Laporan Implementasi Akhir
Proyek Basis Data "Eventify"

Disusun oleh Kelompok 9:

1. Rafael Mahardika Arya Dewamurti (24/536279/PA/22755)
2. Bobby Rahman Hartanto (24/539383/PA/22903)

DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA INSTRUMENTASI
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2025

BAB I PENDAHULUAN

1.1 Latar Belakang

Dalam ekosistem kampus yang dinamis, manajemen kegiatan kemahasiswaan memegang peranan vital sebagai wadah pengembangan diri. Namun, pengamatan di lapangan menunjukkan bahwa pengelolaan informasi seminar, lomba, festival, maupun kegiatan organisasi lainnya masih dilakukan secara terfragmentasi. Informasi tersebar di berbagai kanal media sosial yang tidak terpusat (seperti Instagram, WhatsApp, atau papan pengumuman fisik), yang seringkali menyebabkan asimetri informasi di kalangan mahasiswa.

Akibatnya, banyak mahasiswa melewatkan kesempatan berharga karena tidak mengetahui jadwal acara tepat waktu. Di sisi lain, penyelenggara acara (Organizer) menghadapi kesulitan administratif karena proses pendaftaran peserta masih dilakukan secara manual, berulang-ulang, dan rentan terhadap kesalahan pencatatan (*human error*). Ketiadaan platform terpusat juga menyulitkan proses evaluasi pasca-acara karena umpan balik peserta tidak terdokumentasi dengan baik.

1.2 Tujuan Pengembangan

Proyek "Eventify" dikembangkan sebagai solusi teknologi terintegrasi untuk menjawab permasalahan tersebut. Tujuan utama dari pengembangan sistem ini adalah membangun sebuah platform berbasis web yang mampu:

1. **Sentralisasi Informasi:** Menyediakan satu pintu akses untuk seluruh informasi kegiatan kampus (*Single Source of Truth*).
2. **Efisiensi Administratif:** Mengotomatisasi proses pendaftaran acara (*Transaction Creation*) sehingga mengurangi beban kerja penyelenggara.
3. **Peningkatan Kualitas Acara:** Menyediakan fitur ulasan (*Feedback Loop*) yang terverifikasi untuk membantu penyelenggara meningkatkan kualitas kegiatan di masa depan.

1.3 Ruang Lingkup Implementasi

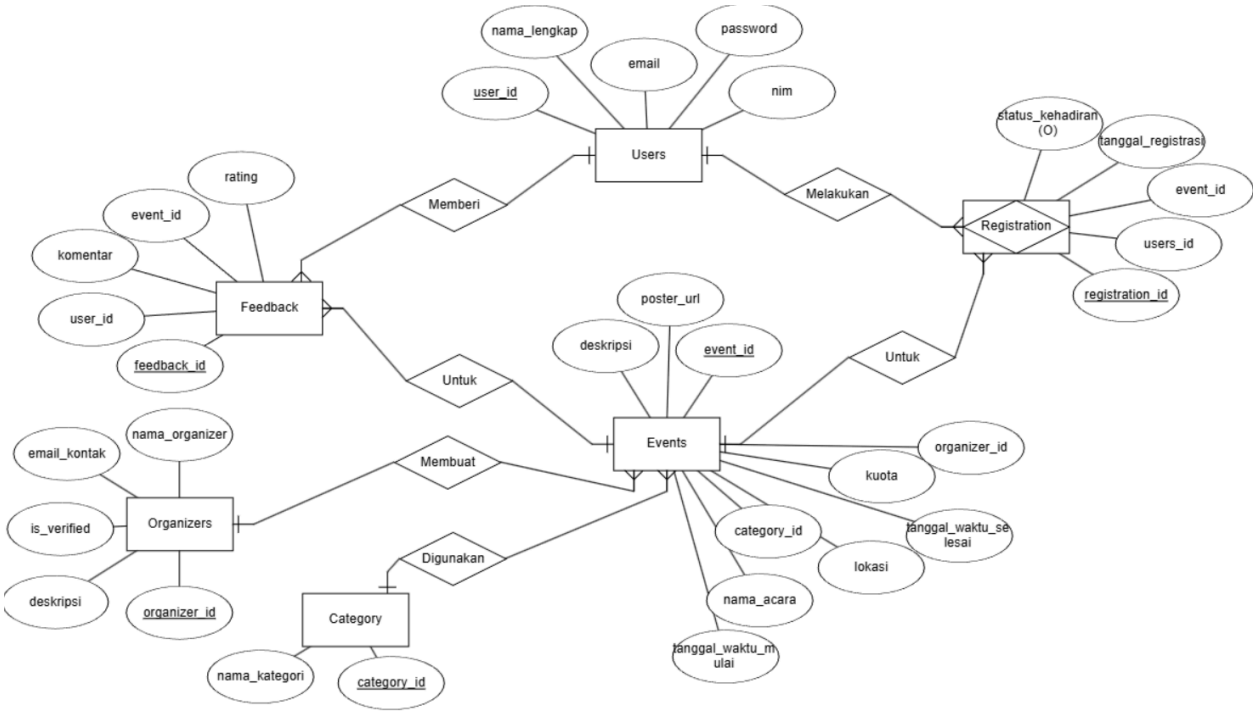
Laporan ini merangkum hasil implementasi tahap akhir (Final Integration) yang mencakup pengembangan penuh dari sisi *Backend*, *Frontend*, hingga *Database*. Pada tahap ini, fokus utama bukan hanya pada fungsi dasar CRUD (*Create, Read, Update, Delete*), melainkan pada penerapan logika bisnis yang kompleks untuk menjaga integritas sistem, antara lain:

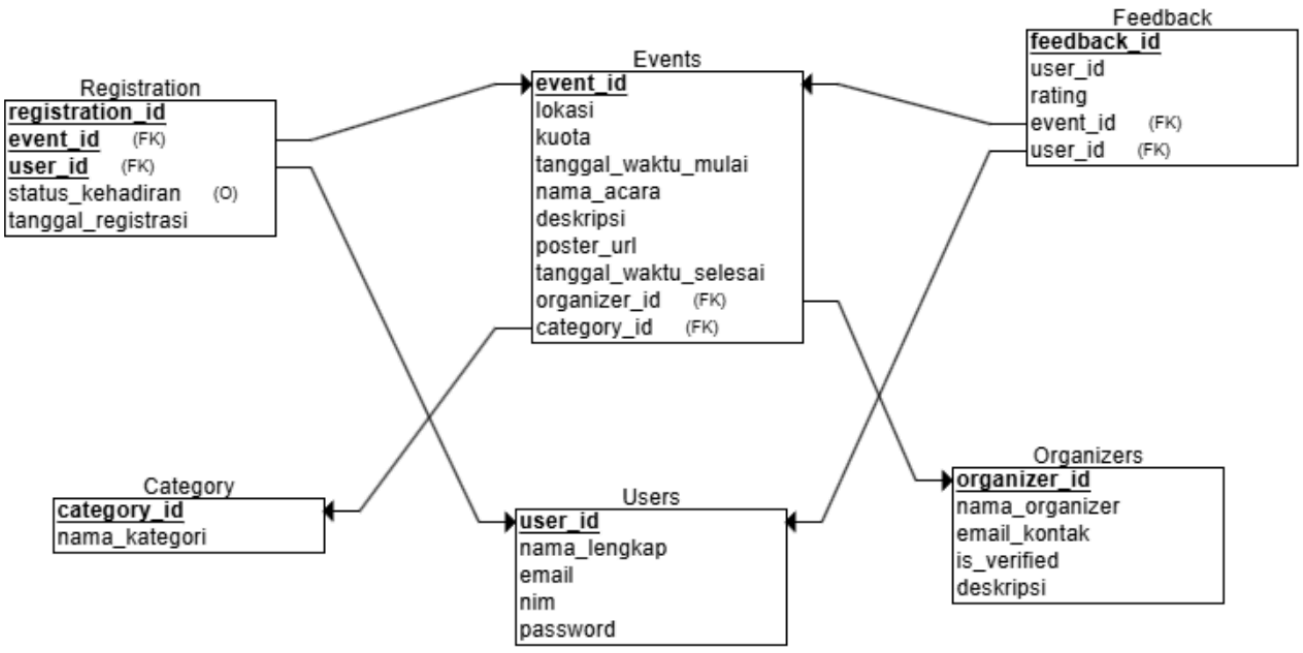
- Integrasi data secara *real-time* antara antarmuka pengguna dan server database.
- Validasi pendaftaran tunggal (*One User One Event*) untuk mencegah duplikasi peserta.
- Mekanisme ulasan tertutup yang hanya dapat diakses oleh peserta yang telah terdaftar.

