DOKUMEN SPESIFIKASI FITUR APLIKASI WEDDING ORGANIZER

Tanggal: 17 Juli 2025 **Disusun oleh:** Owner & Principal Planner

A. Pendahuluan

Dokumen ini merincikan spesifikasi fungsional untuk pengembangan aplikasi manajemen Wedding Organizer (WO) yang komprehensif. Aplikasi ini dirancang sebagai ekosistem digital terpusat yang bertujuan untuk:

1. Meningkatkan efisiensi operasional tim WO.
2. Memberikan transparansi dan pengalaman yang superior bagi klien.
3. Mengelola hubungan dengan vendor secara sistematis.
4. Mendukung pertumbuhan bisnis WO melalui fitur analitik dan pemasaran.

Aplikasi ini akan dibagi menjadi beberapa modul utama yang saling terintegrasi.

MODUL 1: MANAJEMEN INTERNAL & OPERASIONAL

Tujuan Modul: Menjadi pusat komando (command center) bagi tim internal WO untuk mengelola sumber daya, tugas, dan alur kerja.

1.1. Fitur: Manajemen Peran & Akses (Role-Based Access Control)

- **Tujuan:** Menjamin keamanan data dan memastikan setiap anggota tim hanya dapat mengakses informasi yang relevan dengan tanggung jawabnya.
- **Deskripsi Fungsional:** Sistem memungkinkan Admin untuk membuat akun bagi setiap anggota tim dan menetapkan peran (role) spesifik. Setiap peran memiliki hak akses yang berbeda-beda terhadap modul dan data di dalam aplikasi.
- **Poin Kunci/Sub-Fitur:**
    - **Owner/Admin:** Akses penuh ke semua fitur, termasuk laporan keuangan perusahaan, pengaturan sistem, dan manajemen semua peran.
    - **Lead Planner:** Dapat membuat dan mengelola event, menugaskan pekerjaan ke asisten, berkomunikasi dengan klien & vendor, dan melihat anggaran per event.
    - **Planner Assistant:** Akses terbatas pada event yang ditugaskan. Hanya bisa melihat dan mengeksekusi tugas yang diberikan oleh Lead Planner.
    - **Tim Keuangan:** Akses khusus ke Modul Anggaran, invoice, dan laporan pembayaran. Tidak dapat mengubah detail acara.
    - **Tim Lapangan (Day-of Crew):** Akses *read-only* yang sangat terbatas pada Modul Hari-H (rundown, kontak darurat) saat acara berlangsung.

1.2. Fitur: Manajemen Tugas Internal

- **Tujuan:** Mengorganisir pekerjaan internal tim dan memastikan tidak ada tugas yang terlewat.
- **Deskripsi Fungsional:** Lead Planner atau Owner dapat membuat tugas, menetapkan penanggung jawab (assignee) dari tim internal, mengatur tanggal jatuh tempo (due date), dan menetapkan prioritas.
- **Poin Kunci/Sub-Fitur:**
    - Status Tugas: To-Do, In Progress, For Review, Done.
    - Sistem notifikasi untuk tugas baru dan pengingat deadline.
    - Kolom komentar pada setiap tugas untuk diskusi internal.

MODUL 2: MANAJEMEN EVENT & PORTAL KLIEN

Tujuan Modul: Mengelola setiap proyek pernikahan dari awal hingga akhir dan menjadi antarmuka utama untuk berkomunikasi dan berkolaborasi dengan klien.

2.1. Fitur: Dashboard Event & Proyek

- **Tujuan:** Memberikan gambaran umum (helicopter view) dari setiap proyek pernikahan.
- **Deskripsi Fungsional:** Setiap pernikahan yang dikelola akan menjadi sebuah "proyek" di dalam aplikasi. Dashboard ini menampilkan ringkasan dari proyek yang dipilih.
- **Poin Kunci/Sub-Fitur:**
    - Informasi utama: Nama Klien, Tanggal Pernikahan, Venue.
    - Widget Ringkasan Anggaran (Budget Summary).
    - Widget Tugas Mendatang (Upcoming Tasks).
    - Notifikasi terbaru dari klien atau vendor.

2.2. Fitur: Portal Klien (Client Portal)

- **Tujuan:** Memberikan klien akses eksklusif untuk memantau perkembangan, berkolaborasi dengan planner, dan mengakses semua informasi terkait pernikahan mereka di satu tempat.
- **Deskripsi Fungsional:** Setiap pasangan klien mendapatkan login pribadi ke portal yang berisi semua informasi pernikahan mereka. Tampilan didesain agar ramah pengguna dan menenangkan.
- **Poin Kunci/Sub-Fitur:**
    - **Dashboard Klien:** Tampilan sederhana berisi hitung mundur hari-H, tugas yang perlu mereka kerjakan, dan pengingat pembayaran.
    - **Checklist Kolaboratif:** Klien dapat melihat seluruh timeline perencanaan, melihat progres WO, dan menyelesaikan tugas yang menjadi tanggung jawab mereka.
    - **Papan Inspirasi (Mood Board):** Area visual bagi klien dan planner untuk mengunggah dan memberi komentar pada gambar referensi (dekorasi, gaun, dll).
    - **Pusat Dokumen Klien:** Tempat klien mengunggah dokumen pribadi (KTP, dll) dan melihat dokumen dari WO (kontrak, dll).

