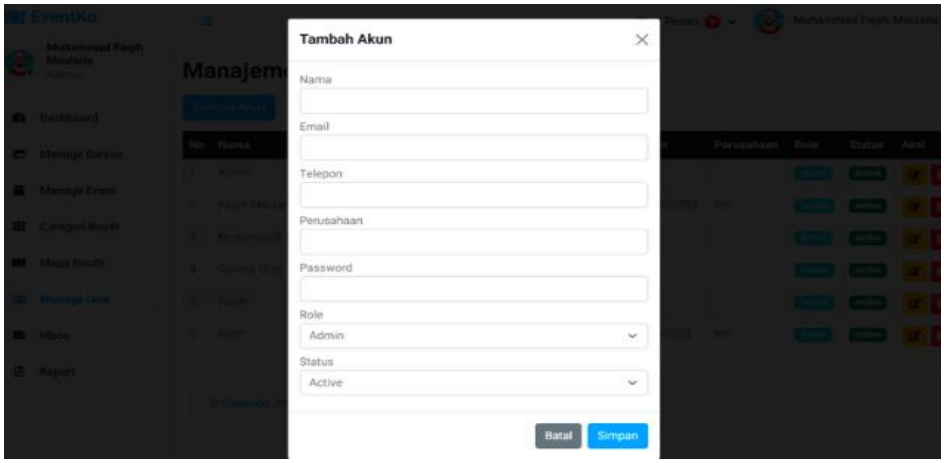
7. Manage User



Gambar 17. Manage User

Menu Manage User Berguna untuk Mengecek Akun yang terdaftar Baik Admin Atau Tenant untuk mempermudah dalam pengelolaan akun apabila ingin hapus akun yang tidak terpakai serta edit, apabila ingin mengedit nomor telepon ataupun ingin mengganti email.

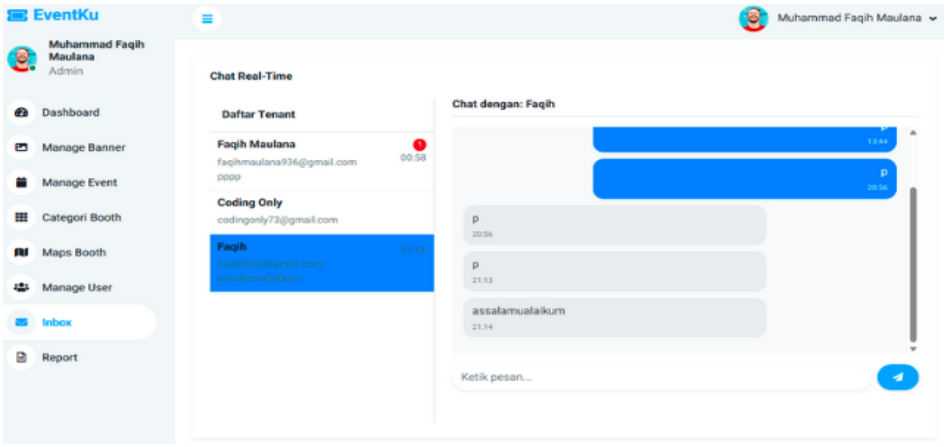
Tambah Akun



Gambar 18. Tambah Akun

Pada Menu Manage User Admin dapat Menambahkan Akun lewat menu tersebut dengan cara klik button **Tambah Akun,** Lalu Tambahkan Akun Berdasarkan Nama Email, Telepon, Perusahaan, Password dan role serta status yang di tentukan, Lalu Klik Button **Simpan.** Maka Otomastis Akun Sudah Berfungsi dan Bisa login untuk mengakses aplikasi dengan role yang di tentukan.

8. Inbox



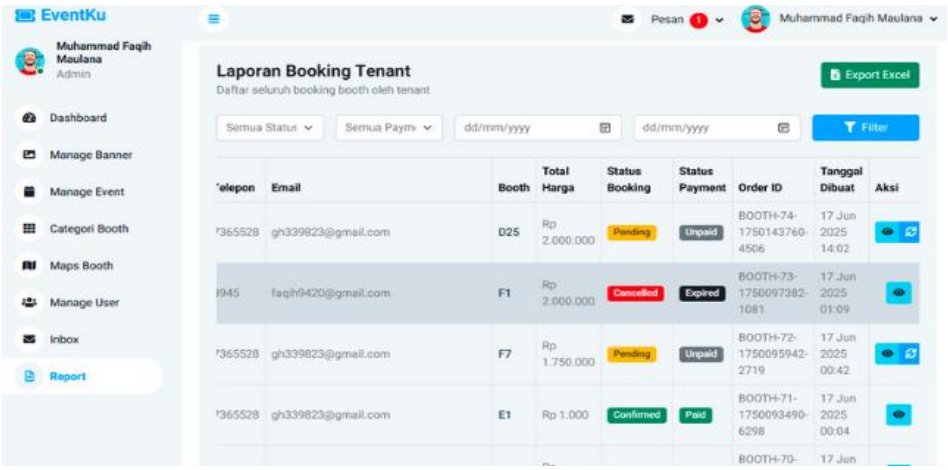
Gambar 19. Inbox

Gambar tersebut menunjukkan fitur chat real-time antara admin dan tenant dalam sistem aplikasi EventKu. Fitur ini memungkinkan admin berkomunikasi langsung dengan tenant secara instan.

1. Di bagian kiri, terdapat daftar tenant yang pernah mengirim pesan ke admin, lengkap dengan nama, email, isi pesan terakhir, dan waktu pengiriman.
2. Tanda notifikasi merah menandakan pesan baru yang belum dibaca admin.
3. Di bagian kanan, terlihat jendela percakapan aktif antara admin dan tenant bernama Faqih.
4. Pesan berwarna biru dikirim oleh admin, sementara warna abu-abu dikirim oleh tenant.
5. Di bagian bawah terdapat kolom input pesan dan tombol kirim berbentuk ikon pesawat kertas.

Fitur ini membantu admin menangani pertanyaan, permintaan, atau diskusi dari tenant dengan cepat dan efisien secara real-time.

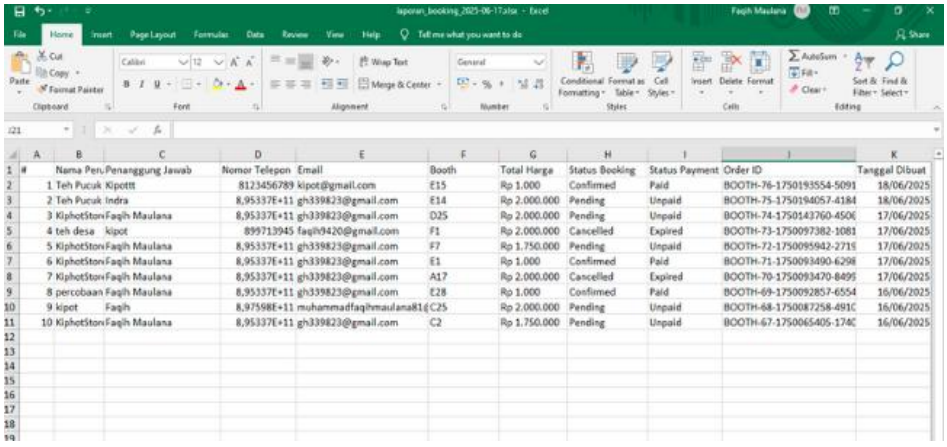
9. Report



Gambar 20. Report

Admin dapat Melihat Laporan Booking Tenant, Berupa Nama Perusahaan, Nomor Telepon, Email, Booth, Total Harga, Status Booking, Status Payment. Serta Dapat Melihat data dengan filtering berdasarkan status, Payment, serta filtering tanggal berdaradkan tanggal yang ingin di cari (contoh : Tanggal 12 agustus 2025 - 15 agustus 2025) dengan klik **filter**. Serta Admin dapat Meng export ke dalam exel.

Export Exel



Gambar 21. Export Exel

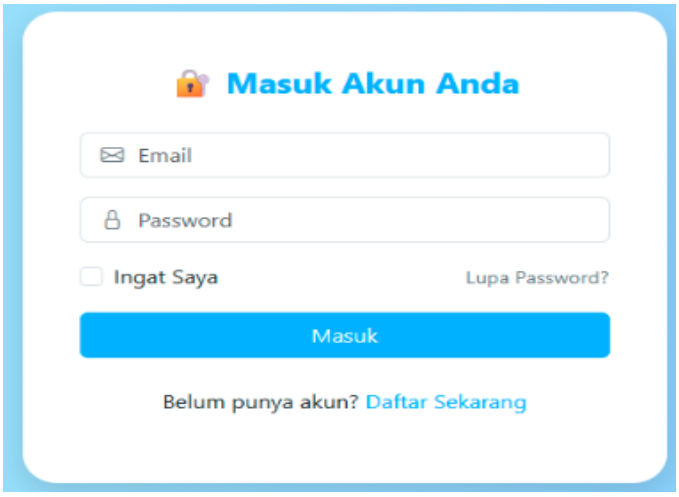
Admin dapat Melihat Laporan yang sudah di export seperti contoh di atas untuk mempermudah dalam mendata booking atau membackup ke exel untuk lebih mudah dalam melihat data per event.

10. Tenant Tampilan Utama

Buka Aplikasi Lalu Muncul Halaman Awal Tenant dengan Tampilan Judul Event, Waktu Event dan lokasi Event, Serta Hitung mundur acara.

Gambar 22. Tenant Tampilan Utama

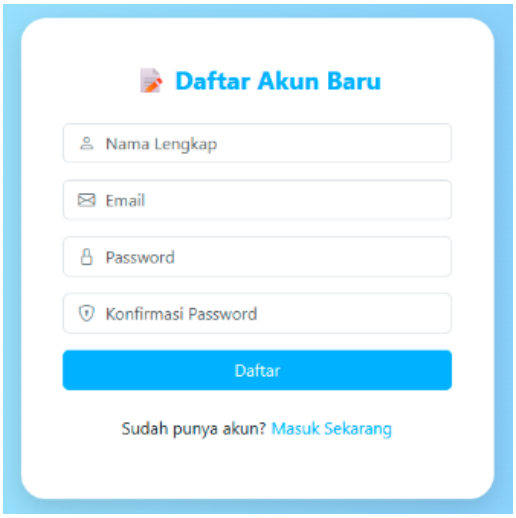
11. Login Tenant



Gambar 23. Login Tenant

Apabila ingin melakukan booking, diharapkan untuk login terlebih dahulu dengan cara memasukkan email dan password pada form yang tersedi, .Pastikan data yang dimasukkan benar, lalu klik tombol "Masuk" untuk melanjutkan ke proses booking. Jika belum memiliki akun, silakan klik "Daftar Sekarang" terlebih dahulu.

Register Tenant

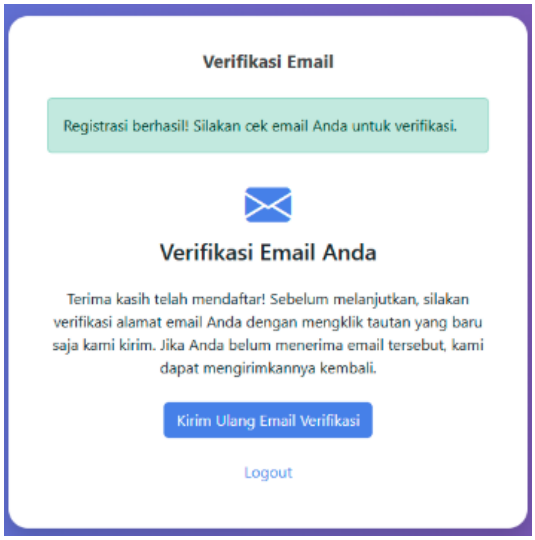


Gambar 24. Register Tenant

Untuk membuat akun, pengguna diminta mengisi nama lengkap, email, password, dan konfirmasi password. Setelah semua data diisi, klik tombol "Daftar" untuk menyelesaikan proses pendaftaran.

Setelah itu, sistem akan mengirimkan email verifikasi ke alamat yang didaftarkan. Silakan buka email Anda dan klik link verifikasi untuk mengaktifkan akun.

Verifikasi Email

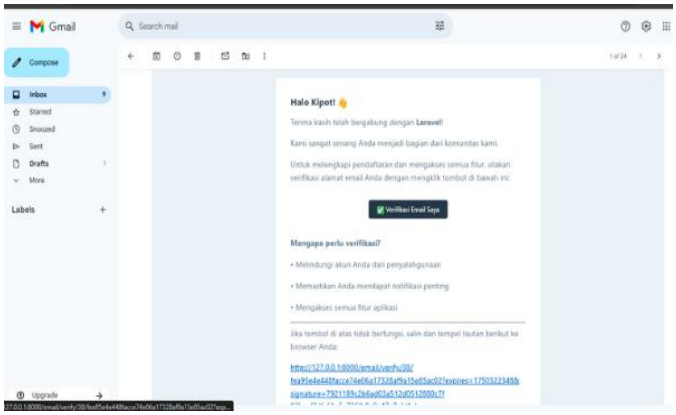


Gambar 25. Verifikasi Email

Pengguna diminta untuk memeriksa email dan mengklik tautan verifikasi yang telah dikirim untuk mengaktifkan akun.

Jika belum menerima email, tersedia tombol "Kirim Ulang Email Verifikasi" untuk mengirim ulang tautan tersebut.

Notifikasi Verifikasi Email



Gambar 26. Notifikasi Verifikasi Email

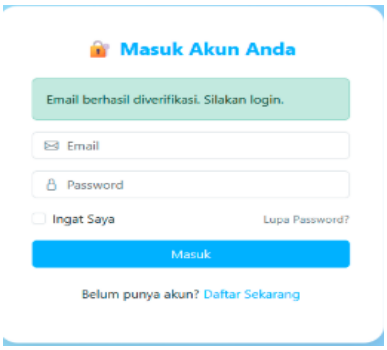
Klik tombol tersebut untuk mengaktifkan akun Anda. Link verifikasi hanya berlaku dalam waktu tertentu, jadi pastikan segera melakukan verifikasi.

Catatan Tambahan:

1. Jika Anda tidak merasa mendaftar akun, abaikan email ini. Tidak ada tindakan lanjutan yang perlu dilakukan.
2. Alternatif Link:
3. Jika tombol verifikasi tidak dapat diklik, tersedia juga tautan langsung yang bisa Anda salin dan tempel ke browser.