BAB II PERANCANGAN DAN IMPLEMENTASI DATABASE

2.1 Visualisasi ERD dan Skema Relasional Logis (LRS)

Gambar di bawah ini adalah ER-Diagram dan *Logical Relational Schema* (LRS) yang menjadi jembatan visual antara desain ERD konseptual (Week 1) dan implementasi kode SQL (Week 2).





LRS ini mengkonfirmasi struktur tabel yang berasal dari ERD Week 1. Penjelasan utama dari skema ini adalah:

- **Tabel Induk (Master):** Tabel Users, Organizers, dan Categories ditampilkan sebagai tabel independen yang menyimpan data master. Kolom PRIMARY KEY (PK) mereka (seperti user_id, organizer_id, category_id) akan menjadi referensi bagi tabel lain.
- **Tabel Transaksi (Events):** Tabel Events adalah pusat dari skema, yang terhubung ke Organizers dan Categories. Garis-garis relasi di LRS menunjukkan bahwa kolom organizer_id dan category_id di dalam Events adalah *Foreign Key* (FK).
- **Tabel Junction (M:N):** Relasi *Many-to-Many* (M:N) diimplementasikan melalui dua tabel penghubung:
    1. Registrations: Menghubungkan Users dan Events.
    2. Feedback: Juga menghubungkan Users dan Events.

Setiap garis di LRS ini diterjemahkan langsung menjadi sintaks FOREIGN KEY (...) REFERENCES (...) di dalam file schema.sql.

2.2 Analisis Sintaks SQL per Tabel

Skema database ini terdiri dari enam tabel. Urutan pembuatan tabel sangat penting untuk memenuhi ketergantungan relasional (*referential integrity*). Tabel tanpa *Foreign Key* (induk) harus dibuat terlebih dahulu.

Urutan Pembuatan:

1. Users, Organizers, Categories (Tabel Induk)
2. Events (Tabel Anak, bergantung pada Organizers dan Categories)
3. Registrations, Feedback (Tabel Anak, bergantung pada Users dan Events)

A. Tabel Users

Tabel ini menyimpan data fundamental mahasiswa atau peserta. Sintaks CREATE TABLE Users (...) mendefinisikan kolom-kolom berikut:

- user_id INT AUTO_INCREMENT PRIMARY KEY: Kolom ini berfungsi sebagai pengidentifikasi unik (Primary Key) untuk setiap pengguna. Penggunaan AUTO_INCREMENT sangat membantu karena membebaskan aplikasi dari tugas mengelola ID; database secara otomatis akan memberikan ID unik (1, 2, 3, ...) untuk setiap pengguna baru.
- nama_lengkap VARCHAR(255) NOT NULL: Kolom teks dengan batas 255 karakter. Diberi batasan NOT NULL untuk memastikan nama pengguna wajib diisi.
- email VARCHAR(255) NOT NULL UNIQUE: Selain NOT NULL, batasan UNIQUE ditambahkan di sini sebagai kunci integritas bisnis yang krusial. Ini mencegah dua

pengguna mendaftar dengan alamat email yang sama.

- password VARCHAR(255) NOT NULL: Kolom ini akan menyimpan *hash* dari kata sandi, bukan teks aslinya, untuk menjaga keamanan data.
- nim VARCHAR(50) UNIQUE: Mirip dengan email, UNIQUE diterapkan untuk memastikan tidak ada duplikasi data mahasiswa berdasarkan NIM.

B. Tabel Organizers dan Categories

Dua tabel ini adalah tabel *master* atau *lookup* yang berfungsi untuk menyediakan data bagi tabel Events. Desain ini penting untuk normalisasi (3NF), karena detail penyelenggara atau kategori tidak bergantung pada acara, melainkan sebaliknya.

- organizer_id dan category_id keduanya diatur sebagai INT AUTO_INCREMENT PRIMARY KEY.
- Pada Organizers, kolom is_verified BOOLEAN diberi nilai DEFAULT false. Ini adalah pengaturan yang logis, di mana setiap penyelenggara baru yang mendaftar akan otomatis dianggap "belum terverifikasi" sampai disetujui oleh Administrator.
- Pada Categories, kolom nama_kategori juga disetel UNIQUE untuk mencegah duplikasi data master (misalnya, mencegah seseorang memasukkan "Seminar" dua kali).

C. Tabel Events

Ini adalah tabel inti dari aplikasi yang menyimpan semua data acara. Tabel ini memiliki ketergantungan (Foreign Key) pada Organizers dan Categories.

- tanggal_waktu_mulai DATETIME NOT NULL: Penggunaan tipe data DATETIME sangat penting untuk penjadwalan. Kolom ini diwajibkan (NOT NULL).
- tanggal_waktu_selesai DATETIME: Kolom ini **tidak** memiliki NOT NULL. Ini adalah keputusan desain yang disengaja. Ini mengizinkan penyelenggara untuk membuat "draft" acara di sistem meskipun waktu selesainya belum final atau belum ditentukan.
- kuota INT DEFAULT 0: Jika penyelenggara tidak mengisi kuota, sistem akan otomatis mengaturnya ke 0 (nol), bukan NULL, untuk menghindari ambiguitas dalam perhitungan.

D. Tabel Registrations dan Feedback (Tabel Junction)

Kedua tabel ini merupakan implementasi teknis dari relasi *Many-to-Many* (M:N) yang menghubungkan entitas Users dengan Events. Tabel ini tidak hanya mencatat relasi, tetapi juga menerapkan *Business Logic* yang ketat melalui *constraints* database.

Berikut adalah spesifikasi aturan yang diterapkan untuk menjaga integritas data:

- **Pencegahan Duplikasi (Logika 1 User 1 Event):** Kami menerapkan batasan **UNIQUE KEY (user_id, event_id)** pada kedua tabel.
    - Pada **Registrations**, ini memastikan seorang mahasiswa hanya bisa mendaftar satu

kali untuk satu acara yang sama.

- Pada **Feedback**, ini mencegah *spamming* ulasan, di mana satu user hanya diizinkan memberikan satu ulasan per acara. Database akan menolak input ganda secara otomatis.