MODUL 3: MANAJEMEN VENDOR

Tujuan Modul: Menjadi database digital dan alat manajemen untuk semua vendor rekanan.

3.1. Fitur: Database & Profil Vendor

- **Tujuan:** Mengelola "buku hitam" digital berisi daftar vendor terpercaya milik WO.
- **Deskripsi Fungsional:** Sebuah database yang dapat dicari dan disaring untuk menemukan vendor dengan cepat. Setiap vendor memiliki halaman profilnya sendiri.
- **Poin Kunci/Sub-Fitur:**
    - Filter berdasarkan: Kategori (Fotografi, Katering, dll.), Lokasi, Rentang Harga.
    - Profil Vendor berisi: Kontak, Portofolio, Pricelist (dapat diunggah), rekening bank.
    - **Catatan & Rating Internal:** Kolom khusus untuk tim WO memberi rating dan catatan pribadi tentang kinerja vendor, yang tidak dapat dilihat oleh pihak luar.

3.2. Fitur: Manajemen Kontrak & Booking Vendor

- **Tujuan:** Melacak status pemesanan dan menyimpan semua dokumen legal vendor di satu tempat.
- **Deskripsi Fungsional:** Untuk setiap event, sistem dapat menandai vendor mana yang telah dibooking. Terdapat fitur untuk mengunggah dan menyimpan salinan digital dari kontrak dan invoice dari vendor.
- **Poin Kunci/Sub-Fitur:**
    - Status Vendor per Event: Terpilih, Menunggu Kontrak, DP Dibayar, Lunas.
    - Pengingat jadwal pembayaran ke vendor.

MODUL 4: PERENCANAAN & KOLABORASI INTI

Tujuan Modul: Menyediakan alat-alat kerja utama yang digunakan dalam proses perencanaan pernikahan.

4.1. Fitur: Timeline & Checklist Cerdas

- **Tujuan:** Menyediakan panduan langkah-demi-langkah yang terstruktur dari awal hingga akhir perencanaan.
- **Deskripsi Fungsional:** Sistem secara otomatis menghasilkan daftar tugas (checklist) berdasarkan tanggal pernikahan. Contoh: jika pernikahan 12 bulan lagi, tugas "Booking Venue" akan otomatis muncul.
- **Poin Kunci/Sub-Fitur:**
    - Berbasis Template: Kemampuan membuat template checklist untuk berbagai jenis pernikahan (Intimate, Adat, dll).
    - Tugas dapat dialokasikan ke WO atau Klien.

4.2. Fitur: Manajemen Tamu & Denah Tempat Duduk

- **Tujuan:** Membantu klien (dan WO) dalam mengelola tugas yang paling rumit: daftar tamu dan penempatan tempat duduk.
- **Deskripsi Fungsional:** Klien dapat mengelola daftar tamu mereka di dalam portal. Data ini kemudian digunakan dalam alat pembuat denah tempat duduk yang visual.
- **Poin Kunci/Sub-Fitur:**
    - Impor daftar tamu dari file Excel.
    - Pelacakan status RSVP (Hadir, Tidak Hadir, Ragu-ragu).
    - Kategorisasi tamu (Keluarga Pria, Teman Wanita, dll).
    - Pencatatan kebutuhan khusus (alergi, vegetarian).
    - Alat *drag-and-drop* untuk menempatkan nama tamu ke denah meja yang sudah dibuat.

MODUL 5: MANAJEMEN ANGGARAN & FINANSIAL

Tujuan Modul: Memberikan kontrol penuh dan transparansi atas semua aspek keuangan dari setiap event.

5.1. Fitur: Pelacak Anggaran (Budget Tracker)

- **Tujuan:** Memantau pengeluaran secara real-time dan memastikan event tetap sesuai anggaran yang disepakati.
- **Deskripsi Fungsional:** Sebuah spreadsheet canggih yang memecah anggaran ke dalam kategori-kategori yang dapat disesuaikan.
- **Poin Kunci/Sub-Fitur:**
    - Kolom: Item, Kategori, Estimasi Anggaran, Harga Deal, Sudah Dibayar, Sisa Pembayaran, Status.
    - Ringkasan otomatis total anggaran dan sisa pembayaran.

5.2. Fitur: Manajemen Pembayaran & Invoice

- **Tujuan:** Mengotomatiskan proses penagihan ke klien dan memastikan pembayaran tepat waktu.
- **Deskripsi Fungsional:** Sistem dapat menghasilkan invoice untuk dikirim ke klien dan melacak statusnya.
- **Poin Kunci/Sub-Fitur:**
    - Jadwal pembayaran otomatis yang mengirimkan notifikasi ke klien dan tim keuangan WO saat mendekati jatuh tempo.
    - Klien dapat mengunggah bukti pembayaran langsung melalui portal mereka.

MODUL 6: EKSEKUSI HARI-H (D-DAY)

Tujuan Modul: Memastikan eksekusi acara di lapangan berjalan mulus, terkoordinasi, dan tanpa hambatan.