Email ini merupakan bagian dari sistem keamanan untuk memastikan bahwa alamat email yang digunakan memang milik Anda dan menjaga keaslian data pengguna.

Login Setelah Verifikasi

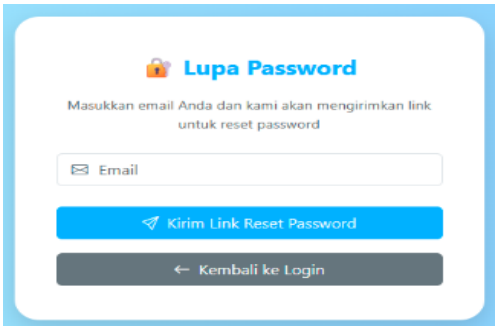


Gambar 27. Login Setelah Verifikasi

Terdapat pesan “Email berhasil diverifikasi. Silakan login”.

Pengguna dapat langsung masuk dengan mengisi email dan password, lalu klik tombol "Masuk" untuk mengakses aplikasi.

12. Lupa Password Tenant



Gambar 28. Lupa Passwword Tenant

Klik tombol "Kirim Link Reset Password" untuk mengirimkan tautan reset ke email Anda. Di halaman ini, Anda dapat melakukan pemulihan akses akun apabila lupa kata sandi.

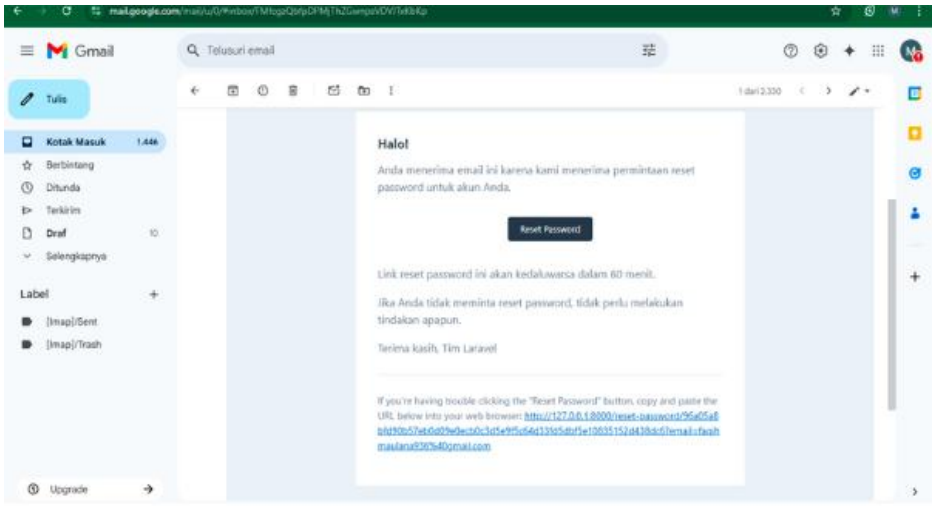
Cukup masukkan alamat email yang terdaftar, kemudian sistem akan mengirimkan link untuk mengganti password melalui email.

Fitur ini berguna untuk:

1. Mengamankan akun jika Anda lupa kata sandi,
2. Mendapatkan akses kembali dengan cepat,
3. Melakukan reset password tanpa harus menghubungi admin.

Jika Anda ingin kembali ke halaman login, klik tombol "Kembali ke Login" di bawahnya.

Email Reset Password



29. Email Reset Password

Email ini dikirim karena pengguna melakukan permintaan untuk me-reset password akun mereka.

Tombol “Reset Password”:

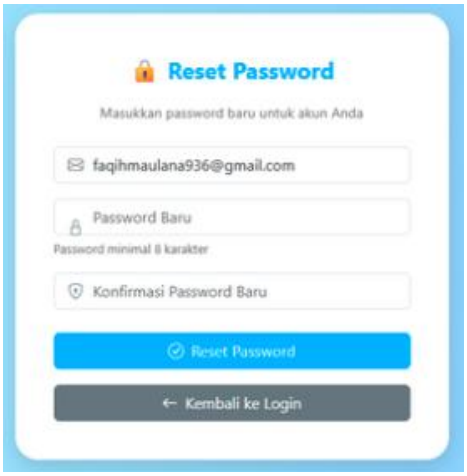
1. Klik tombol ini untuk menuju halaman pengaturan ulang password. Link hanya berlaku selama 6 menit.
2. Catatan Tambahan:
3. Jika Anda tidak merasa meminta reset password, abaikan email ini. Tidak ada tindakan lanjutan yang perlu dilakukan.

Alternatif Link:

- Jika tombol tidak dapat diklik, tersedia juga tautan langsung yang bisa disalin ke browser.

Email ini merupakan bagian dari sistem keamanan untuk membantu pengguna memulihkan akses ke akun mereka secara mandiri.

Reset-Password Tenant

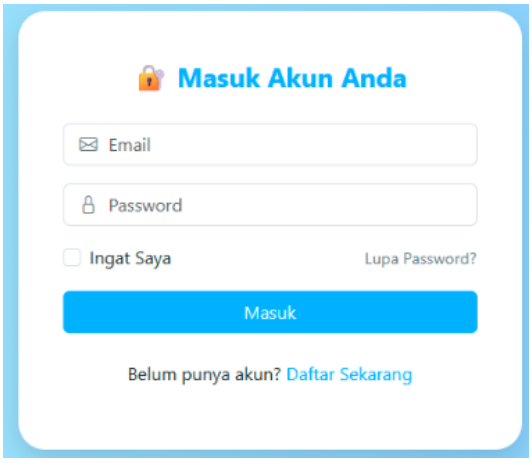


Gambar 30. Reset-Password Tenant

Halaman ini muncul setelah Anda mengklik link reset password dari email. Di sini, Anda dapat memasukkan password baru dan konfirmasi password untuk mengganti kata sandi akun Anda. Setelah diisi, klik tombol "Reset Password" untuk menyimpan perubahan.

Jika ingin membatalkan, Anda bisa klik tombol "Kembali ke Login".

Login Tenant Kembali



Gambar 31. Login Tenant Kembali

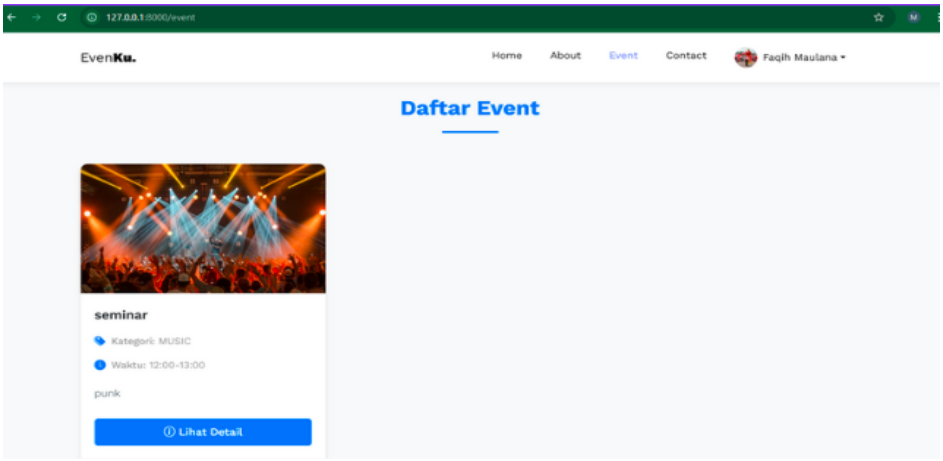
Setelah Anda mengklik tombol "Reset Password", sistem akan menyimpan password baru Anda dan secara otomatis mengarahkan Anda ke halaman login.

Di halaman login, Anda dapat langsung masuk kembali menggunakan:

1. Email Anda, dan
2. Password baru yang baru saja Anda atur.

Proses ini memastikan akun Anda dapat diakses kembali dengan aman menggunakan kata sandi yang baru.

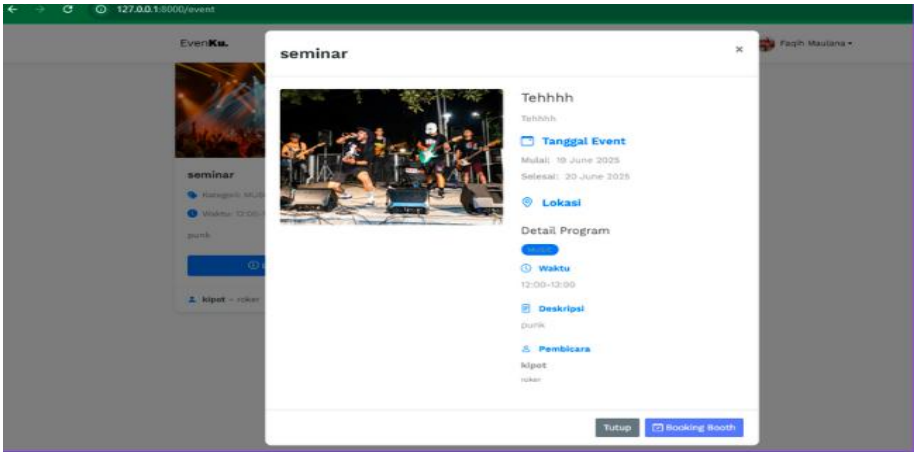
13. Event



Gambar 32. Event

Setelah Login Tenant Dapat Mengklik menu event untuk melihat daftar acara apa saja yang di selenggarakan di dalam event yang di adakan.

Detail Event



Gambar 33. Detail Event

Klik tombol "Detail Event" untuk melihat informasi lengkap mengenai acara yang akan diselenggarakan. Di dalam halaman detail ini, Anda dapat menemukan berbagai informasi penting seperti:

1. Nama dan deskripsi acara
2. Daftar bintang tamu atau pengisi acara.
3. Tanggal dan waktu pelaksanaan event.
4. Lokasi acara.
5. serta informasi tambahan lainnya yang akan membantu Anda lebih memahami isi dan tujuan dari event tersebut.

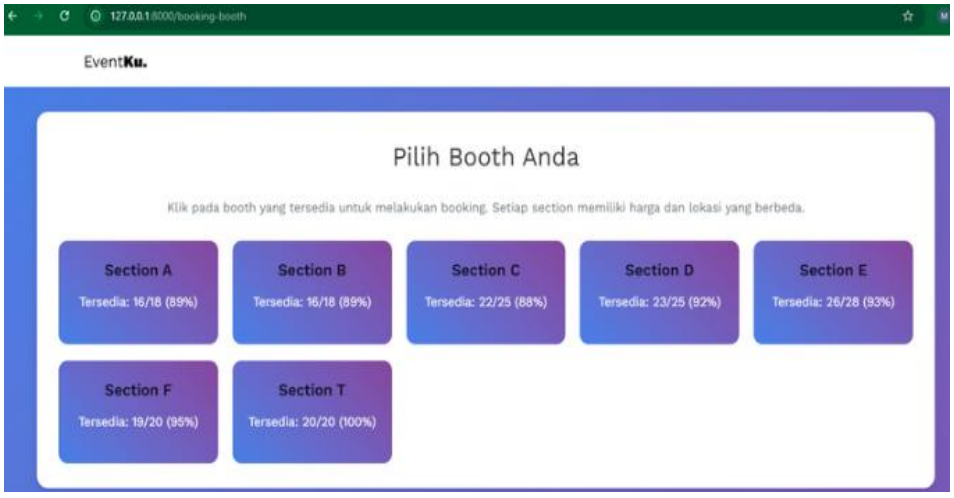
Apabila ingin Melakukan Booking Bisa Klik Button **Booking Booth** dibawah.

14. Booking Booth

Lihat Section Tersedia

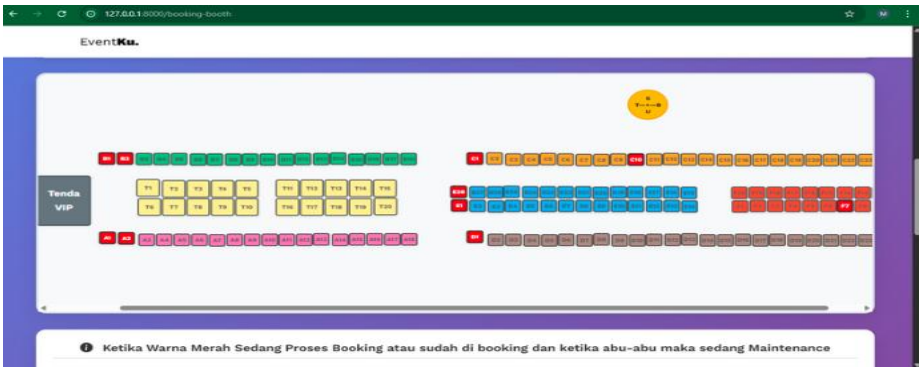
Tampilan Tersebut memungkinkan tenant (penyewa) untuk Melihat Informasi Booth Yang tersedia, Pada halaman booking, tenant bisa:

1. Melihat section booth yang tersedia (misalnya Section A, B, C, dst).
2. Mengetahui jumlah booth yang masih bisa dipilih di setiap section.
3. Mempermudah tenant untuk memilih booth yang diinginkan.



Gambar 34. Lihat Section Tersedia

Pilih Booth Manual



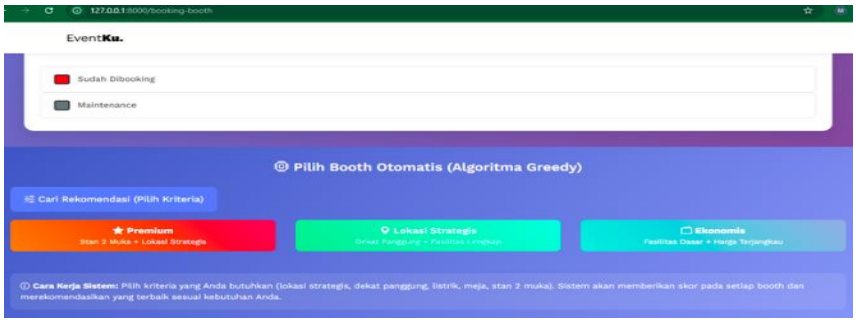
Gambar 35. Pilih Booth Manual

Tenant dapat melakukan pemesanan booth secara online melalui halaman Booking Booth di aplikasi EventKu. Cukup ikuti langkah berikut:

1. Pilih Section yang masih tersedia pada tampilan awal.
2. Setelah itu, pilih booth yang berwarna selain merah atau abu-abu (booth berwarna merah berarti sudah dibooking, abu-abu sedang maintenance).
3. Klik booth yang diinginkan, lalu isi data yang diminta (seperti nama tenant, kontak, dan lainnya).
4. Lakukan konfirmasi dan ikuti instruksi pembayaran jika diperlukan.