- **Pembersihan Data Otomatis (Referential Integrity):** Kedua tabel menggunakan aturan **ON DELETE CASCADE** pada *Foreign Key*. Artinya, jika akun seorang User dihapus (misal lulus) atau sebuah Event dibatalkan dan dihapus oleh admin, maka seluruh data pendaftaran dan ulasan yang terkait dengan user/event tersebut akan **ikut terhapus secara otomatis**. Ini menjaga database tetap bersih dari data "yatim" (*orphaned records*) yang tidak memiliki induk.
- **Validasi Nilai Input:** Pada tabel Feedback, terdapat batasan **CHECK (rating >= 1 AND rating <= 5)**. Ini adalah validasi level database yang menjamin integritas data penilaian. Jika aplikasi (Frontend/Backend) mencoba mengirimkan nilai rating di luar angka 1 sampai 5, database akan menolak transaksi tersebut.
- **Pencatatan Waktu Otomatis:** Kolom tanggal_registrasi (di tabel Registrations) dan created_at (di tabel Feedback) menggunakan nilai default **DEFAULT CURRENT_TIMESTAMP**. Fitur ini memungkinkan database mencatat waktu server secara presisi saat transaksi terjadi tanpa perlu input manual dari sisi aplikasi.

E. Analisis Relasi dan Foreign Key (Aksi ON DELETE)

Bagian terpenting dari implementasi skema adalah mendefinisikan bagaimana tabel-tabel saling terhubung dan apa yang terjadi jika data induk dihapus.

Sintaks: FOREIGN KEY (kolom_lokal) REFERENCES TabelInduk(kolom_induk) [aksi]

Dalam desain ini, kami menggunakan dua strategi aksi ON DELETE yang berbeda:

1. **ON DELETE SET NULL** (pada tabel Events)
   - **Logika:** Aksi ini diterapkan pada organizer_id dan category_id. Jika sebuah Organizer (UKM) dibubarkan dan dihapus dari database, apa yang terjadi pada Events yang mereka buat?
   - **Keuntungan:** Dengan SET NULL, acara tersebut **tidak ikut terhapus**. Kolom organizer_id-nya hanya akan diatur menjadi NULL. Ini adalah keputusan desain untuk menjaga data historis acara. Acara tersebut menjadi "yatim" (orphaned), tapi datanya tetap ada untuk arsip.
2. **ON DELETE CASCADE** (pada Registrations dan Feedback)
   - **Logika:** Aksi ini diterapkan pada event_id dan user_id. Jika sebuah Event dibatalkan permanen dan dihapus dari database, semua data pendaftaran (Registrations) dan ulasan (Feedback) untuk acara itu sudah tidak relevan lagi.
   - **Keuntungan:** CASCADE (Beruntun) berarti "jika induknya dihapus, hapus juga

semua data anaknya." Ini secara otomatis membersihkan database dari data "sampah" yang sudah tidak memiliki induk, menjaga database tetap bersih, efisien, dan relevan.

2.3 Verifikasi Normalisasi (3NF)

Skema SQL ini divalidasi berdasarkan aturan normalisasi yang disyaratkan:

1. **1NF:** Terpenuhi. Semua kolom bersifat atomik (tidak ada *multi-valued attributes*).
2. **2NF:** Terpenuhi. Semua atribut non-kunci (seperti nama_acara, tanggal_registrasi) bergantung pada keseluruhan Primary Key.
3. **3NF:** Terpenuhi. Tidak ada *transitive dependency*. Atribut yang bergantung pada atribut non-kunci lain telah dipisah.
    - **Contoh:** nama_organizer bergantung pada organizer_id (PK-nya), bukan pada event_id. Oleh karena itu, kita membuat tabel Organizers. nama_kategori bergantung pada category_id (PK-nya), bukan pada event_id. Oleh karena itu, kita membuat tabel Categories.

BAB III IMPLEMENTASI SISTEM BACKEND (SERVER-SIDE)

Sisi server (*Backend*) dibangun menggunakan lingkungan eksekusi **Node.js** dengan kerangka kerja **Express.js**. Peran backend di sini sangat krusial sebagai "otak" yang memproses logika bisnis sebelum data disimpan atau diambil dari database.

3.1 Arsitektur Aplikasi (3-Tier Architecture)

Dalam pengembangan sistem "Eventify", kami mengadopsi standar industri arsitektur perangkat lunak yang dikenal sebagai *Three-Tier Architecture* atau Arsitektur Tiga Lapis. Konsep ini membagi sistem aplikasi menjadi tiga lapisan logis dan fisik yang terpisah namun saling terintegrasi. Pendekatan ini dipilih untuk memastikan modularitas sistem, di mana setiap lapisan memiliki tanggung jawab spesifik dan independen. Hal ini sangat krusial untuk menjaga keamanan data, karena lapisan antarmuka tidak diizinkan mengakses lapisan data secara langsung tanpa melalui lapisan logika sebagai perantara.

Secara spesifik, implementasi ketiga lapisan tersebut dalam proyek ini adalah sebagai berikut:

1. **Presentation Tier (Frontend):** Merupakan lapisan terluar yang berinteraksi langsung dengan pengguna (*Client-Side*). Dibangun menggunakan HTML, CSS (Tailwind), dan JavaScript, lapisan ini bertanggung jawab untuk menyajikan antarmuka visual (UI) yang responsif dan menangkap input pengguna. Lapisan ini bersifat "agnostik" terhadap database, artinya ia tidak mengetahui struktur penyimpanan data dan hanya berkomunikasi melalui API.
2. **Logic Tier (Backend API):** Berperan sebagai "otak" atau jembatan pemroses (*Server-Side*) yang dibangun menggunakan Node.js dan Express. Lapisan ini menangani seluruh logika bisnis (*Business Logic*), validasi data (misalnya: memastikan input rating 1-5), otentikasi pengguna, serta menerjemahkan permintaan dari Frontend menjadi instruksi yang dimengerti oleh Database.
3. **Data Tier (MySQL Database):** Merupakan lapisan terdalam yang berfungsi sebagai repositori penyimpanan data persisten (*Storage*). Lapisan ini bertugas menyimpan, mengambil, dan menjaga integritas data relasional menggunakan MySQL, serta menerapkan aturan integritas referensial (seperti *Unique Constraints* dan *Foreign Keys*).

Penerapan arsitektur ini memberikan keuntungan signifikan dalam siklus pengembangan perangkat lunak. Dengan pemisahan tanggung jawab yang tegas, pemeliharaan kode (*maintainability*) menjadi lebih mudah dan terorganisir. Selain itu, arsitektur ini memungkinkan pengembangan fitur dilakukan secara paralel; misalnya, perbaikan pada desain antarmuka di Frontend dapat dilakukan tanpa risiko mengganggu logika bisnis yang kompleks di Backend

ataupun merusak struktur data yang tersimpan di Database.

3.2 Logika "Smart Query" (Fitur Unggulan)

Salah satu tantangan teknis terbesar dalam proyek ini adalah bagaimana memberitahu Frontend mengenai status pengguna terhadap suatu event secara efisien. Apakah pengguna sudah mendaftar? Apakah sudah memberi ulasan?

Untuk menjawab ini tanpa melakukan *multiple request*, kami mengimplementasikan teknik *Subquery* pada endpoint GET /api/events. Berikut adalah logika query yang kami bangun dalam backend.js:

```
app.get('/api/events', async (req, res) => {
    const { userId } = req.query;

    // Query cerdas yang melakukan pengecekan status secara real-time
    let query = `
        SELECT e.*, c.nama_kategori, o.nama_organizer,
        -- Subquery 1: Cek apakah user ini sudah terdaftar di event X? (Return 1/0)
        (SELECT COUNT(*) FROM Registrations r WHERE r.event_id = e.event_id AND r.user_id = ?) AS is_registered,

        -- Subquery 2: Cek apakah user ini sudah memberi ulasan di event X? (Return 1/0)
        (SELECT COUNT(*) FROM Feedback f WHERE f.event_id = e.event_id AND f.user_id = ?) AS has_reviewed

        FROM Events e
        LEFT JOIN Categories c ON e.category_id = c.category_id
        LEFT JOIN Organizers o ON e.organizer_id = o.organizer_id
    `;
    // ... eksekusi query dengan parameter userId
});
```

Dengan pendekatan ini, dalam satu kali pengambilan data, Backend langsung melampirkan "status" personal pengguna (is_registered dan has_reviewed). Ini membuat aplikasi sangat responsif dan hemat *bandwidth* karena Frontend tidak perlu bertanya berkali-kali ke server untuk mengecek status setiap tombol.