6.1. Fitur: Rundown Digital Interaktif

- **Tujuan:** Memberikan panduan jadwal yang akurat dan *real-time* kepada seluruh kru dan vendor yang bertugas.
- **Deskripsi Fungsional:** Jadwal acara menit-demi-menit yang dapat diakses melalui aplikasi mobile. Perubahan yang dibuat oleh koordinator utama akan langsung tersinkronisasi ke perangkat semua orang.
- **Poin Kunci/Sub-Fitur:**
    - Detail per item: Waktu, Durasi, Aktivitas, Penanggung Jawab (PIC), Lokasi, Catatan.
    - **Mode Offline:** Rundown dan kontak penting dapat diakses bahkan tanpa koneksi internet.

6.2. Fitur: Kontak Darurat & Informasi Cepat

- **Tujuan:** Menyediakan akses cepat ke semua kontak penting selama hari acara.
- **Deskripsi Fungsional:** Sebuah halaman khusus di dalam modul Hari-H yang berisi daftar semua PIC dari tim WO, keluarga, dan setiap vendor.
- **Poin Kunci/Sub-Fitur:**
    - Tombol "Call" atau "WhatsApp" sekali klik.

MODUL 7: PERTUMBUHAN BISNIS & ANALITIK

Tujuan Modul: Menyediakan data dan alat untuk membantu Owner membuat keputusan bisnis yang lebih baik dan mendapatkan klien baru.

7.1. Fitur: Manajemen Prospek (CRM)

- **Tujuan:** Mengelola calon klien (leads) secara terstruktur.
- **Deskripsi Fungsional:** Sebuah kanban board atau daftar untuk melacak semua pertanyaan yang masuk, dari kontak pertama hingga penandatanganan kontrak.
- **Poin Kunci/Sub-Fitur:**
    - Status Prospek: Baru, Sudah Dihubungi, Proposal Terkirim, Negosiasi, Deal, Gagal.
    - Pencatatan sumber prospek (Instagram, Pameran, dll).

7.2. Fitur: Analitik & Laporan

- **Tujuan:** Memberikan wawasan tentang kinerja bisnis.
- **Deskripsi Fungsional:** Dashboard analitik untuk Owner yang menampilkan data dari semua event yang telah selesai.
- **Poin Kunci/Sub-Fitur:**
    - Laporan Profitabilitas per Event.
    - Analitik Vendor: Vendor mana yang paling sering dipakai & paling menguntungkan.
    - Analitik Klien: Rata-rata budget klien per tahun.

MODUL 8: FITUR LANJUTAN & INTEGRASI

Tujuan Modul: Memberikan nilai tambah (value-added services) yang membedakan WO di pasar.

8.1. Fitur: Pembuat Website & Undangan Digital

- **Tujuan:** Menawarkan layanan tambahan yang sangat berguna bagi klien tanpa biaya ekstra.
- **Deskripsi Fungsional:** Alat sederhana bagi klien untuk membuat wedding website dan undangan digital berdasarkan template yang disediakan.
- **Poin Kunci/Sub-Fitur:**
    - Website berisi: Cerita Pasangan, Galeri, Detail Acara, dan Formulir RSVP.
    - Data dari formulir RSVP otomatis terhubung ke Modul Manajemen Tamu.

8.2. Fitur: Integrasi Eksternal

- **Tujuan:** Menghubungkan aplikasi dengan alat lain yang biasa digunakan untuk alur kerja yang lebih lancar.
- **Deskripsi Fungsional:** Kemampuan untuk menyinkronkan data dari aplikasi ke platform eksternal.
- **Poin Kunci/Sub-Fitur:**
    - **Integrasi Kalender:** Sinkronisasi jadwal meeting dan pembayaran ke Google Calendar atau Apple Calendar milik tim WO dan klien.
    - **Integrasi Peta:** Alamat venue terhubung ke Google Maps/Waze.

B. Spesifikasi Teknis Sistem

Bagian ini merinci arsitektur teknis dan teknologi inti yang akan digunakan dalam pengembangan aplikasi Wedding Organizer.

1. Arsitektur Aplikasi

- **Tipe Arsitektur:** Monolitik (Frontend dan Backend digabungkan dalam satu repositori project) dengan pendekatan "Single Page Application (SPA)" melalui Inertia.js.
- **Komunikasi**: Komunikasi antara Frontend (React.js) dan Backend (Laravel) akan ditangani sepenuhnya oleh Inertia.js, memanfaatkan routing dan controller Laravel untuk merender komponen React.
- **Layering Arsitektur Backend**:
    - **Presentation Layer (Controller)**: Menangani request HTTP, memvalidasi input, dan mendelegasikan tugas ke Service Layer.
    - **Service Layer:** Berisi logika bisnis utama aplikasi. Bertanggung jawab mengorkestrasi operasi dari Repository dan DTO.
    - **Data Transfer Object (DTO):** Objek yang digunakan untuk membawa data antar layer aplikasi, memastikan data yang masuk ke Service Layer terstruktur dan tervalidasi.