Setelah berhasil, booth akan otomatis terdaftar ke dalam admin atas nama tenant dan ditandai sebagai booked di sistem.

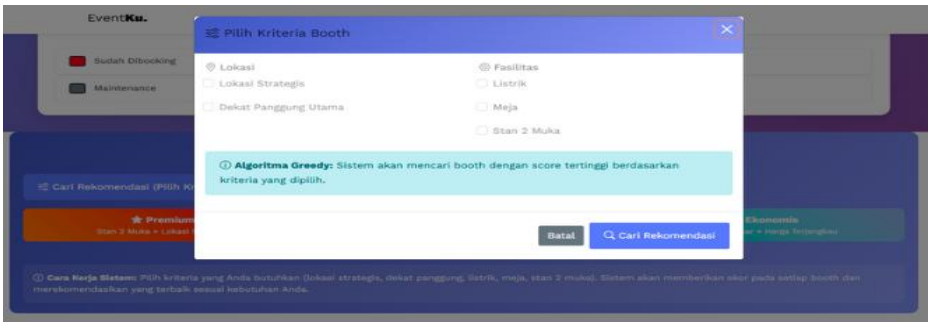
Pilih Booth Rekomendasi Algoritma Greedy



Gambar 36. Pilih Booth Rekomendasi Algoritma Greedy

Sistem akan memilih booth terbaik secara cepat (greedy) berdasarkan poin tertinggi dari kombinasi kriteria yang dipilih — tanpa mengecek semua kemungkinan kombinasi seperti algoritma kompleks lainnya. Ini membuat rekomendasi cepat, efisien, dan relevan.

Cari Rekomendasi



Gambar 37. Cari Rekomendasi

Fitur ini membantu tenant memilih booth secara otomatis berdasarkan kriteria yang dipilih, seperti:

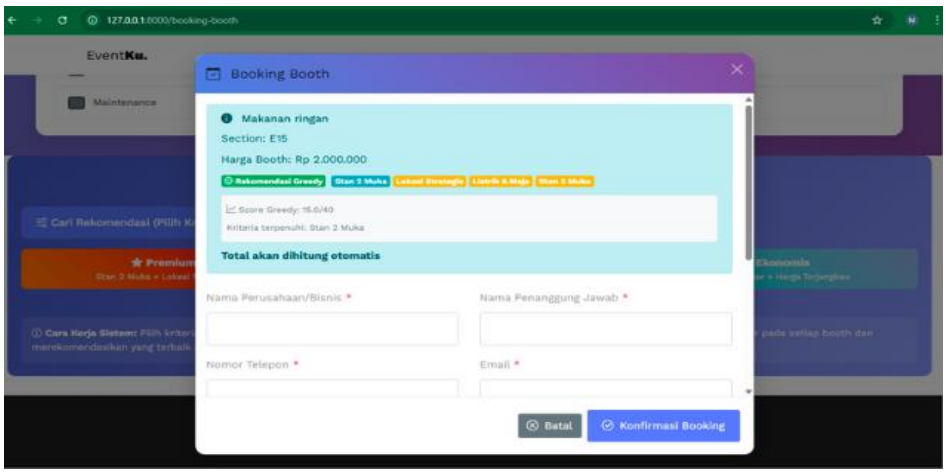
1. Lokasi Strategis
2. Dekat Panggung Utama
3. Fasilitas: Listrik, Meja, Stan 2 Muka

Setelah tenant memilih kriteria yang dibutuhkan, klik tombol “Cari Rekomendasi”, maka:

Algoritma Greedy akan bekerja dengan menghitung skor tertinggi dari setiap booth berdasarkan kecocokan kriteria, lalu merekomendasikan booth terbaik yang tersedia.

Fitur ini sangat cocok untuk tenant yang ingin cepat memilih booth tanpa harus mengecek satu per satu secara manual.

Pilihan Rekomendasi



Gambar 38. Pilihan Rekomendasi

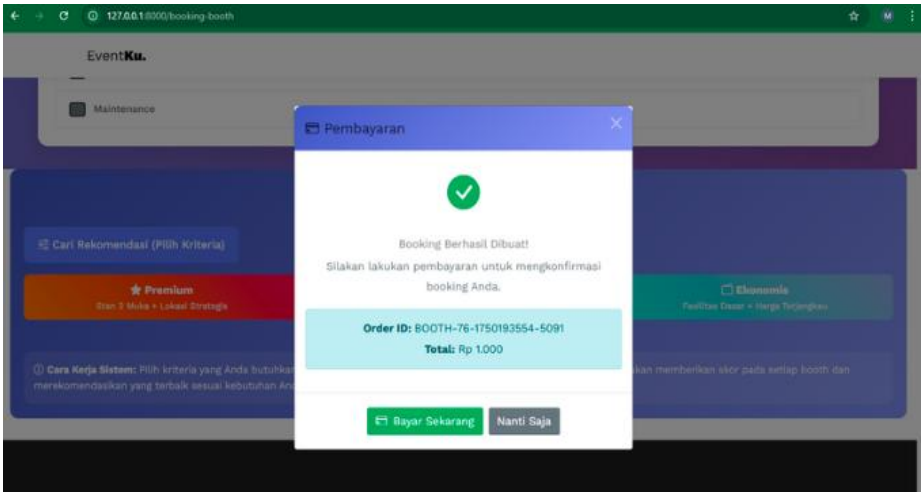
Tampilan ini menampilkan booth yang direkomendasikan secara otomatis oleh sistem berdasarkan algoritma Greedy sesuai kriteria yang dipilih tenant (seperti Stan 2 Muka, Lokasi Strategis, Listrik & Meja).

Informasi yang ditampilkan meliputi:

1. Section dan harga booth.
2. Skor rekomendasi (Score Greedy) berdasarkan kecocokan kriteria.
3. Form isian data tenant (Nama Bisnis, Penanggung Jawab, Kontak).

Tenant cukup mengisi formulir dan klik "Konfirmasi Booking" untuk menyelesaikan pemesanan.

15. Pembayaran



Gambar 39. Pembayaran

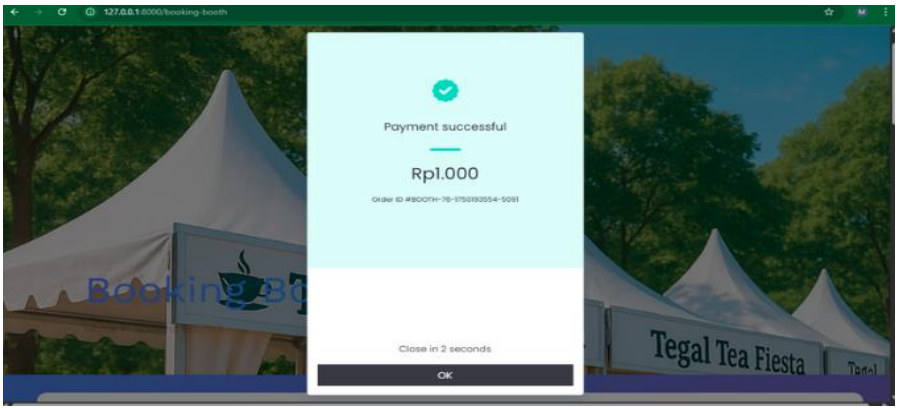
Tampilan ini muncul setelah tenant berhasil melakukan booking booth.

1. Booking Berhasil Dibuat!
2. Ditampilkan Order ID dan Total Biaya.
3. Tenant diminta untuk segera melakukan pembayaran untuk mengonfirmasi booking-nya.

Tersedia dua pilihan:

1. Bayar Sekarang → melanjutkan ke proses pembayaran.
2. Nanti Saja → menunda pembayaran, namun booking tetap tercatat sementara.

Hasil Pembayaran



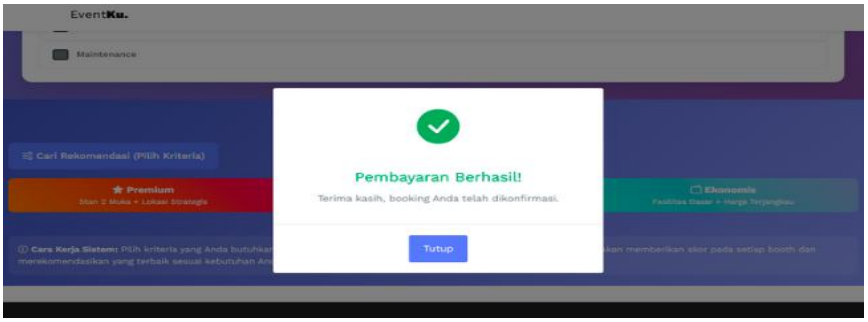
Gambar 40. Hasil Pembayaran

Tampilan ini muncul sebagai konfirmasi bahwa proses pembayaran telah berhasil.

1. Status: Payment successful
2. Jumlah: Rp 1.000
3. Order ID: Ditampilkan sebagai bukti transaksi
4. Penutup Otomatis: Halaman akan tertutup otomatis setelah hitungan mundur selesai, atau bisa klik OK secara manual.

Tampilan ini menandakan bahwa booking booth telah terkonfirmasi secara resmi dan tenant siap menempati booth yang telah dipilih.

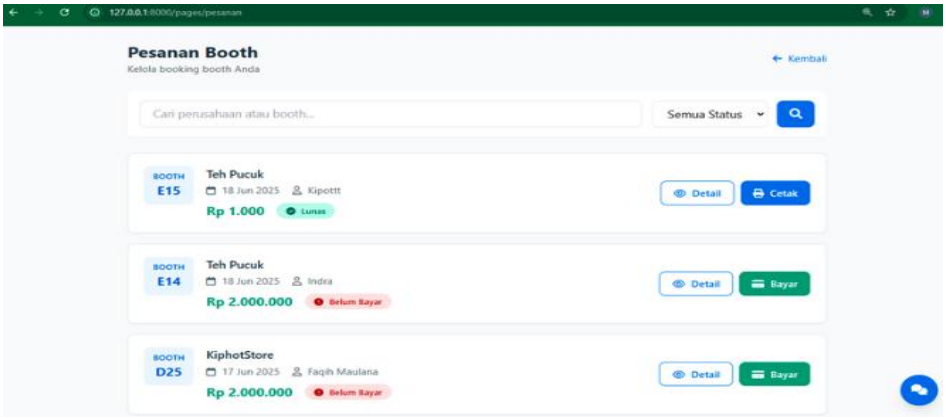
Status Pembayaran



Gambar 41. Status Pembayaran

sistem secara otomatis akan menampilkan notifikasi "Pembayaran Berhasil" sebagai bukti bahwa proses booking booth telah selesai dan dikonfirmasi. Tenant tidak perlu melakukan tindakan tambahan—booth yang dipilih kini telah resmi dibooking dan siap digunakan saat event berlangsung.

16. Pesanan



Gambar 42. Pesanan

Halaman ini menampilkan daftar seluruh booth yang telah dibooking oleh tenant. Informasi yang disajikan meliputi:

1. Nomor Booth, Nama Usaha, Tanggal Event, dan Nama Pemesan
2. Harga Booth dan Status Pembayaran:

a. Lunas → Booking sudah dibayar

b. Belum Bayar → Booking belum dibayar

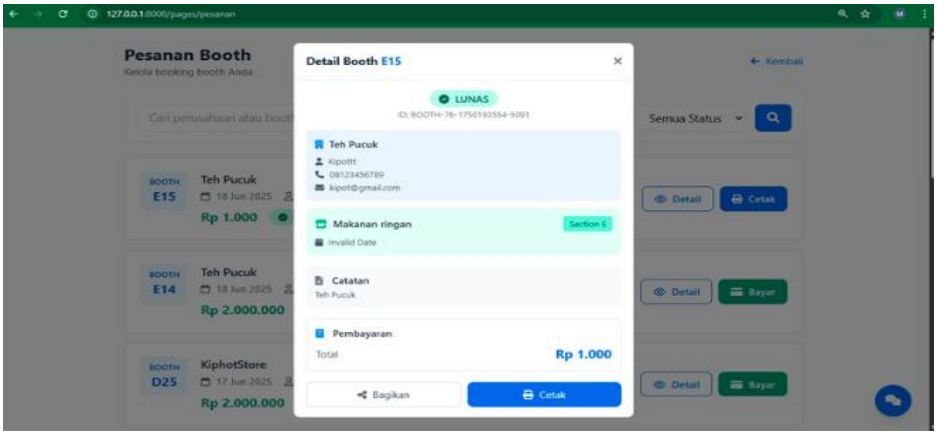
3. Tombol aksi:

a. Detail → Melihat informasi lengkap pemesanan

b. Bayar → Melanjutkan pembayaran

c. Cetak → Mencetak bukti booking (jika sudah lunas)

Detail Pesanan



Gambar 43. Detail Pesanan

Tampilan ini menampilkan detail lengkap dari booth yang telah dipesan oleh tenant. Informasi yang disajikan meliputi:

1. Nama Bisnis & Penanggung Jawab: Nama usaha, nama pemesan, nomor telepon, dan email.
2. Kategori Booth: Seperti "Makanan ringan" dan lokasi di "Section E".
3. Status Pembayaran: Ditandai sebagai LUNAS, menunjukkan bahwa pembayaran telah berhasil.
4. Total Biaya: Tertera biaya akhir pemesanan booth.
5. Fitur Tambahan: Terdapat tombol Cetak untuk mencetak bukti pemesanan.