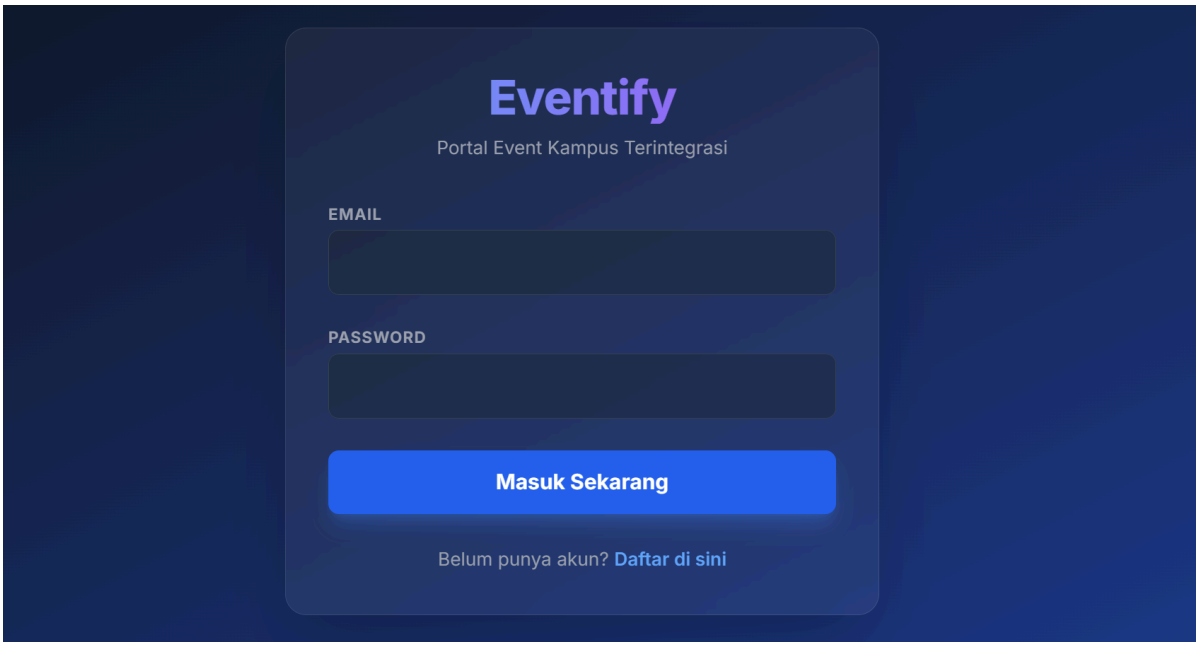
BAB IV IMPLEMENTASI FRONTEND (UI/UX)

Sisi klien (*Frontend*) bertugas menerjemahkan data JSON dari Backend menjadi antarmuka visual yang interaktif dan mudah dipahami. Kami menggunakan **HTML5** dan **TailwindCSS** untuk *styling*, serta **Vanilla JavaScript** untuk logika interaksi.

4.1 *Results UI Implementation*

Berikut adalah kumpulan screenshot dari antarmuka aplikasi yang telah dikembangkan. Setiap halaman ditampilkan untuk memberikan gambaran visual mengenai alur penggunaan serta fitur yang tersedia.

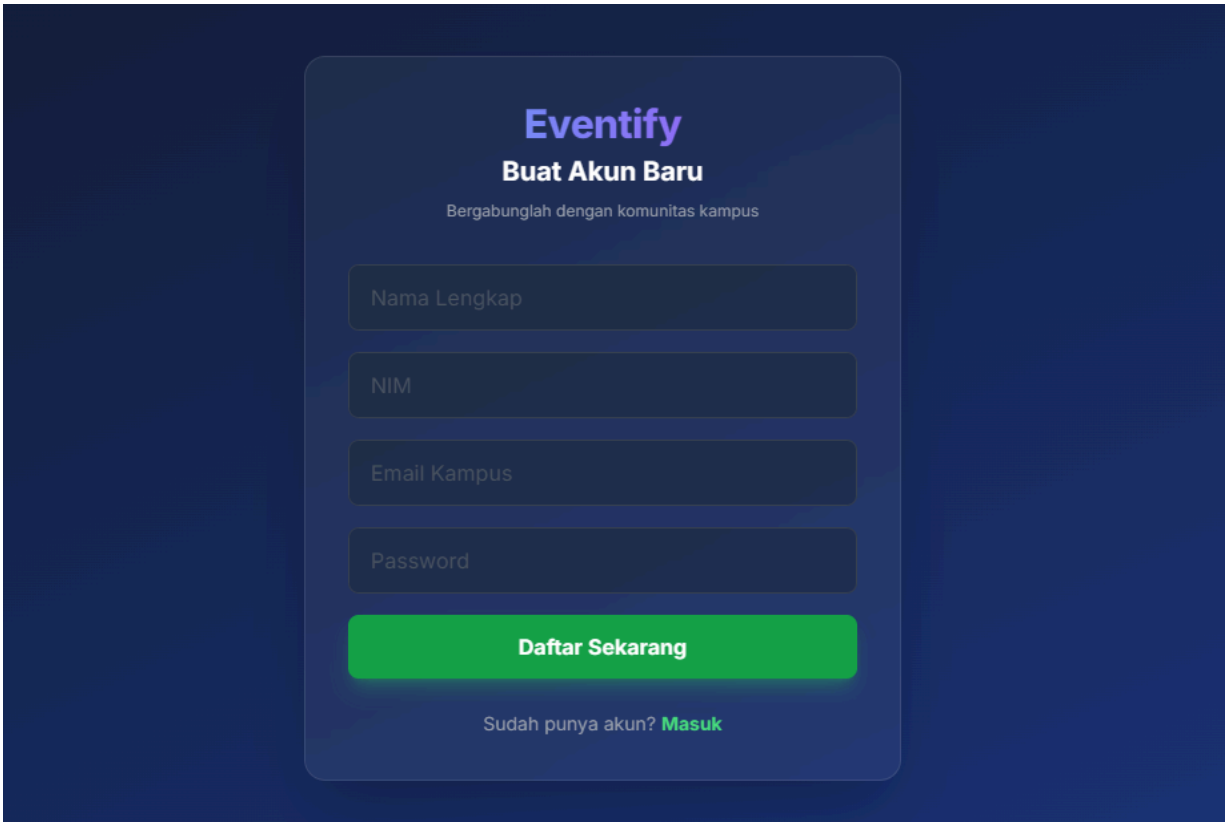
1. **Login Page**



Halaman *login*

Halaman Login berfungsi sebagai titik masuk utama yang menerapkan mekanisme autentikasi aman untuk memverifikasi identitas pengguna sebelum memberikan hak akses ke dalam sistem. Secara teknis, halaman ini tidak melakukan manipulasi data baru (*stateless transaction*), melainkan beroperasi dalam mode pembacaan data (*Read-Only*) yang ketat. Saat pengguna memasukkan kredensial, sistem *Backend* akan melakukan pencocokan informasi tersebut dengan rekam data yang tersimpan pada entitas User di dalam basis data eventify_db. Proses validasi ini membandingkan kombinasi atribut email dan password yang diinputkan dengan data terdaftar; jika cocok, sistem akan mengembalikan user_id sebagai token sesi yang valid, memastikan bahwa hanya pengguna terverifikasi yang dapat berinteraksi dengan fitur-fitur internal aplikasi.