- **Repository Layer**: Berinteraksi langsung dengan model Eloquent dan database, mengabstraksi detail persistensi data dari Service Layer.
- **Model Layer**: Representasi tabel database (Eloquent Models).
- **Framework:** Laravel 12
    - **Versi PHP:** PHP 8.2 atau yang lebih baru (sesuai rekomendasi Laravel 12).
    - **Database:** MySQL 8.x atau PostgreSQL 13+
        - **Driver Database:** Eloquent ORM.
    - **Caching:** Redis (disarankan untuk performa lebih baik, terutama untuk sesi dan cache aplikasi).
    - **Queue Driver:** Redis atau Database (untuk tugas-tugas background seperti notifikasi atau pengiriman email).
    - **Storage Driver:** Local Disk, dengan opsi untuk S3 kompatibel (contoh: AWS S3, DigitalOcean Spaces) untuk penyimpanan file yang lebih scalable (gambar mood board, dokumen, portofolio vendor).
    - **Autentikasi:** Laravel Sanctum atau Laravel Fortify (direkomendasikan Fortify untuk fitur autentikasi siap pakai).
    - **Testing:** PHPUnit, PestPHP (opsional, untuk pengujian unit dan fungsional).
- **Implementasi Arsitektur Backend:**
    - **Service Pattern:**
        - Setiap domain/modul bisnis (misalnya, `UserService`, `EventService`, `VendorService`) akan memiliki kelas Service-nya sendiri.
        - Service bertanggung jawab atas validasi data lanjutan (jika tidak sepenuhnya ditangani DTO), logika bisnis kompleks, dan koordinasi dengan multiple repository.
    - **Repository Pattern:**
        - Setiap model utama akan memiliki Repository-nya sendiri (misalnya, `UserRepository`, `EventRepository`, `VendorRepository`).
        - Repository akan berisi method untuk operasi CRUD dasar dan query database yang kompleks, mengembalikan model atau koleksi model.
    - **Data Transfer Object (DTO) Implementation:**
        - **Tujuan:** Memvalidasi, mengkapsulasi, dan menstandardisasi data input sebelum mencapai Service Layer, mengurangi ketergantungan Service pada format request.
        - **Penggunaan:**
            - **Data Banyak:** Jika data yang diserahkan ke Service banyak (misalnya, membuat order baru beserta item-itemnya), maka data tersebut **harus** melalui DTO terlebih dahulu.
                - DTO akan memiliki method khusus untuk memparsing data sesuai kebutuhan repository atau model.
                - **Contoh:** Untuk data yang masuk ke 2 model atau lebih, DTO akan menyediakan fungsi terpisah.

Service akan menerima instance DTO, kemudian memanggil method `toOrderData()` atau `toItemsData()` dari DTO tersebut sebelum menyerahkan data ke repository.

- **Data Sedikit:** Jika data yang diserahkan ke Service sedikit (misalnya, update status boolean, atau hanya 1-2 parameter), maka data dapat langsung di-parsing ke dalam argumen method Service tanpa harus melalui DTO yang terpisah. Namun, validasi input tetap harus dilakukan di Controller atau menggunakan Form Request.

3. Teknologi Frontend

- **Library UI:** React.js (Versi terbaru stabil)
  - **Builder/Bundler:** Vite.js (direkomendasikan oleh Laravel untuk proses development yang cepat).
  - **State Management:** React Context API atau Zustand/Jotai (untuk state management yang ringan dan efisien). Redux (jika diperlukan state yang lebih kompleks dan terpusat).
  - **Styling:** Tailwind CSS (untuk pengembangan UI yang cepat dan konsisten) atau CSS-in-JS (seperti Styled Components/Emotion, jika diperlukan kontrol styling yang lebih granular).
  - **UI Component Library (opsional):** Headless UI, Shadcn/UI, atau Ant Design/Material-UI (jika diperlukan komponen UI siap pakai yang kaya fitur).
- **Adaptor SPA:** Inertia.js (Versi terbaru stabil)
  - **Tujuan:** Menghubungkan frontend React.js dengan backend Laravel, memungkinkan pengalaman SPA tanpa membangun API terpisah.
  - **Fitur yang Digunakan:**
    - Server-side routing dengan respons JSON Inertia.
    - Client-side navigation tanpa refresh halaman penuh.
    - Form submission dengan `useForm` dari Inertia.
    - Shared data (props) dari Laravel ke komponen React.

4. Lingkungan Pengembangan & Deployment

- **Sistem Kontrol Versi:** Git.
- **Repository Code:** GitHub.
- **Deployment Otomatis:** GitHub Actions(untuk proses deployment yang terotomatisasi).

5. Keamanan

- **Autentikasi & Otorisasi:** Menggunakan sistem Role-Based Access Control (RBAC) yang dibangun di atas Laravel Auth dan otorisasi menggunakan Gates/Policies, dikombinasikan dengan middleware Inertia untuk memastikan akses yang benar berdasarkan peran.

- **Validasi Input:** Validasi sisi server menggunakan Laravel Validator untuk mencegah injeksi data berbahaya.
- **Proteksi CSRF:** Laravel CSRF protection.
- **Sanitasi Data:** Sanitasi input untuk mencegah XSS.
- **HTTPS:** Wajib menggunakan HTTPS untuk semua komunikasi.