Pdf Detail Pesanan

Gambar 44. Pdf Detail Pesanan

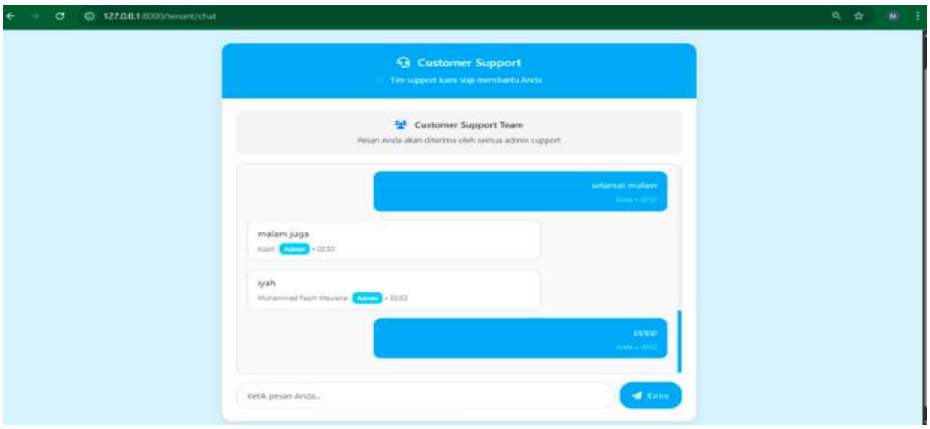
setelah pengguna mengklik tombol "Cetak", sistem akan menampilkan atau menghasilkan invoice booking booth dalam format PDF seperti pada gambar. Penjelasan Singkat:

1. Invoice ini merupakan bukti pembayaran resmi untuk pemesanan booth di platform EventKu.
2. Status pembayaran: LUNAS

3. Menampilkan informasi penting seperti Data perusahaan dan kontak tenant
4. Detail booth yang dipesan (ID, nama booth, section, tanggal booking)
5. Catatan khusus (jika ada)
6. Rincian pembayaran, termasuk total dan waktu pembayaran

Dokumen ini dapat diunduh atau dicetak sebagai arsip bukti transaksi antara tenant dan platform EventKu.

17. Chat



Gambar 45. Chat

Pada Tampilan Pesanan, pengguna (tenant) dapat berkomunikasi langsung dengan tim admin dengan mengklik icon chat di pojok kanan bawah untuk bertanya atau menyampaikan kendala.

Fitur utama:

1. Chat dua arah dengan tampilan pesan yang dibedakan antara tenant dan admin.
2. Label Admin memudahkan identifikasi pesan dari pihak support.
3. Form input dan tombol Kirim memungkinkan pengiriman pesan secara instan.
4. Informasi dukungan seperti nama pengirim, waktu pengiriman, dan isi pesan ditampilkan secara rapi.

Fitur ini dirancang untuk meningkatkan pelayanan pelanggan dan memastikan komunikasi yang cepat serta responsif antara tenant dan admin EventKu.

Lampiran 5. Dokumen Teknikal

Dokumen Teknikal

Rancang Bangun Aplikasi Booking Booth Tenant pada Cresindo Event Organizer

Muhammad Faqih Maulana
Ir. Ginanjar Wiro Sasmito, S.Kom.,M.Kom.
M. Nishom, S.Kom., M.Kom.

Pendahuluan

Eventku Adalah Aplikasi Booking Booth Tenant Dengan Algoritma Greedy yang dirancang khusus untuk memfasilitasi proses penyewaan booth bagi para tenant (pedagang atau pemilik usaha) dalam suatu event, seperti bazar, pameran, atau festival.

Tujuan Utama Dari Pembuatan Aplikasi ini Adalah untuk :

1. Memudahkan tenant dalam memesan booth secara online tanpa harus datang langsung atau melalui proses manual.
2. Menggunakan algoritma greedy untuk memilih booth terbaik secara cepat berdasarkan preferensi tenant seperti lokasi, ukuran, dan jenis booth.
3. Memastikan seluruh booth terisi secara efisien, meminimalkan ruang kosong, dan menyeimbangkan penempatan tenant di area strategis.
4. Memberikan pengalaman pemesanan yang cepat, transparan, dan nyaman, sehingga tenant merasa puas dan lebih loyal terhadap penyelenggara event.
5. Memungkinkan admin event untuk memantau proses booking secara real-time, mengatur layout booth, serta melihat data pemesanan secara sistematis.
6. Tenant dapat langsung melakukan pembayaran booth secara online melalui aplikasi menggunakan berbagai metode transfer bank sehingga proses menjadi lebih praktis, aman, dan real-time.

Dengan antarmuka yang interaktif dan ramah pengguna, Eventku memudahkan proses pemesanan booth bagi tenant, sekaligus membantu penyelenggara event dalam mengelola data penyewaan secara efisien.

Penerapan algoritma Greedy memungkinkan sistem secara cerdas memilih booth terbaik sesuai preferensi tenant secara otomatis dan cepat. Selain itu, fitur pembayaran langsung dalam aplikasi membuat seluruh proses dari pemilihan hingga pembayaran menjadi lebih praktis.

Spesifikasi Teknis

Spesifikasi teknis meliput :

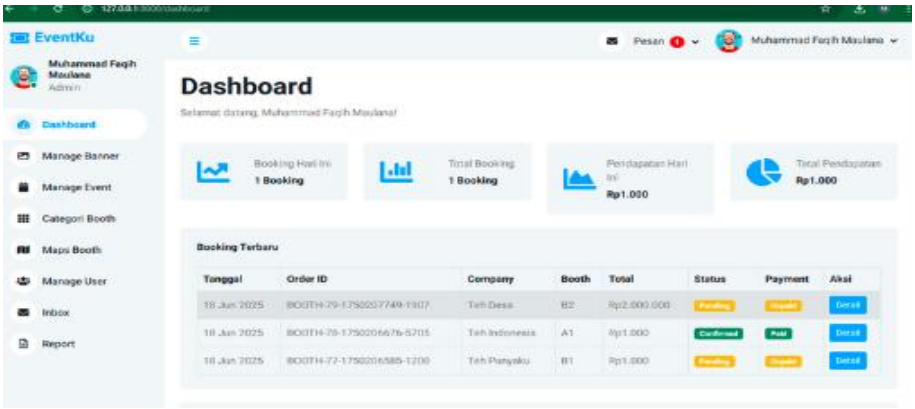
1. Modul Pengguna
2. Source Code

Uraian Spesifikasi yang diperlukan saat Pembuatan Aplikasi :

1. Windows 11 Ram 12 SSD 256
2. Visual Studio Code
3. Web Browser

Di Bawah ini Adalah Uraian Spesifikasi Modul :

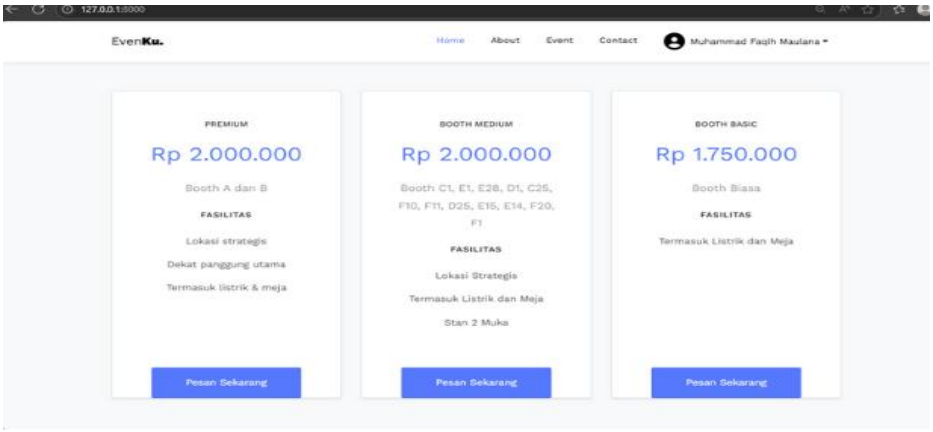
1. Modul Untuk Tampilan Admin



Gambar 1. Modul Untuk Tampilan Admin

Tampilan dashboard ini merupakan halaman khusus yang hanya dapat diakses oleh admin. Sistem menggunakan mekanisme Role-Based Access Control (RBAC) untuk membatasi akses, sehingga hanya pengguna dengan role admin yang bisa melihat dan mengelola data seperti booking, pendapatan, banner, event, dan user.

2. Modul Untuk Tampilan Pengguna (Tenant)



Gambar 2. Modul Untuk Tampilan Pengguna (Tenant)

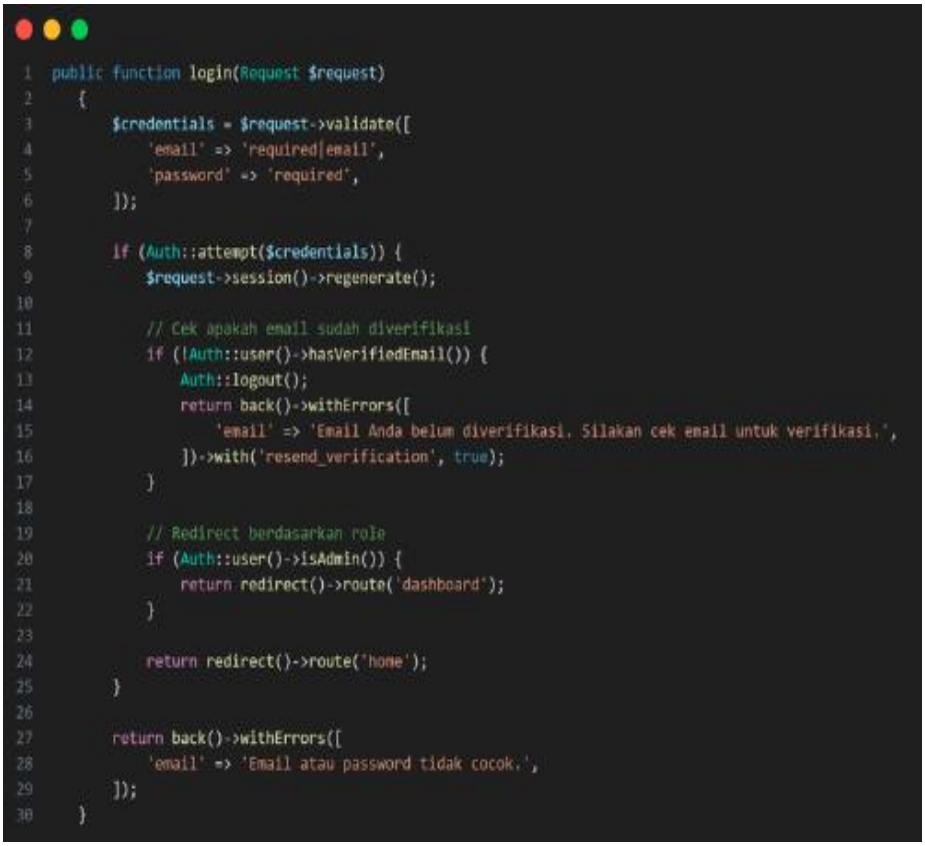
Tampilan ini diperuntukkan bagi masyarakat umum atau tenant yang ingin berpartisipasi dalam event dengan menyewa booth. Melalui modul ini, tenant dapat melihat informasi booth yang tersedia, memilih kategori booth (Premium, Medium, atau Basic), melihat harga dan fasilitas yang ditawarkan, serta melakukan pemesanan langsung secara online melalui tombol "Pesan Sekarang". Modul ini memudahkan tenant untuk melakukan booking booth secara cepat, transparan, dan terintegrasi dengan sistem pembayaran.

SOURCE CODE :

- Source Code dari Aplikasi Booking Booth Tenant dibangun menggunakan framework Laravel sebagai backend untuk menangani proses autentikasi, pengelolaan data tenant, kategori booth, transaksi booking, dan status pembayaran. Laravel digunakan untuk mempermudah pengembangan fitur CRUD (Create, Read, Update, Delete), validasi data, dan pengaturan hak akses berdasarkan peran pengguna (admin dan tenant).
- Untuk penyimpanan data, digunakan database MySQL yang berfungsi menyimpan informasi seperti data pengguna, data booth, riwayat pemesanan, dan status pembayaran. Selain itu, tampilan antarmuka frontend dirancang agar responsif dan mudah digunakan oleh masyarakat/tenant dalam melakukan pemesanan secara online.

1. Login

Proses login memungkinkan pengguna yang sudah terdaftar untuk mengakses akun mereka sesuai dengan peran masing-masing, baik sebagai admin maupun tenant. Sistem autentikasi ini menggunakan middleware bawaan Laravel yang dikombinasikan dengan pengecekan role dari tabel roles, sehingga setiap pengguna hanya dapat mengakses halaman dan fitur sesuai dengan hak aksesnya. Admin memiliki akses penuh ke manajemen sistem, sedangkan tenant hanya dapat mengakses fitur pemesanan booth dan riwayat booking.



```php
public function login(Request $request)
    {
        $credentials = $request->validate([
            'email' => 'required|email',
            'password' => 'required',
        ]);

        if (Auth::attempt($credentials)) {
            $request->session()->regenerate();

            // Cek apakah email sudah diverifikasi
            if (!Auth::user()->hasVerifiedEmail()) {
                Auth::logout();
                return back()->withErrors([
                    'email' => 'Email Anda belum diverifikasi. Silakan cek email untuk verifikasi.',
                ])->with('resend_verification', true);
            }

            // Redirect berdasarkan role
            if (Auth::user()->isAdmin()) {
                return redirect()->route('dashboard');
            }

            return redirect()->route('home');
        }

        return back()->withErrors([
            'email' => 'Email atau password tidak cocok.',
        ]);
    }
```

Gambar 3. Login

Kode ini mengelola proses autentikasi pengguna menggunakan Laravel dengan role-based access control (RBAC), dengan rincian sebagai berikut:

1. Validasi Input Login:
   a. Menggunakan $request->validate() untuk memastikan:
   b. email: wajib diisi dan dalam format email valid

c. password: wajib diisi

2. Autentikasi User:

Jika kredensial valid (Auth::attempt($credentials)), maka:

a. Sistem melakukan session()->regenerate() untuk menghindari session hijacking.
b. User berhasil login dan dilanjutkan ke proses verifikasi email.

3. Verifikasi Email:

Setelah login:

a. Sistem mengecek apakah user sudah verifikasi email melalui hasVerifiedEmail().
b. Jika belum, user langsung di-logout() dan muncul pesan bahwa email harus diverifikasi.
c. Opsi pengiriman ulang link verifikasi juga ditambahkan melalui flag resend_verification.

4. Redirect Berdasarkan Role:

Jika email sudah diverifikasi:

a. Sistem memeriksa peran pengguna dengan isAdmin():
b. Jika admin, diarahkan ke route dashboard
c. Jika bukan (misal tenant), diarahkan ke route home

5. Jika Gagal Login:

Jika autentikasi gagal Sistem memberikan pesan error Email atau password tidak cocok.