2. Sign-Up Page



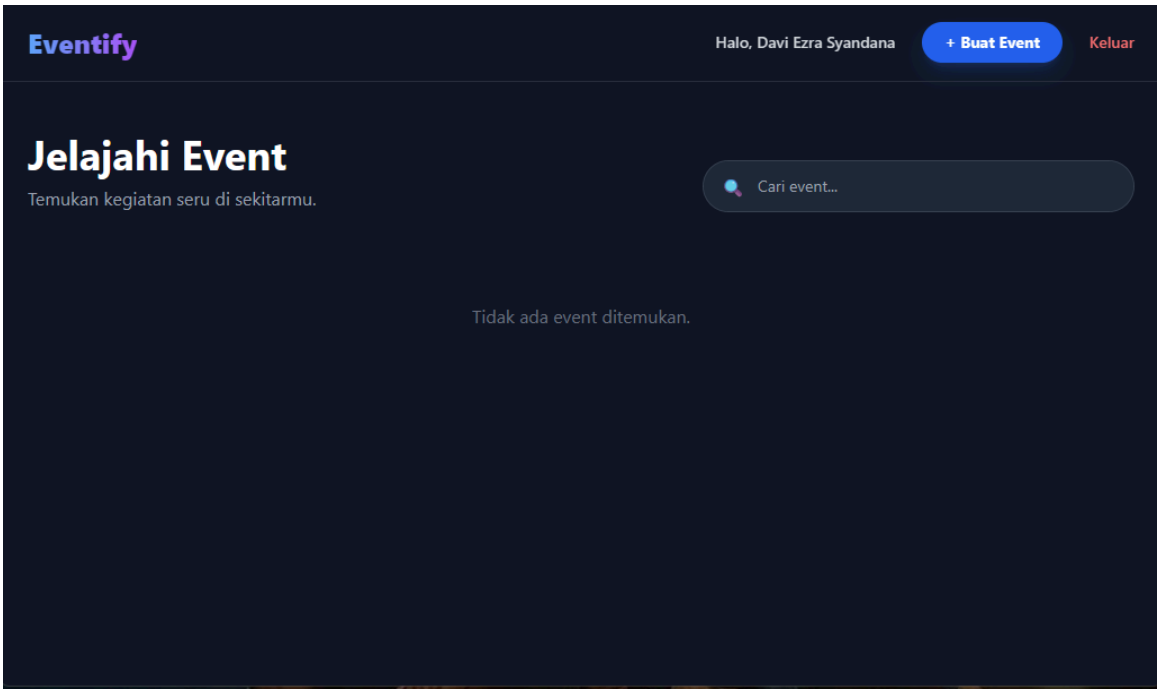
Halaman *sign up*

Halaman Registrasi (*Sign Up*) berfungsi sebagai gerbang akuisisi pengguna baru yang dirancang untuk mengumpulkan data identitas mahasiswa secara terstruktur dan valid. Pada antarmuka ini, calon pengguna diwajibkan mengisi serangkaian atribut vital yang meliputi nama lengkap, Nomor Induk Mahasiswa (NIM), alamat email resmi kampus, serta kata sandi untuk keamanan akun. Desain formulir ini tidak hanya berfokus pada penginputan data semata, tetapi juga bertindak sebagai filter awal untuk memastikan kelengkapan informasi sebelum diproses lebih lanjut oleh sistem. Interaksi ini merepresentasikan langkah awal yang krusial dalam siklus hidup pengguna di dalam ekosistem Eventify, di mana data yang valid menjadi prasyarat mutlak untuk mendapatkan hak akses penuh terhadap fitur-fitur transaksional seperti pendaftaran acara dan pemberian ulasan.

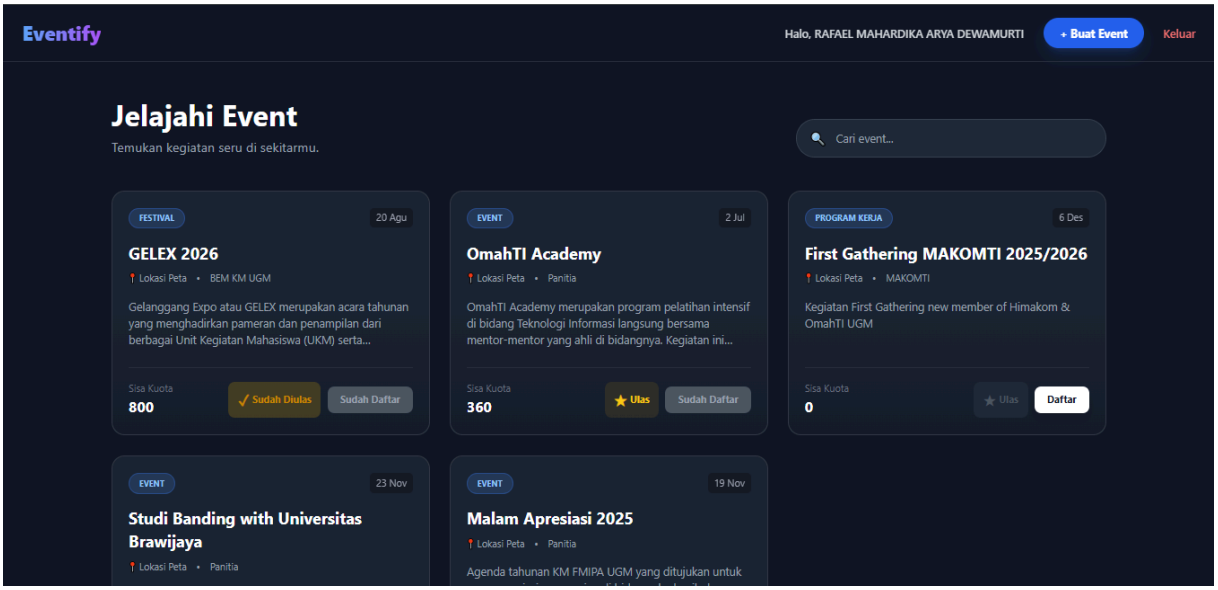
Dari perspektif teknis, proses registrasi ini merupakan implementasi nyata dari operasi *Create* dalam paradigma CRUD. Ketika tombol pendaftaran ditekan, aplikasi *Frontend* mengemas data input menjadi muatan JSON dan mengirimkannya melalui protokol HTTP POST menuju *Backend API*. Server kemudian merespons dengan mengeksekusi perintah SQL INSERT INTO Users, yang secara efektif menciptakan rekam data (*record*) baru di dalam tabel Users pada basis data eventify_db. Dalam proses ini, mesin basis data

memainkan peran krusial dengan secara otomatis membangkitkan user_id yang bersifat unik dan inkremental sebagai *Primary Key*, sekaligus menerapkan validasi integritas melalui *Unique Constraint* untuk menjamin bahwa tidak ada duplikasi data NIM atau email yang dapat mengacaukan konsistensi sistem.