6. Pertimbangan Performa

- **Optimasi Database:** Indeksasi database, penggunaan eager loading (Laravel Eloquent) untuk menghindari N+1 problem.
- **Caching:** Penggunaan caching pada data yang sering diakses (Redis).
- **Lazy Loading:** Implementasi lazy loading pada komponen React dan gambar.
- **Optimasi Aset Frontend:** Minifikasi dan kompresi JavaScript/CSS, penggunaan CDN untuk aset statis (jika diperlukan skala besar).

BAGIAN II: SPESIFIKASI TEKNIS

B. Pendahuluan Teknis

Bagian ini merincikan arsitektur teknis, alur kerja data, dan standar pengkodean yang akan digunakan dalam pengembangan aplikasi. Tujuannya adalah untuk menciptakan aplikasi yang *scalable*, *maintainable*, dan aman, dengan mengikuti praktik terbaik dalam ekosistem Laravel dan React.

2.1. Teknologi yang Digunakan

- **Backend Framework:** Laravel 12
- **Frontend Library:** React JS
- **Frontend Server State Management:** TanStack Query v5 (React Query)
- **Adapter Backend-Frontend:** Inertia.js
- **Database:** MySQL 8.0+ atau PostgreSQL 14+
- **Real-time Event Broadcasting:** Laravel Reverb
- **Cache Driver:** `file` (Bawaan Laravel)
- **Queue Driver:** `database` (Bawaan Laravel)
- **Styling:** Tailwind CSS

2.2. Arsitektur Aplikasi & Alur Data

Berikut adalah alur data standar untuk setiap *request* yang mengubah data (Create, Update, Delete):

1. Frontend: React JS, TanStack Query & Routing Laravel

- **Deskripsi:** Lapisan frontend bertanggung jawab atas presentasi UI dan manajemen *server state*. Interaksi pengguna akan memicu *request* ke backend melalui alur yang terdefinisi dengan jelas.

1. **Routing:** Semua *endpoint* aplikasi didefinisikan secara eksklusif menggunakan sistem *routing* standar Laravel di dalam file `routes/web.php`. Tidak ada sistem *routing* frontend yang terpisah.
2. **Server State Management:** TanStack Query akan menjadi satu-satunya penanggung jawab untuk semua interaksi dengan backend, termasuk mengambil data (*query*), mengubah data (*mutation*), caching di sisi klien, dan mengelola status UI (loading, error, success).

- **Alur Kerja Mutasi Data (Contoh: Membuat Event Baru):**
  1. Pengguna mengisi formulir di komponen React dan menekan tombol "Simpan".
  2. Aksi `onClick` pada tombol tersebut akan memanggil fungsi `mutate` dari *hook* `useMutation` yang disediakan oleh TanStack Query.

Hook `useMutation` akan dikonfigurasi untuk menjalankan fungsi yang melakukan *request*. Fungsi ini akan menggunakan *router* dari Inertia.js untuk mengirim data. Contoh:
JavaScript
```
// Di dalam komponen React
const { mutate, isPending } = useMutation({
  mutationFn: (newEventData) => router.post('/events', newEventData),
  onSuccess: () => { /* Tampilkan notifikasi sukses */ },
  onError: (errors) => { /* Tampilkan error validasi */ }
});
```

3.
4. TanStack Query secara otomatis akan mengelola status `isPending` (untuk menampilkan *loading spinner*) dan menangani *callback* `onSuccess` atau `onError` dari Inertia.
5. *Request* kemudian dikirim ke *endpoint* `/events` yang telah didefinisikan di `routes/web.php`.

- **Arsitektur Frontend (Atomic Design):**

  Prinsip: Mengorganisir komponen UI menjadi hierarki yang jelas, dari elemen terkecil hingga kompleks.

  Struktur Folder (Contoh):

  - `resources/js/Components/`
    - `Atoms/`: Elemen UI dasar dan tidak dapat dipecah lagi. Contoh: Button, Input, Icon, Text, Link.
    - `Molecules/`: Kelompok Atom yang bekerja sama sebagai satu unit. Contoh: SearchBar (Input + Button), FormField (Label + Input), DropdownMenu.
    - `Organisms/`: Kelompok Molekul dan/atau Atom yang membentuk bagian yang lebih besar dan berbeda dari suatu halaman. Contoh: Navbar, Sidebar, CardEvent, TableData.
    - `Templates/`: Mengatur Organisme ke dalam tata letak halaman tanpa data spesifik. Ini tentang struktur, bukan konten. Contoh: PageLayout (dengan Header, Sidebar, Content Area), FormTemplate.

- `Pages/`: Instance spesifik dari Template dengan data nyata. Ini adalah komponen yang dirender oleh Inertia. Contoh: EventListPage, CreateEventPage, UserProfilePage.

2. Validasi Request (Laravel Form Request)

- **Deskripsi:** Setiap *route* yang menerima data akan dilindungi oleh sebuah *Form Request class* khusus (misalnya, `App\Http\Requests\StoreEventRequest`). Kelas ini bertanggung jawab untuk dua hal:
  1. **Authorization:** Memastikan pengguna yang melakukan *request* memiliki izin untuk melakukannya.
  2. **Validation:** Memvalidasi semua data yang masuk sesuai aturan yang ditentukan (misalnya, `name` harus `required|string`, `event_date` harus `required|date`).
- **Alur:** Jika validasi gagal, Laravel secara otomatis akan mengembalikan respons ke frontend dengan pesan-pesan error, yang kemudian akan ditangkap oleh *callback* `onError` pada `useMutation` di TanStack Query untuk ditampilkan di UI. Jika berhasil, eksekusi akan berlanjut ke Controller.