2. Register

Proses registrasi memungkinkan pengguna baru untuk membuat akun sesuai dengan peran yang ditentukan, seperti tenant. Setelah berhasil mendaftar, sistem akan secara otomatis mengirimkan email verifikasi ke alamat email yang didaftarkan. Sistem ini menggunakan autentikasi Laravel yang dikombinasikan dengan middleware dan pengecekan role dari tabel roles, sehingga setiap pengguna hanya dapat mengakses halaman dan fitur sesuai dengan hak aksesnya setelah email berhasil diverifikasi. Verifikasi

email ini bertujuan untuk memastikan keaslian akun dan menjaga keamanan sistem.

```php
public function register(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:8|confirmed',
        ]);

        if ($validator->fails()) {
            return back()->withErrors($validator)->withInput();
        }

        // Dapatkan role tenant
        $tenantRole = Role::where('name', 'tenant')->firstOrFail();

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
            'role_id' => $tenantRole->id,
        ]);

        // Login user setelah registrasi (perlu untuk verification notice)
        Auth::login($user);

        // Trigger event untuk mengirim email verifikasi
        event(new Registered($user));

        return redirect()->route('verification.notice')
            ->with('message', 'Registrasi berhasil! Silakan cek email Anda untuk verifikasi.');
    }
```

Gambar 4. Register

Kode ini mengelola proses registrasi pengguna menggunakan Laravel, dengan rincian sebagai berikut:

1. Properti yang Digunakan:
    a. $request->all() untuk mengambil input dari form registrasi, seperti name, email, dan password.
2. Aturan Validasi dilakukan untuk memastikan:
    a. name: wajib diisi, berupa teks, maksimal 255 karakter.
    b. email: wajib diisi, format email yang valid, dan belum pernah terdaftar.
    c. password: wajib diisi, minimal 8 karakter, dan harus dikonfirmasi.
    d.

3. Pesan Validasi:

   Jika validasi gagal, sistem akan mengembalikan pengguna ke halaman sebelumnya dengan menampilkan pesan kesalahan yang relevan.
4. Penentuan Role:

   Sistem akan mengambil role dengan nama "tenant" dari tabel roles, untuk memastikan bahwa user baru diberi hak akses sebagai tenant.
5. Pembuatan Akun:

   Akun pengguna dibuat dengan menyimpan data ke database, dan password dienkripsi menggunakan Hash. Role pengguna juga disematkan melalui role_id.
6. Login Otomatis Setelah Registrasi:

   Setelah berhasil mendaftar, pengguna langsung login untuk proses verifikasi email selanjutnya.
7. Verifikasi Email:

   Event Registered dipicu agar Laravel mengirimkan email verifikasi ke email pengguna.
8. Redirect:

   Setelah registrasi selesai, pengguna diarahkan ke halaman notifikasi verifikasi email dengan pesan bahwa registrasi berhasil dan diminta memverifikasi email.

3. Desain Layout

Kode Desain layout booth untuk admin ini dirancang agar admin dapat memantau dan mengelola denah booth secara efisien melalui antarmuka visual. Semua fungsi terkait hanya dapat dijalankan oleh admin dengan proteksi role via middleware, serta setiap perubahan akan langsung tersimpan ke database dan tampil di halaman utama tenant.

1. Function Crate Booth

```php
public function store(Request $request): JsonResponse
    {
        try {
            $request->validate([
                'booth_id' => 'required|string|unique:booths,booth_id',
                'booth_name' => 'required|string',
                'section' => 'required|string|in:A,B,C,D,E,F,T',
                'price' => 'required|numeric|min:0',
                'position_x' => 'required|integer',
                'position_y' => 'required|integer',
                'width' => 'nullable|integer|min:20',
                'height' => 'nullable|integer|min:20',
                'status' => 'nullable|string|in:available,booked,maintenance'
            ]);

            // Set default values
            $data = $request->all();
            $data['width'] = $data['width'] ?? 25;
            $data['height'] = $data['height'] ?? 25;
            $data['status'] = $data['status'] ?? 'available';

            $booth = Booth::create($data);

            return response()->json([
                'success' => true,
                'message' => 'Booth created successfully',
                'booth' => $booth
            ]);
        } catch (\Illuminate\Validation\ValidationException $e) {
            return response()->json([
                'success' => false,
                'message' => 'Validation failed',
                'errors' => $e->errors()
            ], 422);
        } catch (\Exception $e) {
            return response()->json([
                'success' => false,
                'message' => 'Failed to create booth: ' . $e->getMessage()
            ], 500);
        }
    }
```

Gambar 5. Funtion Create Booth

Fungsi ini bertanggung jawab untuk menambahkan booth baru ke dalam database. Fitur ini hanya dapat diakses oleh admin, biasanya melalui panel manajemen booth.

a. Validasi Data Masukan

1. Sebelum menyimpan, data dari permintaan ($request) akan divalidasi dengan ketentuan:
2. booth_id: wajib, string, dan unik di tabel booths.
3. booth_name: wajib, string.
4. section: wajib, dan harus salah satu dari: A, B, C, D, E, F, T.
5. price: wajib, angka, minimal 0.
6. position_x, position_y: wajib, integer (untuk penempatan booth dalam layout).
7. width, height: opsional, integer minimal 20.
8. status: opsional, nilai default 'available', hanya boleh available, booked, atau maintenance.

b. Set Nilai Default

1. Jika nilai width, height, atau status tidak disediakan, sistem akan mengisi otomatis width dan height default = 25.
2. status default = 'available'.

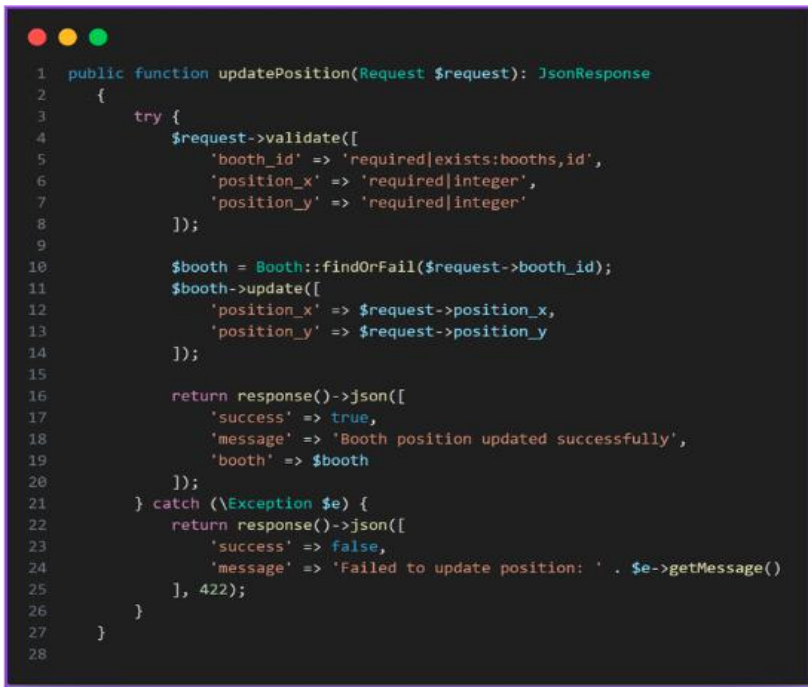
c. Menyimpan Data

- Data yang sudah valid akan disimpan ke dalam tabel booths menggunakan Booth::create().

d. Respons JSON

1. Sukses: Mengembalikan status success, pesan sukses, dan data booth yang dibuat.
2. Gagal Validasi: Mengembalikan status 422 dengan detail error validasi.
3. Gagal Umum: Mengembalikan status 500 jika terjadi kesalahan selain validasi.

2. Function Update Posisi Booth



```php
public function updatePosition(Request $request): JsonResponse
    {
        try {
            $request->validate([
                'booth_id' => 'required|exists:booths,id',
                'position_x' => 'required|integer',
                'position_y' => 'required|integer'
            ]);

            $booth = Booth::findOrFail($request->booth_id);
            $booth->update([
                'position_x' => $request->position_x,
                'position_y' => $request->position_y
            ]);

            return response()->json([
                'success' => true,
                'message' => 'Booth position updated successfully',
                'booth' => $booth
            ]);
        } catch (\Exception $e) {
            return response()->json([
                'success' => false,
                'message' => 'Failed to update position: ' . $e->getMessage()
            ], 422);
        }
    }

```

Gambar 6. Funtion Update Posisi Booth

Fungsi ini digunakan untuk mengubah posisi koordinat booth pada layout event. Umumnya dipakai oleh admin melalui fitur drag & drop atau editor denah.

1. Validasi Data Masukan
    a. Sebelum melakukan pembaruan, data yang dikirim melalui request akan divalidasi.
    b. booth_id: wajib diisi dan harus ada di tabel booths.
    c. position_x: wajib, berupa angka (integer).
    d. position_y: wajib, berupa angka (integer).
2. Update Data Setelah validasi berhasil:
    a. Sistem mencari booth berdasarkan booth_id
    b. Nilai position_x dan position_y diperbarui sesuai input

3. Respons JSON

a. Sukses: Menampilkan status success, pesan sukses, dan data booth terbaru.

b. Gagal: Jika ada kesalahan (seperti booth_id tidak ditemukan), sistem akan mengembalikan status error 422 beserta pesan kesalahan.

3. Function Update Isi Booth

```php
public function update(Request $request, $id): JsonResponse
    {
        try {
            $booth = Booth::findOrFail($id);

            // Validate with unique rule excluding current booth
            $validated = $request->validate([
                'booth_id' => [
                    'sometimes',
                    'string',
                    Rule::unique('booths', 'booth_id')->ignore($booth->id)
                ],
                'booth_name' => 'sometimes|string',
                'section' => 'sometimes|string|in:A,B,C,D,E,F,T',
                'price' => 'sometimes|numeric|min:0',
                'status' => 'sometimes|string|in:available,booked,maintenance',
                'width' => 'sometimes|integer|min:20',
                'height' => 'sometimes|integer|min:20',
                'position_x' => 'sometimes|nullable|integer',
                'position_y' => 'sometimes|nullable|integer',
            ]);

            // Update only validated fields
            $booth->update($validated);

            return response()->json([
                'success' => true,
                'message' => 'Booth updated successfully',
                'booth' => $booth->fresh() // Get fresh data from database
            ]);

        } catch (\Illuminate\Database\Eloquent\ModelNotFoundException $e) {
            return response()->json([
                'success' => false,
                'message' => 'Booth not found'
            ], 404);
        } catch (\Illuminate\Validation\ValidationException $e) {
            return response()->json([
                'success' => false,
                'message' => 'Validation failed',
                'errors' => $e->errors()
            ], 422);
        } catch (\Exception $e) {
            return response()->json([
                'success' => false,
                'message' => 'Failed to update booth: ' . $e->getMessage()
            ], 500);
        }
    }
```

Gambar 7. Funtion Update isi booth

Fungsi ini digunakan untuk memperbarui data booth tertentu berdasarkan ID yang dikirim. Umumnya digunakan dalam fitur manajemen booth oleh admin.

1. Validasi Dinamis
    a. Sistem melakukan validasi terhadap input yang dikirim, namun bersifat optional (gunakan sometimes), artinya hanya field yang dikirim saja yang akan divalidasi dan diperbarui.
    b. booth_id: unik di tabel booths, tetapi mengabaikan booth dengan ID saat ini (ignore($booth->id)).
    c. booth_name: string.
    d. section: hanya boleh A, B, C, D, E, F, atau T.
    e. price: angka positif.
    f. status: hanya available, booked, atau maintenance.
    g. width dan height: integer minimal 20.
    h. position_x, position_y: opsional (boleh null), integer.
2. Proses Update
    a. Sistem mengambil data booth berdasarkan ID (menggunakan findOrFail).
    b. Jika ditemukan, hanya field yang valid dan dikirim saja yang di-update.
    c. Setelah update berhasil, ->fresh() dipanggil untuk mengambil data terbaru dari database.
3. Respon JSON
    a. Sukses:
    - Status true, pesan sukses, dan data booth terbaru.
    b. Gagal:
    - Jika ID tidak ditemukan → error 404.
    - Jika validasi gagal → error 422 + rincian kesalahan.
    - Jika error lain terjadi → error 500.

4. Function Hapus Booth

```php
public function destroy($id): JsonResponse
{
    try {
        $booth = Booth::findOrFail($id);
        $booth->delete();

        return response()->json([
            'success' => true,
            'message' => 'Booth deleted successfully'
        ]);
    } catch (\Illuminate\Database\Eloquent\ModelNotFoundException $e) {
        return response()->json([
            'success' => false,
            'message' => 'Booth not found'
        ], 404);
    } catch (\Exception $e) {
        return response()->json([
            'success' => false,
            'message' => 'Failed to delete booth: ' . $e->getMessage()
        ], 500);
    }
}

```

Gambar 8. Funtion Hapus Booth

Fungsi ini digunakan oleh admin untuk menghapus data booth berdasarkan ID tertentu. Umumnya diakses dari halaman manajemen booth (admin dashboard).

1. Pencarian Booth
   a. Mencari data booth berdasarkan ID.
   b. Jika tidak ditemukan, akan dilempar ke blok catch dengan status error 404.
2. Proses Penghapusan
   - Menghapus data booth dari database secara permanen.
3. Respon JSON Jika berhasil:
   a. Status success: true
   b. Pesan sukses: "Booth deleted successfully".
4. Jika ID tidak ditemukan:
   a. Status 404 dengan pesan "Booth not found".
   b. Jika gagal umum (misalnya kesalahan database).
   c. Status 500 dengan pesan error yang dijelaskan.

5. Function Inisialisasi Booth

```php
public function initializeBooths(): JsonResponse
    {
        try {
            // Initialize default booths if table is empty
            if (Booth::count() > 0) {
                return response()->json([
                    'success' => false,
                    'message' => 'Booths already initialized'
                ]);
            }

            $defaultBooths = $this->getDefaultBoothsData();

            foreach ($defaultBooths as $boothData) {
                Booth::create($boothData);
            }

            return response()->json([
                'success' => true,
                'message' => 'Booths initialized successfully',
                'count' => count($defaultBooths)
            ]);
        } catch (\Exception $e) {
            return response()->json([
                'success' => false,
                'message' => 'Failed to initialize booths: ' . $e->getMessage()
            ], 500);
        }
    }
```

Gambar 9. Function Inisialisasi booth

Fungsi ini digunakan untuk mengisi data awal booth secara otomatis ke dalam tabel booths, biasanya saat sistem pertama kali digunakan. Umumnya dipakai oleh admin dalam fase setup awal event atau sistem.