3. Dashboard



Halaman *dashboard (kosong)*



Halaman *dashboard* (terisi)

Halaman Dashboard ini berfungsi sebagai antarmuka visual utama yang menyajikan data dinamis secara real-time dari basis data eventify_db. Mengingat struktur basis data yang dirancang dengan prinsip normalisasi, informasi yang ditampilkan pada setiap kartu acara sejatinya merupakan hasil agregasi kompleks dari lima tabel berbeda, yaitu Events, Categories, Organizers, Users, dan Feedback, yang disatukan melalui operasi JOIN di sisi Backend. Dalam satu kali pemrosesan, sistem tidak hanya mengambil data inti acara (seperti judul dan waktu), tetapi juga melengkapinya dengan label referensi (nama kategori dan penyelenggara) serta menyuntikkan konteks personalisasi untuk menentukan apakah pengguna yang sedang login memiliki status pendaftaran atau riwayat ulasan pada acara tersebut. Pendekatan ini memastikan pengguna menerima informasi yang komprehensif dan relevan dalam satu tampilan terpadu tanpa membebani kinerja jaringan.

a. Kartu Event

Komponen kartu acara (*Event Card*) yang mendominasi tampilan dashboard, seperti yang terlihat pada kartu "GELEX 2026" atau "OmahTI Academy" berfungsi sebagai wadah presentasi data dinamis yang ditarik langsung (*fetched*) dari tabel entitas Events. Secara teknis, setiap elemen visual pada kartu ini dipetakan secara presisi (*mapped*) terhadap atribut-atribut basis data: judul acara dirender dari kolom nama_acara, ringkasan kegiatan diambil dari kolom deskripsi, serta jadwal pelaksanaan ditampilkan berdasarkan data pada kolom tanggal_waktu_mulai. Selain itu, indikator sisa kuota tidak sekadar menampilkan angka statis, melainkan menyajikan data ketersediaan kursi secara *real-time* yang bersumber dari kolom kuota, memberikan transparansi informasi kepada pengguna sebelum mereka memutuskan untuk mendaftar.

b. Label Kategori & Penyelenggara

Informasi di dalam kartu tidak berdiri sendiri, melainkan mengambil dari tabel lain melalui *Foreign Key*:

- Label kategori (contohnya "FESTIVAL") diambil dari tabel Categories

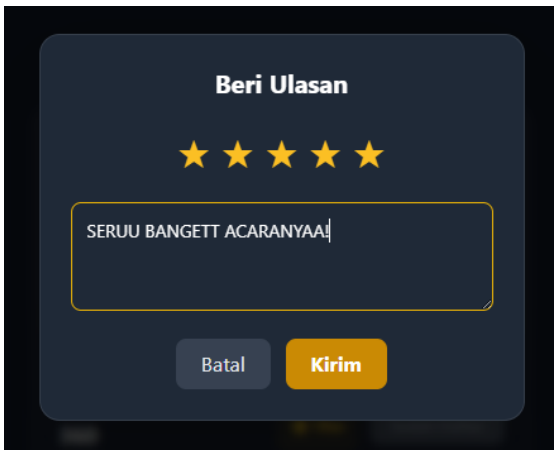
(nama_kategori) yang terhubung lewat category_id di tabel Events.

- Penyelenggara (contohnya "BEM KM UGM") diambil dari tabel Organizers (nama_organizer) yang terhubung lewat organizer_id di tabel Events

c. Tombol Daftar

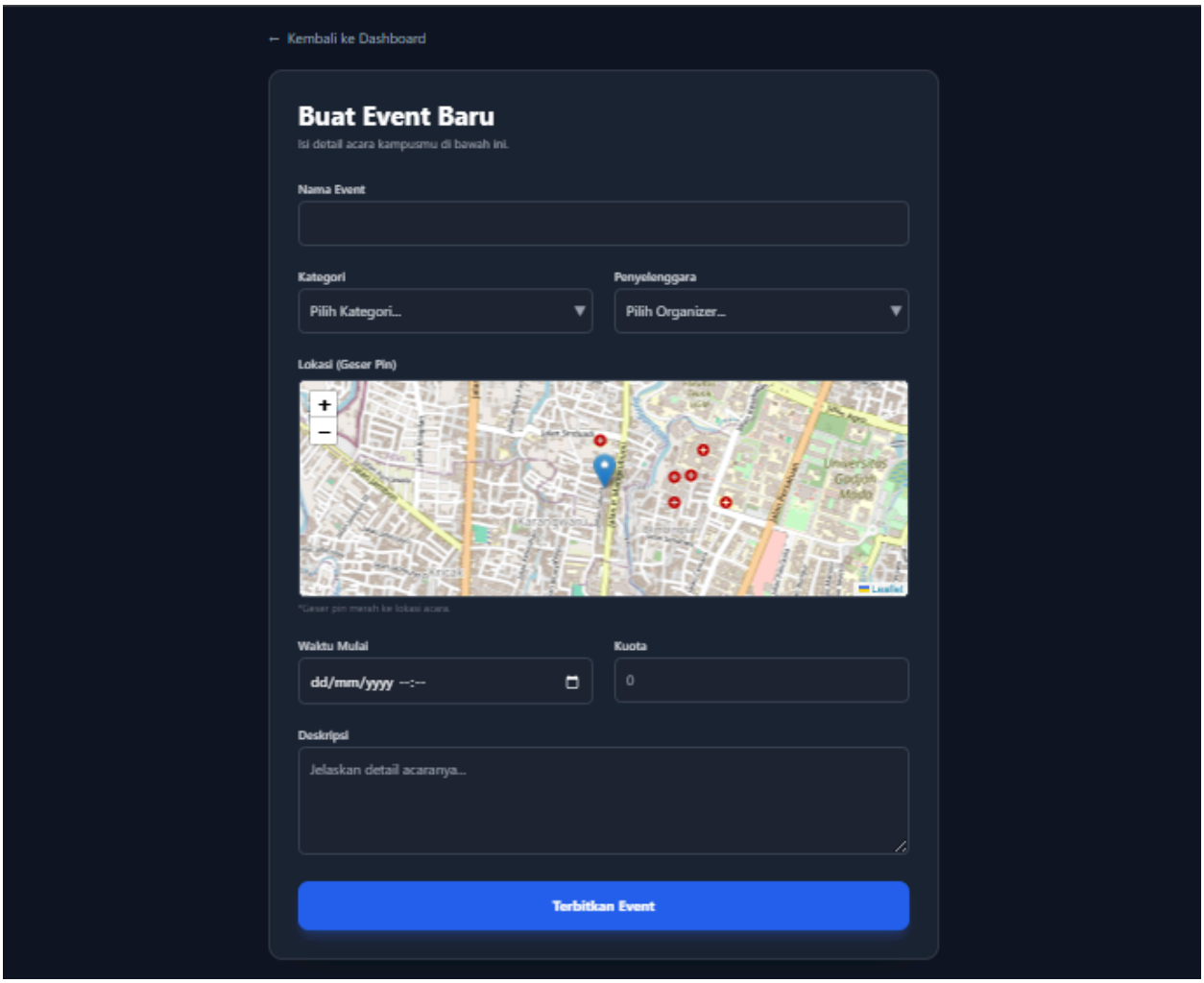
Tombol "Daftar" berperan sebagai pemicu transaksi kritis yang menjembatani interaksi pengguna di Frontend dengan lapisan keamanan data di Backend. Saat pengguna melakukan konfirmasi pendaftaran, aplikasi menyusun paket data berisi identitas pengguna (user_id) dan acara (event_id) untuk dikirimkan ke server. Di balik layar, basis data MySQL akan melakukan validasi ketat memanfaatkan Unique Constraint untuk memastikan bahwa satu mahasiswa tidak dapat mendaftar ganda pada acara yang sama. Apabila validasi berhasil dan data tersimpan, sistem antarmuka akan segera merespons dengan menonaktifkan tombol tersebut dan mengubah labelnya menjadi "Sudah Daftar", memberikan kepastian visual kepada pengguna bahwa hak kepesertaan mereka telah terkunci aman di dalam sistem.

d. Tombol Ulasan



Tombol Ulasan merupakan fitur interaktif yang secara eksklusif hanya dapat diakses oleh peserta yang telah berstatus terdaftar pada suatu event. Saat ulasan dikirim, sistem akan menyimpan data rating dan komentar ke dalam tabel Feedback sembari melakukan validasi otomatis menggunakan Unique Constraint di database. Mekanisme ini memastikan bahwa satu pengguna hanya dapat memberikan satu penilaian per acara, sehingga mencegah manipulasi data dan menjaga kredibilitas ulasan.