3. Data Transfer Object (DTO)

- **Deskripsi:** DTO berfungsi sebagai pembungkus data yang bersih dan *type-safe* setelah validasi, memisahkan data *request* dari logika bisnis inti.
- **Aturan Penggunaan:**
  - **Gunakan DTO:** Ketika *request* bertujuan untuk membuat/memperbarui entitas utama dan/atau relasinya (misalnya, membuat Event baru beserta Checklist awalnya). Ini berlaku untuk *request* yang membawa lebih dari 3-4 field yang secara logis membentuk satu objek data.
  - **Jangan Gunakan DTO:** Untuk *request* yang sangat sederhana (misalnya, mengubah status sebuah tugas dari 'To-Do' menjadi 'In Progress' yang hanya mengirimkan 1-2 data). Dalam kasus ini, data yang sudah divalidasi dapat langsung dikirim ke Service.
- **Implementasi:**
  - Setiap DTO akan memiliki *method* statis `fromAppRequest(Request $request)` untuk menginstansiasi DTO dari data *request* yang telah divalidasi.
  - Contoh: `EventData::fromAppRequest($validatedRequest)`
- **Struktur DTO:**
  - DTO akan memiliki properti publik dengan tipe data yang jelas (`public readonly string $clientName`, `public readonly Carbon $eventDate`, dll.).
  - DTO akan berisi *method* untuk mengubah datanya menjadi *array* yang siap digunakan oleh Repository/Model.

4. Transformasi Data di DTO

- **Deskripsi:** DTO akan memiliki *method-method* spesifik untuk menyediakan data dalam format *array* yang dibutuhkan oleh Service. Ini memastikan Service tidak perlu tahu tentang struktur DTO, dan DTO hanya menyediakan data yang relevan untuk aksi tertentu.
- **Contoh:** Sebuah `CreateEventDTO` mungkin memiliki *method*:
    - `toEventArray(): array` -> Mengembalikan *array* data khusus untuk model `Event`.
    - `toChecklistItemsArray(): array` -> Mengembalikan *array* data untuk model `ChecklistItem` yang berelasi.
- **Konteks Tahapan:** *Method* ini akan cerdas. Jika pada tahap awal pembuatan Event, `order_number` belum ada, maka `toEventArray()` tidak akan menyertakan *key* `order_number`. Ini dikontrol di dalam DTO itu sendiri.

5. Service Layer

- **Deskripsi:** Service Class (misalnya, `EventService`) adalah tempat semua logika bisnis berada. Service tidak boleh berinteraksi langsung dengan `Request` atau `Response`. Ia menerima data (baik sebagai DTO atau *array* sederhana) dan mengorkestrasi proses.
- **Transaksi Database:** Jika sebuah proses di dalam Service melibatkan beberapa operasi tulis ke database (misalnya, membuat `Event`, lalu `ChecklistItem`, lalu mengirim notifikasi), keseluruhan proses **wajib** dibungkus dalam `DB::transaction()`. Ini menjamin konsistensi data; jika satu langkah gagal, semua langkah sebelumnya akan di-*rollback*.

6. Repository Pattern

- **Deskripsi:** Lapisan ini bertanggung jawab penuh atas interaksi dengan database. Service tidak boleh memanggil Eloquent Model secara langsung, melainkan melalui Repository.
- **Struktur:**
    - **Interface:** Mendefinisikan "kontrak" atau metode apa saja yang tersedia (misalnya, `App\Repositories\Contracts\EventRepositoryInterface`).
    - **Implementation:** Kelas yang mengimplementasikan *interface* tersebut menggunakan Eloquent (misalnya, `App\Repositories\Eloquent\EventRepository`). Di sinilah semua *query* database berada.
- **Registrasi:** Semua *binding* dari Interface ke Implementation akan didaftarkan dalam sebuah *Service Provider* khusus, misalnya `App\Providers\RepositoryServiceProvider.php`.

2.3. Aspek Teknis Lintas Modul (Cross-Cutting Concerns)

(Bagian ini tetap sama dengan versi 2.1, karena tidak terpengaruh oleh perubahan di sisi frontend)

1. Penanganan Error & Logging

- **Deskripsi:** Setiap error atau *exception* yang tidak terduga, terutama di dalam Service atau Repository, akan ditangkap (`try-catch`) dan dicatat ke dalam sistem *log* standar Laravel (`storage/logs/laravel.log`) menggunakan `Log::error($message, $context)`. Log harus menyertakan informasi yang cukup untuk *debugging*.