1. Cek Apakah Sudah Terisi
   a. Fungsi akan mengecek apakah tabel booths sudah berisi data.
   b. Jika sudah ada data → proses dihentikan dan sistem mengembalikan respon gagal ("Booths already initialized").
2. Ambil Data Booth Default
   a. Memanggil fungsi bantu getDefaultBoothsData() untuk mengambil data booth awal.
   b. Biasanya berisi array booth lengkap (misal: id, nama, posisi, ukuran, dll).
3. Simpan Semua Booth
   - Data booth default di-loop dan dimasukkan satu per satu ke database.
4. Respons JSON

   Sukses:

   a. Menampilkan pesan "Booths initialized successfully"
   b. Menyertakan jumlah booth yang berhasil dimasukkan

Gagal:

a. Jika sudah ada data → respon gagal dengan pesan bahwa booth sudah diinisialisasi.
b. Jika error lain terjadi → respon 500 dengan detail error.

4. Booking Booth

1.Function Create Booking Tenant

```php
public function store(Request $request)
    {
        if (!Auth::check()) {
            return response()->json(['success' => false, 'message' => 'Silakan login terlebih dahulu'], 401);
        }

        $validator = Validator::make($request->all(), [
            'booth_id' => 'required|exists:booths,id',
            'company_name' => 'required|string|max:255',
            'contact_person' => 'required|string|max:255',
            'phone' => 'required|string|max:20',
            'email' => 'required|email|max:255',
            'description' => 'nullable|string|max:1000'
        ]);

        if ($validator->fails()) {
            return response()->json(['success' => false, 'errors' => $validator->errors()], 422);
        }

        $booth = Booth::findOrFail($request->booth_id);
        if ($booth->status !== 'available') {
            return response()->json(['success' => false, 'message' => 'Booth tidak tersedia untuk booking'], 422);
        }

        try {
            $bookingData = [
                'booth_id' => $request->booth_id,
                'company_name' => $request->company_name,
                'contact_person' => $request->contact_person,
                'phone' => $request->phone,
                'email' => $request->email,
                'description' => $request->description,
                'total_price' => $booth->price,
                'status' => 'pending',
                'payment_status' => 'unpaid'
            ];

            if (Schema::hasColumn('bookings', 'user_id')) {
                $bookingData['user_id'] = Auth::id();
            }

            $booking = Booking::create($bookingData);
            $booth->update(['status' => 'booked']);
            $paymentData = $this->midtransService->createTransaction($booking);

            return response()->json([
                'success' => true,
                'message' => 'Booking berhasil dibuat, silakan lakukan payment',
                'booking' => $booking->load('booth'),
                'payment_data' => $paymentData
            ]);

        } catch (\Exception $e) {
            Log::error('Booking creation error: ' . $e->getMessage());
            return response()->json(['success' => false, 'message' => 'Gagal membuat booking: ' . $e->getMessage()], 500);
        }
    }
```

Gambar 10. Function Create Booking Tenant

Fungsi ini digunakan oleh tenant yang sudah login untuk melakukan pemesanan booth.

1. Validasi data: Memastikan input (booth, nama perusahaan, kontak, dll) valid.
2. Cek ketersediaan booth: Hanya booth dengan status 'available' yang bisa dipesan.
3. Simpan booking: Membuat data booking dengan status pending dan unpaid.
4. Update booth: Status booth langsung diubah menjadi 'booked'.
5. Integrasi pembayaran: Sistem menghubungkan ke Midtrans untuk proses pembayaran.
6. Respon JSON: Mengembalikan detail booking dan data pembayaran.

5. Pembayaran

1. Function Pembayaran.

```
public function processPayment(Booking $booking)
    {
        $authCheck = $this->checkBookingAuthorization($booking);
        if ($authCheck) return $authCheck;

        if ($booking->payment_status === 'paid') {
            return response()->json(['success' => false, 'message' => 'Booking sudah dibayar'], 400);
        }

        try {
            if ($booking->order_id) {
                $existingStatus = $this->midtransService->checkTransactionStatus($booking->order_id);

                if ($existingStatus) {
                    $transactionStatus = strtolower($existingStatus['transaction_status'] ?? '');

                    if (in_array($transactionStatus, ['pending', 'settlement', 'capture'])) {
                        if ($booking->snap_token && $this->isSnapTokenValid($booking)) {
                            return response()->json([
                                'success' => true,
                                'payment_data' => [
                                    'snap_token' => $booking->snap_token,
                                    'order_id' => $booking->order_id,
                                    'is_existing' => true
                                ]
                            ]);
                        }
                    }

                    if (in_array($transactionStatus, ['expire', 'deny', 'cancel', 'failure'])) {
                        $this->midtransService->cancelTransaction($booking->order_id);
                        $booking->update([
                            'order_id' => $this->generateNewOrderId($booking),
                            'snap_token' => null,
                            'payment_status' => 'unpaid'
                        ]);
                    }
                }
            }

            $paymentData = $this->midtransService->createTransaction($booking);
            $booking->update([
                'snap_token' => $paymentData['snap_token'],
                'payment_expired_at' => now()->addHours(6)
            ]);

            return response()->json(['success' => true, 'payment_data' => $paymentData]);

        } catch (\Exception $e) {
            Log::error('Payment Process Error: ' . $e->getMessage(), [
                'booking_id' => $booking->id,
                'order_id' => $booking->order_id ?? 'none'
            ]);
            return response()->json(['success' => false, 'message' => 'Gagal memproses pembayaran: ' . $e->getMessage()], 500);
        }
    }
```

Gambar 11. Funtion Pembayaran

Fungsi ini digunakan untuk memproses atau memulai ulang pembayaran booth yang dilakukan oleh tenant.

1. Cek Akses: Hanya pemilik booking yang berhak memproses.
2. Cek Status Pembayaran: Jika sudah 'paid', proses dihentikan.
3. Cek Transaksi Sebelumnya:
   a. Jika masih aktif (pending, settlement, capture) → token lama bisa digunakan kembali.

b. Jika sudah kadaluarsa atau gagal → transaksi dibatalkan, token dan order ID dibuat ulang.

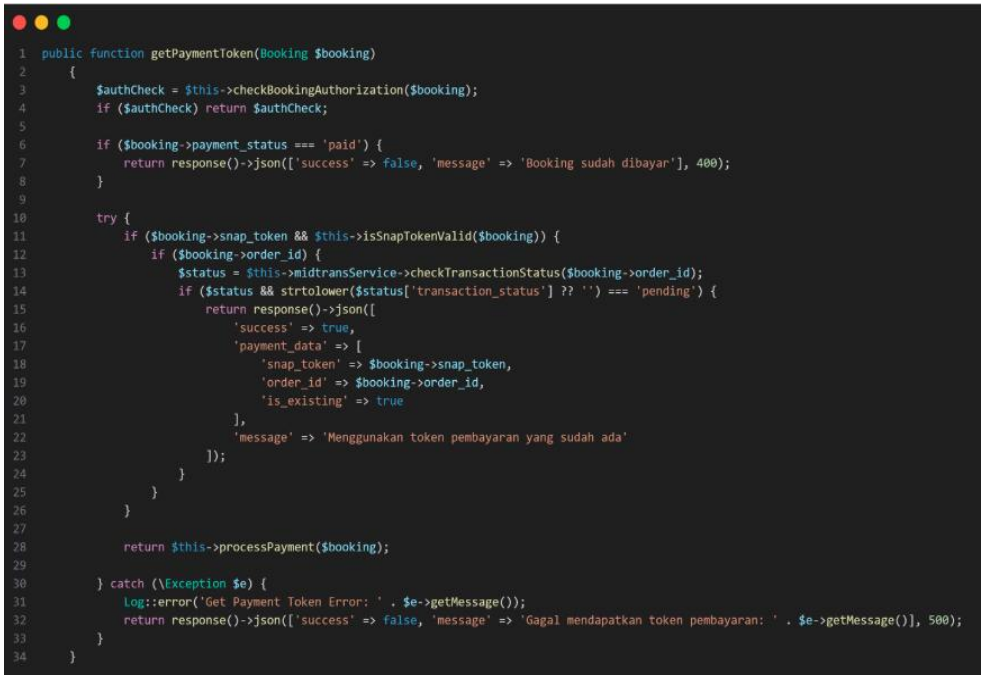
4. Buat Transaksi Baru:

- Jika belum pernah transaksi atau sudah dibatalkan, sistem akan membuat transaksi baru ke Midtrans dan menyimpan snap_token serta batas waktu pembayaran.

5. Respon JSON:

- Mengembalikan data pembayaran (snap_token, order_id) untuk ditampilkan di frontend.

2. Function Payment Token



```php
public function getPaymentToken(Booking $booking)
    {
        $authCheck = $this->checkBookingAuthorization($booking);
        if ($authCheck) return $authCheck;

        if ($booking->payment_status === 'paid') {
            return response()->json(['success' => false, 'message' => 'Booking sudah dibayar'], 400);
        }

        try {
            if ($booking->snap_token && $this->isSnapTokenValid($booking)) {
                if ($booking->order_id) {
                    $status = $this->midtransService->checkTransactionStatus($booking->order_id);
                    if ($status && strtolower($status['transaction_status'] ?? '') === 'pending') {
                        return response()->json([
                            'success' => true,
                            'payment_data' => [
                                'snap_token' => $booking->snap_token,
                                'order_id' => $booking->order_id,
                                'is_existing' => true
                            ],
                            'message' => 'Menggunakan token pembayaran yang sudah ada'
                        ]);
                    }
                }
            }

            return $this->processPayment($booking);

        } catch (\Exception $e) {
            Log::error('Get Payment Token Error: ' . $e->getMessage());
            return response()->json(['success' => false, 'message' => 'Gagal mendapatkan token pembayaran: ' . $e->getMessage()], 500);
        }
    }
```

Gambar 12. Funtion Payment Token

Fungsi ini bertugas mengambil token pembayaran Midtrans (snap_token) untuk tenant agar bisa melanjutkan proses pembayaran booth.

1. Otorisasi Pengguna (Authorization Check)
   a. Sistem terlebih dahulu memverifikasi bahwa pengguna yang sedang login adalah pemilik sah dari data booking.
   b. Jika tidak berhak, sistem langsung menghentikan proses dan mengembalikan respons error otorisasi.

2. Cek Status Pembayaran
    a. Jika status payment_status pada booking sudah "paid", maka proses token dihentikan.
    b. Sistem memberikan respons bahwa pembayaran sudah selesai agar tidak terjadi pembayaran ganda.
3. Penggunaan Token yang Sudah Ada (Reuse Snap Token)
    a. Jika snap_token sudah tersedia dan masih valid, maka sistem akan memeriksa order_id.
    b. Jika order_id terhubung dengan status transaksi yang masih "pending", maka:
    c. Sistem akan mengembalikan token lama dan menandai sebagai is_existing = true.
    d. Hal ini untuk menghindari pembuatan transaksi baru yang tidak perlu (lebih efisien dan aman).
4. Buat Token Baru Jika Tidak Ada atau Kadaluarsa
    a. Jika token tidak tersedia atau sudah tidak valid, atau status transaksi bukan pending, maka:
    b. Fungsi ini akan meneruskan proses ke processPayment().
    c. Di sana akan dilakukan logika lanjutan seperti membuat transaksi baru dan update token.
5. Penanganan Error
    a. Semua error akan ditangkap oleh blok try-catch.
    b. Jika terjadi kegagalan dalam pengecekan atau pembuatan token, sistem akan mencatat ke log dan mengembalikan pesan kesalahan dengan status 500.
    c. 6. Output / Response
    d. Jika berhasil: sistem mengembalikan data snap_token, order_id, dan pesan sukses.
    e. Jika gagal: sistem mengembalikan JSON error dan pesan alasan kegagalan.

3. Function Payment Status

Gambar 13. Funtion Payment Status

Fungsi ini digunakan untuk memeriksa dan memperbarui status pembayaran sebuah booking, dengan integrasi langsung ke Midtrans.

1. Validasi Awal
    a. Booking ID diambil dari parameter $bookingId atau route.
    b. Sistem mengecek apakah:
    c. ID tersedia
    d. Booking ditemukan
    e. User sedang login
    f. User memiliki akses terhadap booking (via checkBookingAuthorization())
2. Cek Order ID
    - Jika order_id tidak tersedia pada booking, maka status pembayaran dianggap unpaid.
3. Ambil Status dari Midtrans
    a. Menggunakan midtransService->checkTransactionStatus() untuk mendapatkan status transaksi terbaru.
    b. Status Midtrans seperti settlement, capture, expire, dsb., akan dipetakan ke sistem lokal (mis. paid, expired, failed) dengan fungsi mapMidtransStatus().
4. Update Data jika Status Berubah
    a. Jika status pembayaran dari Midtrans berbeda dengan data lokal:
    b. Booking diperbarui dengan payment_status, payment_method, transaction_id, dan paid_at (jika sudah lunas).
    c. Status booking dan status booth juga ikut diperbarui melalui updateBookingAndBoothStatus().
5. Respons JSON
    a. Sistem merespons status pembayaran dan booking terbaru.
    b. Jika berhasil update, muncul pesan bahwa status pembayaran diperbarui.
    c. Jika gagal memanggil Midtrans, muncul error log dan pesan kegagalan komunikasi ke gateway.

6. Pesanan

1. Function Menampilkan keseluruhan Pesanan Tenant

```php
public function pesananIndex(Request $request)
    {
        if (!Auth::check()) {
            return redirect()->route('login')->with('error', 'Silakan login terlebih dahulu');
        }

        $query = Booking::with('booth');

        if (!$this->isUserAdmin() && Schema::hasColumn('bookings', 'user_id')) {
            $query->where('user_id', Auth::id());
        }

        $query->latest();

        if ($request->filled('search')) {
            $search = $request->search;
            $query->where(function ($q) use ($search) {
                $q->where('company_name', 'like', "%{$search}%")
                  ->orWhere('contact_person', 'like', "%{$search}%")
                  ->orWhere('phone', 'like', "%{$search}%")
                  ->orWhere('email', 'like', "%{$search}%");
            });
        }

        if ($request->filled('payment_status')) {
            $query->where('payment_status', $request->payment_status);
        }

        $bookings = $query->paginate(10);
        return view('pages.pesanan', compact('bookings'));
    }
```

Gambar 14. Function Menampilkan keseluruhan Pesanan Tenant

Fungsi ini digunakan untuk menampilkan daftar pemesanan booth (booking) yang dapat diakses oleh admin maupun tenant, dengan kontrol akses berbasis role. Penjelasan Fungsional:

1. Autentikasi
   a. Mengecek apakah pengguna sudah login.
   b. Jika belum login, diarahkan ke halaman login dengan pesan error.
2. Query Awal Booking
   - Mengambil data booking beserta relasi booth (with('booth')).