4. Create New Event



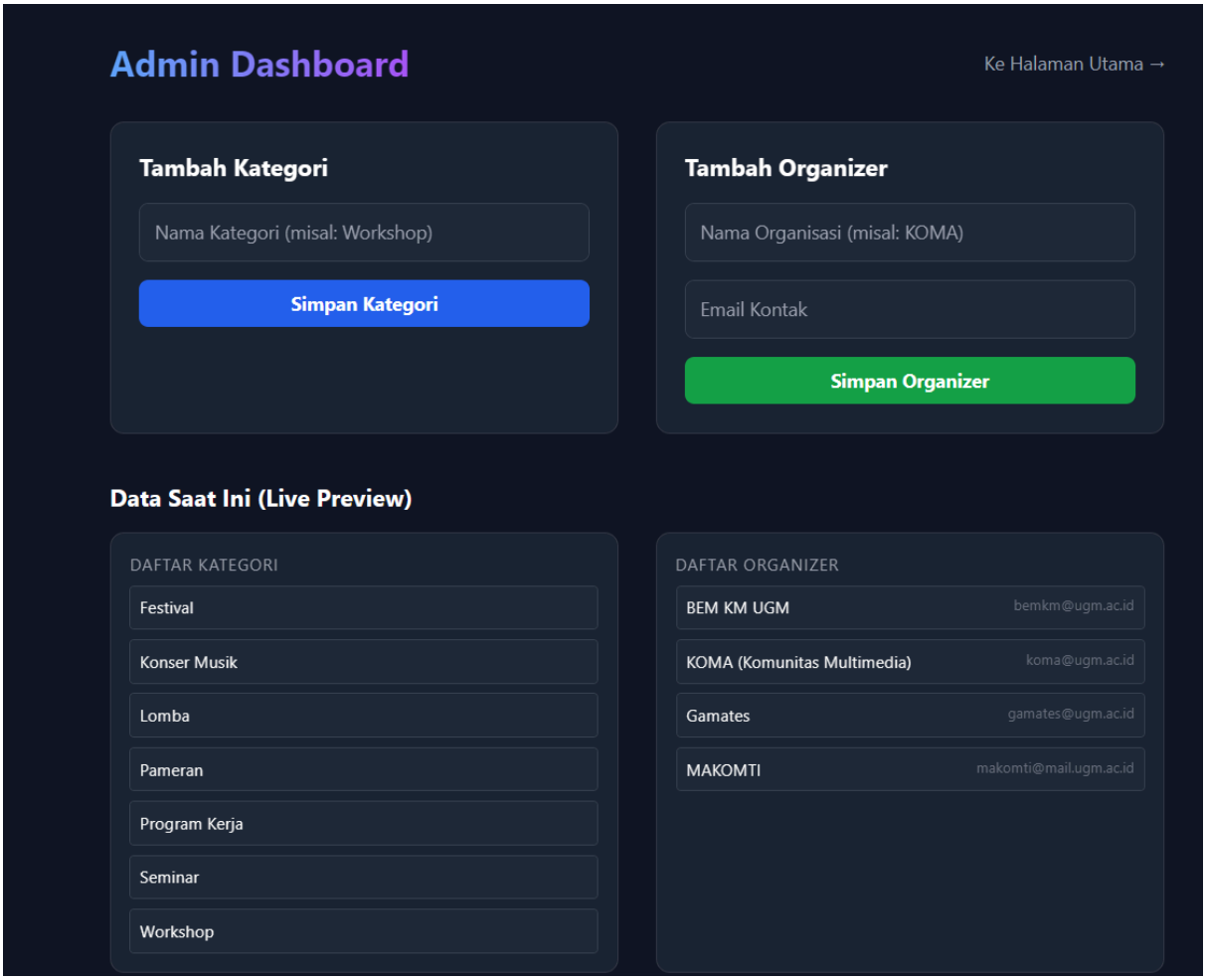
Halaman *Create Event*

Halaman "Buat Event Baru" dirancang sebagai antarmuka utama untuk menjalankan operasi *Create* pada entitas inti Events. Secara teknis, formulir ini berfungsi sebagai jembatan input yang memetakan data visual dari pengguna langsung ke dalam atribut-atribut tabel database eventify_db. Ketika tombol "Terbitkan Event" dieksekusi, sistem akan mengagregasi seluruh elemen input, mulai dari judul, deskripsi, hingga koordinat lokasi agar menjadi objek JSON. Objek ini kemudian dikirimkan melalui protokol HTTP POST ke server, yang selanjutnya menerjemahkannya menjadi perintah SQL INSERT untuk menciptakan satu baris data acara baru yang persisten di dalam sistem penyimpanan.

Lebih dari sekadar input teks, halaman ini mengimplementasikan logika integritas referensial (*Referential Integrity*) yang ketat sesuai desain ERD yang telah dirancang. Pada elemen *dropdown* untuk pemilihan "Kategori" dan "Penyelenggara", sistem tidak mengizinkan input teks bebas, melainkan mewajibkan pengguna memilih dari data

referensi yang diambil secara dinamis (*fetched*) dari tabel Categories dan Organizers. Mekanisme ini memastikan bahwa setiap acara yang terbentuk terhubung secara valid melalui *Foreign Key* (category_id dan organizer_id) ke entitas induknya, sehingga mencegah terjadinya anomali data atau hubungan yang terputus (*orphaned records*) dalam struktur basis data relasional.

5. Admin Dashboard



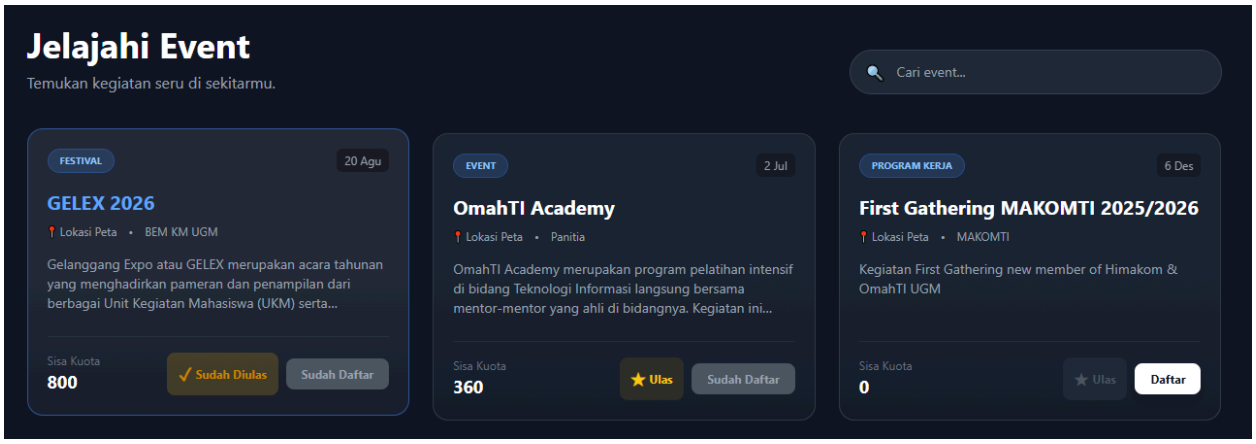
Halaman admin *dashboard*

Halaman Admin Dashboard dirancang sebagai pusat kendali untuk mengelola data referensi (Data Master) yang menjadi fondasi relasi dalam database eventify_db. Keberadaan halaman ini sangat krusial karena tabel utama Events sangat bergantung pada ketersediaan data dari entitas lain. Oleh karena itu, dashboard ini difokuskan untuk menangani operasi CRUD (*Create, Read, Update, Delete*) pada tabel pendukung, memastikan integritas data tetap terjaga sebelum pengguna biasa mulai membuat acara.