2. Pencatatan Aktivitas (Audit Trail)

- **Deskripsi:** Semua aksi penting yang dilakukan oleh pengguna (misalnya, membuat event, mengubah anggaran, menghapus vendor) akan dicatat untuk keperluan audit.
- **Tabel `action_logs`:**
  - `id` (Primary Key)
  - `user_id` (Foreign Key ke tabel users)
  - `action` (String, contoh: 'EVENT_CREATED', 'BUDGET_UPDATED')
  - `loggable_type` (Polymorphic Relation, contoh: `App\Models\Event`)
  - `loggable_id` (Polymorphic Relation)
  - `old_values` (JSON, nullable)
  - `new_values` (JSON, nullable)
  - `ip_address` (String)
  - `user_agent` (Text)
  - `created_at`
- **Proses Latar Belakang (Job & Queue):** Untuk menjaga performa aplikasi, proses pencatatan log ini akan dijalankan secara asinkron. Controller atau Service akan men-*dispatch* sebuah *Job* (misalnya, `LogUserActivity::dispatch(...)`) ke dalam antrean (*queue*). *Worker* di server yang menjalankan perintah `php artisan queue:work` kemudian akan memproses *job* ini dari tabel `jobs` di database dan menyimpan data ke tabel `action_logs`.

3. Mekanisme Caching

- **Deskripsi:** Caching akan diimplementasikan di dalam *Repository Implementation* untuk mempercepat *request* yang bersifat *read-only*. Driver yang digunakan adalah `file` bawaan Laravel, yang tidak memerlukan setup server tambahan.
- **Strategi:**
  - **Read:** Saat sebuah metode seperti `find($id)` atau `allForUser($userId)` dipanggil, Repository pertama-tama akan memeriksa apakah data ada di *cache*. Jika ada, kembalikan data dari *cache*. Jika tidak, ambil data dari database, simpan ke *cache* dengan *key* yang unik (misalnya, `event:{$id}`), lalu kembalikan data tersebut.
  - **Create, Update, Delete:** Setiap kali ada operasi tulis (create, update, delete), semua *cache* yang relevan dengan data yang diubah **wajib** dihapus (`Cache::forget($key)`). Ini memastikan data yang disajikan selalu segar.

4. Notifikasi Real-time dengan Laravel Reverb

- **Deskripsi:** Notifikasi akan dikirimkan secara *real-time* ke *frontend* menggunakan Laravel Reverb, server WebSocket pihak pertama dari Laravel. Pilihan ini menyederhanakan arsitektur, menghilangkan ketergantungan pada layanan pihak ketiga, dan memastikan integrasi yang mulus.
- **Alur Teknis:**
    1. Sebuah aksi di dalam **Service** (misalnya, `TaskService` meng-assign tugas) akan men-*dispatch* sebuah **Laravel Event** (misalnya, `TaskAssigned`).
    2. *Event* `TaskAssigned` ini akan mengimplementasikan *interface* `ShouldBroadcast`.
    3. Laravel, melalui Reverb, akan menyiarkan *event* ini ke sebuah *channel* privat yang telah ditentukan (misalnya, `private-user.{$userId}`).
    4. Di **Frontend**, **Laravel Echo** (yang telah dikonfigurasi untuk terhubung ke server Reverb) akan "mendengarkan" *channel* privat milik pengguna yang sedang login.
    5. Ketika *event* `TaskAssigned` diterima, Echo akan memicu sebuah *callback* di React, yang kemudian akan menampilkan notifikasi visual kepada pengguna (misalnya, menggunakan *library* seperti `react-toastify`).

BAGIAN III : ENTITIES dan MODEL

1. Model Inti: Pengguna & Acara

Ini adalah jantung dari aplikasi, tempat data paling dasar disimpan.

a. `User` (Buku Catatan Pengguna)

- **Tujuan**: Untuk menyimpan data semua orang yang bisa masuk (login) ke aplikasi, entah itu tim WO Anda atau klien.
- **Penjelasan per Kolom**:
    - `id`: Nomor identitas unik untuk setiap orang, seperti nomor KTP di sistem. Ini tidak akan pernah sama antara satu orang dengan yang lain.
    - `name`: Nama lengkap orang tersebut.
    - `email`: Alamat email yang digunakan untuk login dan untuk dihubungi.
    - `password`: Kata sandi untuk login. Di database, ini akan disimpan dalam format acak (ter-hash) demi keamanan.
    - `phone_number`: Nomor telepon yang bisa dihubungi.
    - `profile_picture_url`: Link atau alamat lokasi foto profil pengguna.

b. `Role` (Buku Catatan Peran/Jabatan)

- **Tujuan**: Untuk menentukan jabatan seseorang di dalam aplikasi (misalnya "Owner", "Planner", atau "Klien"), yang nantinya akan menentukan menu apa saja yang bisa mereka lihat dan akses.

- **Penjelasan per Kolom**:
    - `id`: Nomor identitas unik untuk setiap peran.
    - `name`: Nama perannya, contoh: "Lead Planner".
    - `description`: Penjelasan singkat tentang tugas peran tersebut.

c. `Event` (Buku Catatan Acara Pernikahan)

- **Tujuan**: Ini adalah "folder utama" untuk setiap proyek pernikahan. Semua data lain seperti anggaran, vendor, dan tugas akan terhubung ke sini.
- **Penjelasan per Kolom**:
    - `id`: Nomor identitas unik untuk setiap acara.
    - `client_id`: Penanda (berisi `id` dari tabel `User`) untuk menunjukkan siapa klien yang punya acara ini.
    - `lead_planner_id`: Penanda (berisi `id` dari tabel `User`) untuk menunjukkan siapa planner yang bertanggung jawab atas acara ini.
    - `event_name`: Nama acara, contoh: "Pernikahan Budi & Ani".
    - `event_date`: Tanggal berlangsungnya acara pernikahan.
    - `venue_name`: Nama lokasi atau gedung tempat acara.
    - `venue_address`: Alamat lengkap lokasi acara.
    - `status`: Status progres acara saat ini, misalnya "Perencanaan", "Selesai", atau "Dibatalkan".