3. Filter Berdasarkan Role
    - Jika bukan admin, maka data dibatasi hanya untuk booking milik user yang sedang login (user_id = Auth::id()), dengan pengecekan kolom user_id pada tabel bookings.
4. Urutan Data
    - Booking diurutkan berdasarkan data terbaru (latest()).
5. Pencarian (Search)
    - Jika input pencarian tersedia (search), maka query akan mencari pada beberapa kolom: company_name, contact_person, phone, dan email.
6. Filter Status Pembayaran
    - Jika tersedia input payment_status, maka hanya booking dengan status tersebut yang ditampilkan.
7. Pagination
    - Hasil query dipaginasi dengan 10 data per halaman.
8. Render View
    - Menampilkan data ke view pages.pesanan dengan variabel bookings.

2. Function Menampilkan Pesanan tenant

```php
public function generateInvoice($id)
    {
        $booking = Booking::with('booth')->findOrFail($id);

        if ($booking->payment_status !== 'paid') {
            return redirect()->back()->with('error', 'Invoice hanya tersedia untuk pesanan yang sudah dibayar.');
        }

        $pdf = Pdf::loadView('invoice.booking', compact('booking'));
        return $pdf->download('invoice-' . $booking->order_id . '.pdf');
    }
```

Gambar 15. Funtion Menampilkan Pesanan Tenant

Fungsi ini digunakan untuk menampilkan daftar pemesanan booth (riwayat booking) khusus untuk pengguna tenant yang sedang login. Penjelasan Fungsional :

1. Relasi Booth
   - Mengambil data booking sekaligus data relasi booth-nya menggunakan with('booth') untuk menampilkan informasi booth yang dipesan.
2. Filter Berdasarkan User Login
   - Booking difilter berdasarkan user_id yang sesuai dengan Auth::id(), sehingga hanya booking milik tenant yang sedang login yang akan ditampilkan.
3. Urutan Data
   - Data booking diurutkan berdasarkan yang terbaru (latest()).
4. Pagination
   - Hasil data ditampilkan dengan sistem pagination, 10 data per halaman.
5. Tampilan View
   - Menampilkan data ke view tenant.my-bookings dengan variabel bookings.

3. Function Generate Invoice pdf

```php
public function generateInvoice($id)
    {
        $booking = Booking::with('booth')->findOrFail($id);

        if ($booking->payment_status !== 'paid') {
            return redirect()->back()->with('error', 'Invoice hanya tersedia untuk pesanan yang sudah dibayar.');
        }

        $pdf = Pdf::loadView('invoice.booking', compact('booking'));
        return $pdf->download('invoice-' . $booking->order_id . '.pdf');
    }
```

Gambar 16. Funtion Generate Invoice pdf

Fungsi ini digunakan untuk membuat dan mengunduh invoice dalam bentuk file PDF untuk booking booth yang sudah dibayar. Penjelasan Fungsional :

1. Ambil Data Booking dan Relasi Booth
   - Mengambil data booking berdasarkan id dan sekaligus memuat relasi booth menggunakan with('booth').
2. Validasi Status Pembayaran
   a. Mengecek apakah status pembayaran booking adalah 'paid'.

b. Jika belum dibayar, pengguna dikembalikan ke halaman sebelumnya dengan pesan error: "Invoice hanya tersedia untuk pesanan yang sudah dibayar."

3. Generate PDF Invoice
   - Menggunakan Pdf::loadView() untuk merender view invoice.booking menjadi PDF, dengan data booking yang dikirim ke dalamnya.
4. Unduh PDF
   - File invoice akan langsung diunduh dengan nama: invoice-[order_id].pdf.

7. Report Booking

1. Function Menampilkan Detail BookingTenant

```php
public function getBookingDetail(Booking $booking)
    {
        $authCheck = $this->checkBookingAuthorization($booking);
        if ($authCheck) return $authCheck;

        $booking->load('booth');

        return response()->json([
            'success' => true,
            'booking' => [
                'id' => $booking->id,
                'order_id' => $booking->order_id,
                'company_name' => $booking->company_name,
                'contact_person' => $booking->contact_person,
                'phone' => $booking->phone,
                'email' => $booking->email,
                'description' => $booking->description,
                'total_price' => $booking->total_price,
                'status' => $booking->status,
                'payment_status' => $booking->payment_status,
                'payment_method' => $booking->payment_method,
                'transaction_id' => $booking->transaction_id,
                'paid_at' => $booking->paid_at,
                'payment_expired_at' => $booking->payment_expired_at,
                'created_at' => $booking->created_at,
                'booth' => $booking->booth ? [
                    'booth_id' => $booking->booth->booth_id,
                    'booth_name' => $booking->booth->booth_name ?? $booking->booth->name,
                    'section' => $booking->booth->section,
                    'price' => $booking->booth->price,
                    'width' => $booking->booth->width,
                    'height' => $booking->booth->height,
                    'description' => $booking->booth->description
                ] : null
            ]
        ]);
    }
```

Gambar 17. Function Menampilkan Detail BookingTenant

Fungsi ini digunakan untuk mengambil detail lengkap dari sebuah booking, termasuk informasi booth yang terkait, dan hanya dapat diakses oleh pengguna yang berwenang. Penjelasan Fungsional :

1. Otorisasi Akses Booking
   - Menggunakan checkBookingAuthorization() untuk memastikan bahwa hanya user terkait (tenant) atau admin yang dapat melihat detail booking tersebut.
2. Memuat Relasi Booth
   - Memanggil relasi booth menggunakan $booking->load('booth') agar informasi booth ikut tersedia dalam response.
3. Response JSON Mengembalikan data booking lengkap dalam format JSON yang mencakup:
   a. Data tenant: nama perusahaan, kontak, email, dsb.
   b. Status: status booking, status pembayaran, metode pembayaran, ID transaksi.
   c. Timestamp: waktu pembayaran, waktu kedaluwarsa pembayaran, waktu pembuatan.
   d. Data booth: ID booth, nama, section, ukuran, dan harga.

2. Function Update Status Booking Tenant

```php
public function updateStatus(Request $request, Booking $booking)
    {
        $request->validate(['status' => 'required|in:pending,confirmed,cancelled']);

        $authCheck = $this->checkBookingAuthorization($booking);
        if ($authCheck) return $authCheck;

        try {
            $booking->update(['status' => $request->status]);

            if ($request->status == 'cancelled') {
                $booking->booth?->update(['status' => 'available']);
                $this->cancelMidtransTransaction($booking);
            }

            return response()->json(['success' => true, 'message' => 'Status booking berhasil diperbarui']);

        } catch (\Exception $e) {
            Log::error('Update Status Error: ' . $e->getMessage());
            return response()->json(['success' => false, 'message' => 'Gagal memperbarui status: ' . $e->getMessage()], 500);
        }
    }
```

Gambar 18. Function Update Status Booking Tenant

Fungsi ini digunakan untuk mengubah status pesanan (booking) seperti pending, confirmed, atau cancelled, serta menangani konsekuensi perubahan tersebut, khususnya saat dibatalkan. Penjelasan Fungsional :

1. Validasi Input
   - Hanya mengizinkan nilai status berupa: pending, confirmed, atau cancelled.
2. Otorisasi Akses
   - Memastikan hanya pengguna yang berhak (tenant terkait atau admin) yang bisa memperbarui status booking.
3. Update Status Booking
   - Mengubah status booking di database sesuai request.
4. Penanganan Pembatalan Booking Jika status diubah ke cancelled:
   - Booth yang terkait diubah kembali ke status available.
   - Transaksi di Midtrans dibatalkan melalui cancelMidtransTransaction().
5. Response JSON
   - Mengembalikan respons sukses jika proses berhasil, atau pesan kesalahan jika terjadi exception.

3. Function Report Booking

```php
public function index(Request $request)
    {
        $query = Booking::with('booth')->orderBy('created_at', 'desc');

        // Filter berdasarkan status (jika ada)
        if ($request->has('status') && $request->status !== null) {
            $query->where('status', $request->status);
        }

        // Filter berdasarkan tanggal mulai
        if ($request->has('start_date') && $request->start_date !== null) {
            $query->where('booking_date', '>=', $request->start_date);
        }

        // Filter berdasarkan tanggal akhir
        if ($request->has('end_date') && $request->end_date !== null) {
            $query->where('booking_date', '<=', $request->end_date);
        }

        // Ambil data dengan pagination
        $bookings = $query->paginate(10);

        return view('admin.report', compact('bookings'));
    }
```

Gambar 19. Function Report Booking

Fungsi ini digunakan untuk menampilkan data laporan booking di halaman admin dengan fitur filtering dan pagination. Penjelasan Fungsional :

1. Query Awal:
   - Mengambil data booking beserta relasi booth, diurutkan dari yang terbaru (created_at desc).
2. Filter Berdasarkan Status (Opsional):
   - Jika parameter status tersedia, maka data akan difilter berdasarkan status booking (pending, confirmed, dll).
3. Filter Berdasarkan Tanggal:
   a. start_date: Menampilkan booking yang dilakukan setelah atau sama dengan tanggal mulai.
   b. end_date: Menampilkan booking yang dilakukan sebelum atau sama dengan tanggal akhir.
4. Pagination:
   - Menampilkan data booking secara bertahap, 10 data per halaman.
5. Return View:
   - Data booking dikirim ke halaman admin.report untuk ditampilkan ke admin.

8. Chat Realtime

1. Function Menampilkan Daftar Tenant

```php
public function getTenantList()
    {
        $tenants = User::where('role_id', 2)->get(['id', 'name', 'email']);

        $tenantList = $tenants->map(function ($tenant) {
            $unreadCount = Chat::where('sender_id', $tenant->id)
                ->where('receiver_id', Auth::id())
                ->where('is_read', false)
                ->count();

            $lastMessage = Chat::betweenUsers(Auth::id(), $tenant->id)
                ->orderBy('created_at', 'desc')
                ->first();

            return [
                'id' => $tenant->id,
                'name' => $tenant->name,
                'email' => $tenant->email,
                'unread_count' => $unreadCount,
                'last_message' => $lastMessage ? $lastMessage->message : null,
                'last_message_time' => $lastMessage ? $lastMessage->created_at->format('H:i') : null
            ];
        });

        return response()->json($tenantList);
    }
```

Gambar 20. Function Menampilkan Daftar Tenant

Fungsi ini digunakan untuk mengambil daftar tenant (role_id = 2) beserta informasi tambahan untuk keperluan komunikasi/chat admin. Penjelasan Fungsional:

1. Ambil Data Tenant:
   - Mengambil data user dengan role_id = 2 (tenant), hanya id, name, dan email.
2. Loop Untuk setiap tenant:
   a. Hitung jumlah pesan belum dibaca dari tenant ke admin (unread_count).
   b. Ambil pesan terakhir antara admin dan tenant (jika ada), termasuk isi dan waktu kirim.
3. Hasil:
   a. Mengembalikan respons JSON berupa daftar tenant dengan ID, nama, email.
   b. Jumlah pesan belum dibaca (unread_count).
   c. Isi dan waktu pesan terakhir (last_message, last_message_time).

2. Function Notifikasi Chat untuk Admin

```php
public function getAdminNotifications()
    {
        // Only for admin users
        if (Auth::user()->role_id !== 1) {
            return response()->json(['error' => 'Unauthorized'], 403);
        }

        // Get recent unread messages from tenants to current admin
        $recentMessages = Chat::where('receiver_id', Auth::id())
            ->where('is_read', false)
            ->whereHas('sender', function ($query) {
                $query->where('role_id', 2); // Only from tenants
            })
            ->with('sender:id,name,email')
            ->orderBy('created_at', 'desc')
            ->take(5) // Last 5 unread messages
            ->get();

        $notifications = $recentMessages->map(function ($message) {
            return [
                'id' => $message->id,
                'sender_name' => $message->sender->name,
                'sender_email' => $message->sender->email,
                'message' => strlen($message->message) > 50
                    ? substr($message->message, 0, 50) . '...'
                    : $message->message,
                'time_ago' => $this->getTimeAgo($message->created_at),
                'created_at' => $message->created_at->format('Y-m-d H:i:s'),
                'sender_id' => $message->sender_id
            ];
        });

        return response()->json([
            'notifications' => $notifications,
            'total_unread' => $recentMessages->count()
        ]);
    }
```

Gambar 21. Function Notifikasi Chat untuk Admin

Fungsi ini mengambil notifikasi pesan terbaru dari tenant ke admin yang belum dibaca. Penjelasan Fungsional:

1. Akses Khusus Admin:

- Hanya user dengan role_id = 1 (admin) yang bisa mengakses fungsi ini. Jika bukan admin, akan ditolak (403 Unauthorized).

2. Ambil Pesan Belum Dibaca:

a. Mengambil 5 pesan terakhir yang belum dibaca oleh admin, yang dikirim oleh tenant (role_id = 2).

b. Menggunakan relasi sender untuk mengambil data pengirim (nama dan email).

3. Format Data Notifikasi Setiap notifikasi berisi:

a. ID pesan
b. Nama & email pengirim
c. Isi pesan yang dipotong (maks. 50 karakter)
d. Waktu dikirim dalam format time ago
e. Waktu asli created_at
f. ID pengirim

4. Hasil Akhir Mengembalikan data dalam format JSON berisi:

a. Array notifications (isi notifikasi)
b. Jumlah total pesan belum dibaca (total_unread).

3. Function Jumlah Pesan Belum Dibaca (Admin)

```php
public function getAdminUnreadCount()
    {
        // Only for admin users
        if (Auth::user()->role_id !== 1) {
            return response()->json(['error' => 'Unauthorized'], 403);
        }

        $count = Chat::where('receiver_id', Auth::id())
            ->where('is_read', false)
            ->whereHas('sender', function ($query) {
                $query->where('role_id', 2); // Only from tenants
            })
            ->count();

        return response()->json(['count' => $count]);
    }
```

Gambar 22. Function Jumlah Pesan Belum Dibaca (Admin)

Fungsi ini digunakan untuk menampilkan jumlah total pesan masuk dari tenant yang belum dibaca oleh admin — biasanya ditampilkan sebagai badge notifikasi di UI. Penjelasan Fungsional:

1. Akses Khusus Admin:

a. Hanya dapat diakses oleh user dengan role_id = 1 (admin).

b. Jika bukan admin, sistem mengembalikan status 403 Unauthorized.

c. Hitung Pesan Belum Dibaca:

- Menghitung jumlah pesan dari tenant (role_id = 2) yang ditujukan ke admin (receiver_id = Auth::id()), dan belum dibaca (is_read = false).

d. Output:

- Mengembalikan hasil dalam format JSON dengan key count, yang berisi total pesan belum dibaca.