a. **Form Input (Operasi Create)** Pada bagian atas antarmuka, terdapat dua formulir terpisah yang berfungsi untuk memasukkan data baru (*INSERT*). Formulir di sisi kiri didedikasikan untuk tabel Categories, di mana input admin akan disimpan ke kolom nama_kategori. Sementara itu, formulir di sisi kanan terhubung langsung ke tabel Organizers untuk menyimpan identitas penyelenggara (seperti BEM atau UKM) beserta kontak emailnya. Langkah ini merupakan prasyarat mutlak dalam sistem; tanpa adanya data kategori dan organizer yang didaftarkan di sini, formulir pembuatan event tidak akan memiliki opsi referensi untuk dipilih, yang akan menyebabkan kegagalan dalam pengisian data.
b. **Live Preview (Operasi Read)** Beralih ke bagian bawah, terdapat area "Live Preview" yang menyajikan visualisasi data secara *real-time*. Secara teknis, area ini menjalankan perintah SQL SELECT untuk mengambil dan menampilkan seluruh daftar kategori dan organizer yang telah tersimpan di database. Fitur pemantauan ini penting bagi administrator untuk memastikan bahwa data referensi—yang nantinya bertindak sebagai *Foreign Key* (category_id dan organizer_id) pada tabel Events—sudah tersedia dan siap digunakan oleh pengguna.

4.2 Manajemen State Tombol (User Experience Adaptif)

Aspek *User Experience* (UX) menjadi fokus utama pada dashboard. Tampilan kartu event tidak statis, melainkan bersifat dinamis menyesuaikan dengan riwayat aktivitas pengguna. Berdasarkan data yang diterima dari "Smart Query" backend, Frontend merender tombol aksi dalam 3 kondisi (*state*) yang berbeda:



Kondisi 1: Belum Mendaftar (Initial State)

- **Visual:** Tombol "Daftar" menyala terang (Aktif), sedangkan Tombol "Ulas" berwarna abu-abu (Non-aktif/Disabled).
- **Logika:** Pengguna diperbolehkan mendaftar. Tombol ulasan sengaja dimatikan untuk mencegah *fake review* dari orang yang belum menjadi peserta acara.

Kondisi 2: Sudah Daftar (Registered State)

- **Visual:** Tombol "Daftar" berubah menjadi abu-abu dengan label **"Sudah Daftar"** (Disabled). Di saat bersamaan, Tombol "Ulas" menyala kuning (Aktif).
- **Logika:** Sistem mengunci pendaftaran agar tidak terjadi duplikasi (sesuai aturan bisnis). Akses untuk memberikan ulasan kini dibuka sebagai hak eksklusif peserta terdaftar.

Kondisi 3: Sudah Mengulas (Completed State)

- **Visual:** Tombol "Ulas" berubah menjadi status **"Sudah Diulas"** (Disabled dengan indikator centang).
- **Logika:** Siklus interaksi pengguna dengan event tersebut telah selesai. Pengguna tidak dapat lagi mengubah data sejarah pendaftaran maupun ulasan mereka.

BAB V KESIMPULAN DAN REFLEKSI

5.1 Kesimpulan

Pengembangan "Eventify" selama 5 minggu ini telah berhasil mentransformasi sebuah konsep abstrak menjadi aplikasi web fungsional penuh (*Full Stack*).

1. **Dari Desain ke Implementasi:** Kami berhasil menerjemahkan ERD (*Entity Relationship Diagram*) yang dirancang di Week 1 menjadi skema database fisik yang tangguh di Week 2, dan akhirnya dihidupkan dengan logika aplikasi di Week 5.
2. **Integrasi Sistem:** Sistem ini membuktikan bahwa arsitektur 3-Tier (Frontend-Backend-Database) adalah solusi yang tepat untuk memisahkan tanggung jawab, memudahkan *debugging*, dan menjaga keamanan data.
3. **Pemecahan Masalah Logika Bisnis:** Fitur "1 User 1 Event" bukan hanya sekadar tampilan UI, tetapi didukung oleh pertahanan berlapis (*defense in depth*) mulai dari validasi Frontend, pengecekan Backend, hingga *Unique Constraint* di Database.

5.2 Refleksi dan Pembelajaran

Perjalanan pengembangan ini memberikan banyak pelajaran teknis dan non-teknis bagi tim kami:

- **Pentingnya Konsistensi Penamaan:** Salah satu tantangan terbesar yang kami hadapi di Week 5 adalah bug "Null ID". Kami belajar dengan cara yang keras bahwa ketidaksesuaian kecil antara nama kolom di database (event_id) dan pemanggilan properti di JavaScript (event.id) dapat melumpuhkan seluruh fitur. Konsistensi adalah kunci dalam integrasi sistem.
- **Asynchronous Programming itu Rumit namun Powerful:** Mengelola urutan eksekusi kode (seperti kapan harus menutup modal vs kapan harus mengirim data) mengajarkan kami pentingnya memahami konsep *Promise* dan *Async/Await* dalam JavaScript agar data tidak hilang di tengah jalan.
- **Database Constraint sebagai Penyelamat:** Kami menyadari bahwa validasi di Frontend saja tidak cukup. Penerapan *Constraint* (seperti UNIQUE dan CHECK) di level database adalah jaring pengaman terakhir yang sangat vital untuk menjaga kebersihan data.

5.3 Tantangan yang Dihadapi

Tantangan terbesar yang kami hadapi adalah dalam hal **Manajemen State Aplikasi**. Membuat tombol dashboard yang dapat berubah warna dan status secara otomatis (misalnya dari "Daftar" menjadi "Sudah Daftar") membutuhkan logika query yang cukup kompleks (*Subquery SQL*)

yang harus disinkronkan dengan logika rendering di Frontend. Selain itu, proses **Debugging Full Stack** juga menuntut ketelitian tinggi untuk melacak di lapisan mana sebuah error terjadi—apakah data salah dikirim oleh Frontend, salah diproses oleh Backend, atau ditolak oleh Database.

5.4 Rencana Pengembangan Selanjutnya (*Future Improvements*)

Mengingat ini adalah prototipe akademis, masih banyak ruang untuk peningkatan agar "Eventify" siap digunakan secara nyata:

1. **Sistem Pembayaran:** Menambahkan integrasi *Payment Gateway* untuk event berbayar.
2. **Notifikasi Email:** Mengirimkan tiket elektronik ke email mahasiswa setelah pendaftaran berhasil (menggunakan Nodemailer).
3. **Dashboard Admin Super:** Fitur bagi admin BEM/Fakultas untuk memverifikasi atau menolak pengajuan event dari organizer sebelum tampil di publik.
4. **Fitur Profil:** Memungkinkan mahasiswa mengupload foto profil dan melihat riwayat sertifikat kegiatan mereka.

Dengan pondasi yang kuat ini, kami optimis "Eventify" memiliki potensi besar untuk dikembangkan menjadi sistem manajemen kegiatan yang diandalkan di lingkungan kampus.