4. Function kirim ke semua admin

```php
public function sendToAllAdmins(Request $request)
    {
        $request->validate([
            'message' => 'required|string|max:1000'
        ]);

        // Get all admins
        $admins = User::where('role_id', 1)->pluck('id');

        if ($admins->isEmpty()) {
            return response()->json([
                'success' => false,
                'message' => 'Tidak ada admin yang tersedia'
            ], 404);
        }

        $chatIds = [];

        // Send message to all admins
        foreach ($admins as $adminId) {
            $chat = Chat::create([
                'sender_id' => Auth::id(),
                'receiver_id' => $adminId,
                'message' => $request->message,
                'is_read' => false
            ]);

            $chatIds[] = $chat->id;
        }

        return response()->json([
            'success' => true,
            'message' => 'Pesan berhasil dikirim',
            'data' => [
                'message' => $request->message,
                'sender_id' => Auth::id(),
                'is_sent' => true,
                'time' => now()->format('H:i'),
                'date' => now()->format('d/m/Y')
            ]
        ]);
    }
```

Gambar 23. Function kirim ke semua admin

Fungsi ini memungkinkan pengguna dengan role tenant untuk mengirim satu pesan yang sama ke seluruh admin yang terdaftar di sistem. Penjelasan Fungsional:

1. Validasi Pesan:
   - Memastikan input message wajib diisi, bertipe string, dan maksimal 1000 karakter.
2. Ambil Daftar Admin:
   - Mengambil semua user dengan role_id = 1 (admin), hanya id-nya saja.
3. Cek Ketersediaan Admin:
   - Jika tidak ada admin terdaftar, fungsi akan menghentikan proses dan mengembalikan pesan error.
4. Pengiriman Pesan:
   a. Melakukan loop untuk mengirim pesan yang sama ke setiap admin dengan mencatat sender_id, receiver_id, dan message.
   b. Menandai pesan sebagai is_read = false secara default.
5. Respons Output:
   - Mengembalikan JSON berisi status keberhasilan, isi pesan, pengirim, dan waktu pengiriman.

5. Function Mendapatkan Pesan Konsolidasi Tenant

```php
public function getTenantMessages()
    {
        // Get all admin IDs
        $adminIds = User::where('role_id', 1)->pluck('id');

        // Get unique messages using DISTINCT on message content, timestamp, and sender
        $messages = Chat::where(function ($query) use ($adminIds) {
            // Messages sent by tenant to any admin
            $query->where('sender_id', Auth::id())
                ->whereIn('receiver_id', $adminIds);
        })
            ->orWhere(function ($query) use ($adminIds) {
                // Messages sent by any admin to tenant
                $query->whereIn('sender_id', $adminIds)
                    ->where('receiver_id', Auth::id());
            })
            ->orderBy('created_at', 'asc')
            ->with('sender:id,name')
            ->get(['id', 'sender_id', 'receiver_id', 'message', 'created_at', 'is_read']);

        // Remove duplicate messages sent by tenant (keep only one copy of each message)
        $uniqueMessages = collect();
        $processedTenantMessages = [];

        foreach ($messages as $message) {
            if ($message->sender_id == Auth::id()) {
                // For tenant messages, only keep one copy based on message content and timestamp
                $messageKey = $message->message . '|' . $message->created_at->format('Y-m-d H:i:s');

                if (!in_array($messageKey, $processedTenantMessages)) {
                    $uniqueMessages->push($message);
                    $processedTenantMessages[] = $messageKey;
                }
            } else {
                // For admin messages, keep all
                $uniqueMessages->push($message);
            }
        }

        // Mark messages from admins as read
        Chat::whereIn('sender_id', $adminIds)
            ->where('receiver_id', Auth::id())
            ->where('is_read', false)
            ->update(['is_read' => true]);

        $formattedMessages = $uniqueMessages->map(function ($message) {
            $isFromAdmin = in_array($message->sender_id, User::where('role_id', 1)->pluck('id')->toArray());

            return [
                'id' => $message->id,
                'message' => $message->message,
                'sender_id' => $message->sender_id,
                'receiver_id' => $message->receiver_id,
                'is_sent' => $message->sender_id == Auth::id(),
                'sender_name' => $isFromAdmin ? $message->sender->name : 'Anda',
                'time' => $message->created_at->format('H:i'),
                'date' => $message->created_at->format('d/m/Y')
            ];
        });

        return response()->json($formattedMessages);
    }
```

Gambar 24. Function Mendapatkan Pesan Konsolidasi Tenant

1. Fungsi ini mengambil dan menampilkan seluruh percakapan antara tenant dan admin, baik pesan masuk maupun keluar.
2. Ringkasan Proses:
3. Ambil semua pesan antara tenant dan admin.
4. Hapus duplikasi pesan tenant yang dikirim ke banyak admin.
5. Tandai pesan dari admin sebagai telah dibaca.
6. Format pesan ke bentuk yang siap ditampilkan di frontend (nama pengirim, waktu, dll).

6. Function Mengirim Pesan (Admin ke Tenant)

```php
public function sendMessage(Request $request)
{
    $request->validate([
        'receiver_id' => 'required|exists:users,id',
        'message' => 'required|string|max:1000'
    ]);

    $chat = Chat::create([
        'sender_id' => Auth::id(),
        'receiver_id' => $request->receiver_id,
        'message' => $request->message,
        'is_read' => false
    ]);

    return response()->json([
        'success' => true,
        'message' => 'Pesan berhasil dikirim',
        'data' => [
            'id' => $chat->id,
            'message' => $chat->message,
            'sender_id' => $chat->sender_id,
            'receiver_id' => $chat->receiver_id,
            'is_sent' => true,
            'time' => $chat->created_at->format('H:i'),
            'date' => $chat->created_at->format('d/m/Y')
        ]
    ]);
}
```

Gambar 25. Function Mengirim Pesan (Admin ke Tenant)

Fungsi ini menangani pengiriman pesan langsung dari admin ke tenant melalui sistem chat internal. Hanya pengguna yang sudah login dan memiliki hak akses (sebagai admin) yang dapat memanggil fungsi ini. Alur Proses:

1. Validasi Input Fungsi memvalidasi dua input penting:
   a. receiver_id: ID penerima pesan yang wajib ada dan harus cocok dengan ID yang ada di tabel users.
   b. message: Isi pesan yang wajib diisi dan memiliki panjang maksimum 1000 karakter.
   c. Jika validasi gagal, Laravel otomatis akan menampilkan pesan kesalahan.
2. Penyimpanan Pesan Jika validasi berhasil, maka data pesan akan disimpan ke dalam tabel chats dengan informasi:

a. sender_id: ID pengirim pesan, diambil dari user yang sedang login (dalam hal ini admin).
b. receiver_id: ID tenant tujuan.
c. message: Isi dari pesan.
d. is_read: Diset ke false karena pesan baru dikirim dan belum dibaca penerima.

3. Response JSON Setelah pesan berhasil disimpan, sistem akan merespons permintaan dengan data JSON, berisi:

a. Status keberhasilan (success: true)
b. Isi pesan
c. ID pengirim dan penerima
d. Waktu dan tanggal pengiriman dalam format yang mudah dibaca (H:i dan d/m/Y)
e. Data ini berguna untuk menampilkan pesan real-time di tampilan frontend chat.

7. Function untuk Mendapatkan Jumlah Pesan Belum Dibaca

```php
public function getUnreadCount()
    {
        $count = Chat::where('receiver_id', Auth::id())
            ->where('is_read', false)
            ->count();

        return response()->json(['count' => $count]);
    }
```

Gambar 26. Function untuk Mendapatkan Jumlah Pesan Belum Dibaca

Fungsi ini digunakan untuk menghitung jumlah pesan yang belum dibaca oleh pengguna yang sedang login. Sistem memfilter data di tabel chats berdasarkan receiver_id dan is_read = false, lalu mengembalikannya dalam format JSON.

8. Function untuk Menandai Notifikasi Sebagai Dibaca

```php
public function markNotificationAsRead($messageId)
 {
     $message = Chat::where('id', $messageId)
         ->where('receiver_id', Auth::id())
         ->first();

     if ($message) {
         $message->update(['is_read' => true]);
         return response()->json(['success' => true]);
     }

     return response()->json(['error' => 'Message not found'], 404);
 }
```

Gambar 27. Function untuk Menandai Notifikasi Sebagai Dibaca

Fungsi ini bertugas untuk menandai pesan notifikasi sebagai sudah dibaca oleh pengguna yang sedang login. Fitur ini mendukung sistem notifikasi berbasis chat di aplikasi, khususnya untuk menunjukkan bahwa pesan tertentu telah dilihat oleh penerimanya.Penjelasan Rinci ada di bawah ini:

1. Pencarian Pesan:
   a. Fungsi mencari pesan berdasarkan id pesan ($messageId) dan memastikan bahwa pengguna yang sedang login (Auth::id()) adalah penerima pesan (receiver_id).
   b. Ini mencegah pengguna lain mengakses atau mengubah pesan yang bukan ditujukan kepadanya.
2. Update Status Pesan:
   a. Jika pesan ditemukan, kolom is_read akan diperbarui menjadi true, menandakan bahwa pesan tersebut sudah dibaca.
   b. Sistem kemudian mengembalikan respons JSON: { "success": true }.
3. Jika Pesan Tidak Ditemukan:
   - Fungsi mengembalikan respon error dengan status HTTP 404 dan pesan: "Message not found".

9. Function untuk Menandai Semua Pesan dari Pengirim Sebagai Dibaca

```php
public function markAllFromSenderAsRead($senderId)
    {
        Chat::where('sender_id', $senderId)
            ->where('receiver_id', Auth::id())
            ->where('is_read', false)
            ->update(attributes: ['is_read' => true]);

        return response()->json(['success' => true]);
    }
```

Gambar 28. Function untuk Menandai Semua Pesan dari Pengirim Sebagai Dibaca

Fungsi markAllFromSenderAsRead($senderId) menandai semua pesan yang belum dibaca dari pengirim tertentu sebagai sudah dibaca oleh pengguna yang sedang login. Pesan yang memenuhi syarat (sender_id, receiver_id, dan is_read = false) akan diperbarui sekaligus, lalu sistem mengembalikan respons sukses dalam format JSON.

10. Function untuk Format Waktu Relatif

```php
private function getTimeAgo($datetime)
    {
        $time = time() - strtotime($datetime);

        if ($time < 60) {
            return 'Baru saja';
        } elseif ($time < 3600) {
            $minutes = floor($time / 60);
            return $minutes . ' menit yang lalu';
        } elseif ($time < 86400) {
            $hours = floor($time / 3600);
            return $hours . ' jam yang lalu';
        } else {
            $days = floor($time / 86400);
            return $days . ' hari yang lalu';
        }
    }
```

Gambar 29. Function untuk Format Waktu Relatif

Fungsi getTimeAgo($datetime) adalah helper yang digunakan untuk mengubah format waktu menjadi bentuk yang lebih mudah dipahami oleh pengguna, seperti "Baru saja", "5 menit yang lalu", atau "2 hari yang lalu". Fungsi ini menghitung selisih waktu saat ini dengan waktu yang diberikan ($datetime) dalam detik, lalu menentukan:

a. Jika selisihnya kurang dari 60 detik, maka ditampilkan "Baru saja".
b. Jika kurang dari 1 jam, maka dihitung berapa menit yang telah berlalu.
c. Jika kurang dari 24 jam, maka dihitung berapa jam yang telah berlalu.
d. Jika lebih dari 24 jam, maka ditampilkan dalam bentuk jumlah hari.

Fungsi ini bermanfaat untuk menampilkan waktu pesan atau notifikasi dalam antarmuka pengguna agar lebih ramah dan informatif.

Lampiran 6. Sertifikat HKI

REPUBLIK INDONESIA
KEMENTERIAN HUKUM

SURAT PENCATATAN CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC00202507781 3, 28 Juni 2025
Pencipta		
Nama	:	**Muhammad Faqih Maulana, Ir. Gunanjar Wiro Sasmito, S.Kom., M.Kom. dkk**
Alamat	:	Jl. Mbah Bregas No.40 RT 02/RW 09 Desa Sidakaton, Dukuhturi, Kab. Tegal, Jawa Tengah, 52192
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**Pusat Penelitian dan Pengabdian Masyarakat (P3M) Politeknik Harapan Bersama**
Alamat	:	Jalan Mataram No. 9, Pesurungan Lor, Kecamatan Margadana, Margadana, Kota Tegal, Jawa Tengah, 52142
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Program Komputer**
Judul Ciptaan	:	**Rancang Bangun Aplikasi Booking Booth Tenant pada Cresindo Event Organizer Menggunakan Algoritma Greedy**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	28 Juni 2025, di Kota Tegal
Jangka waktu pelindungan	:	Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.
Nomor Pencatatan	:	000918074

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n. MENTERI HUKUM
DIREKTUR JENDERAL KEKAYAAN INTELEKTUAL
u.b
Direktur Hak Cipta dan Desain Industri

Agung Damarsasongko,SH.,MH.
NIP. 196912261994031001

Disclaimer:
1. Dalam hal pemohon memberikan keterangan tidak sesuai dengan surat pernyataan, Menteri berwenang untuk mencabut surat pencatatan permohonan.
2. Surat Pencatatan ini telah ditandatangani secara elektronik menggunakan segel elektronik yang diterbitkan oleh Balai Besar Sertifikasi Elektronik, Badan Siber dan Sandi Negara.
3. Surat Pencatatan ini dapat dibuktikan keasliannya dengan memindai kode QR pada dokumen ini dan informasi akan ditampilkan dalam browser.

LAMPIRAN PENCIPTA

No	Nama	Alamat
1	Muhammad Faqih Maulana	Jl. Mbah Bregas No.40 RT 02/RW 09 Desa Sidakaton. Dukuhturi, Kab. Tegal
2	Ir. Ginanjar Wiro Sasmito, S.Kom., M.Kom.	Jln. Raya Kluwut Timur No. 24 Bulakamba, Kab. Brebes
3	M. Nishom, S.Kom., M.Kom.	Jl. Jepara, Perum Griya Putri Land Blok A6, RT 03/ RW 04 Margadana, Kota Tegal