---

2. Model Manajemen Vendor

a. `Vendor` (Buku Alamat Vendor)

- **Tujuan**: Untuk menyimpan semua data kontak vendor rekanan Anda (fotografer, katering, MUA, dll).
- **Penjelasan per Kolom**:
    - `id`: Nomor unik untuk setiap vendor.
    - `name`: Nama vendor, contoh: "Cahaya Katering".
    - `category`: Jenis layanan vendor, contoh: "Katering", "Dekorasi".
    - `contact_person`: Nama orang yang bisa dihubungi di vendor tersebut.
    - `phone_number` & `email`: Kontak vendor.
    - `address`: Alamat kantor vendor.
    - `portfolio_url`: Link ke portofolio online mereka.
    - `internal_notes`: Catatan rahasia untuk tim WO Anda tentang vendor ini (misal: "kerjanya bagus, tapi suka telat"). Klien tidak bisa melihat ini.

b. `EventVendor` (Catatan Penghubung Acara & Vendor)

- **Tujuan**: Ini adalah "jembatan" yang menghubungkan sebuah acara dengan vendor yang dipesan. Di sinilah detail pemesanan disimpan.

- **Penjelasan per Kolom**:
    - `event_id`: Menunjuk ke acara mana.
    - `vendor_id`: Menunjuk ke vendor mana yang dipesan.
    - `contract_document_url`: Tempat menyimpan file kontrak dengan vendor.
    - `total_price`: Harga total kesepakatan dengan vendor.
    - `amount_paid`: Jumlah uang yang sudah dibayarkan ke vendor.
    - `payment_status`: Status pembayaran ke vendor (misal: "DP Dibayar", "Lunas").

---

3. Model Perencanaan & Kolaborasi

a. `ChecklistItem` (Daftar Tugas / To-Do List)

- **Tujuan**: Untuk mencatat semua tugas yang harus dikerjakan untuk sebuah acara.
- **Penjelasan per Kolom**:
    - `event_id`: Tugas ini milik acara yang mana.
    - `assignee_id`: Siapa (dari tabel `User`) yang bertanggung jawab mengerjakan tugas ini.
    - `title`: Nama tugasnya, contoh: "Booking Gedung".
    - `due_date`: Tanggal paling lambat tugas ini harus selesai.
    - `status`: Status pengerjaan tugas ("Belum Dikerjakan", "Sedang Dikerjakan", "Selesai").
    - `assigned_to`: Penanda apakah ini tugas untuk tim "WO" atau untuk "Klien".

b. `Guest` (Buku Tamu Undangan)

- **Tujuan**: Untuk mengelola daftar tamu yang diundang oleh klien.
- **Penjelasan per Kolom**:
    - `event_id`: Daftar tamu ini untuk acara yang mana.
    - `name`: Nama tamu yang diundang.
    - `category`: Pengelompokan tamu, contoh: "Keluarga Pria", "Teman Kantor Wanita".
    - `rsvp_status`: Status konfirmasi kehadiran tamu ("Hadir", "Tidak Hadir").
    - `special_needs`: Catatan kebutuhan khusus tamu, misal: "Alergi udang".
    - `table_number`: Jika pakai denah duduk, tamu ini akan duduk di meja nomor berapa.

---

4. Model Finansial

a. `BudgetItem` (Rincian Anggaran)

- **Tujuan**: Untuk mencatat rincian anggaran acara, baris per baris, seperti di tabel Excel.
- **Penjelasan per Kolom**:
    - `item_name`: Nama pengeluaran, contoh: "Sewa Gaun Pengantin".
    - `estimated_budget`: Perkiraan biaya awal.
    - `final_price`: Harga deal atau biaya final setelah negosiasi.
    - `amount_paid`: Jumlah yang sudah dibayarkan untuk item ini.
    - `status`: Status pembayaran item ini ("Lunas", "Belum Lunas").

b. `Invoice` (Tagihan untuk Klien)

- **Tujuan**: Untuk membuat dan melacak tagihan (invoice) yang Anda kirimkan ke klien.
- **Penjelasan per Kolom**:
    - `invoice_number`: Nomor unik untuk setiap tagihan.
    - `issue_date`: Tanggal tagihan dibuat.
    - `due_date`: Tanggal jatuh tempo pembayaran.
    - `total_amount`: Total nilai tagihan.
    - `status`: Status tagihan ("Sudah Dikirim", "Sudah Dibayar", "Telat Bayar").
    - `payment_proof_url`: Tempat klien mengunggah bukti transfer.

---

5. Model Pendukung & Lanjutan

a. `Document` (Lemari Arsip Digital)

- **Tujuan**: Untuk menyimpan semua jenis file (KTP, Kontrak, Gambar Inspirasi). Model ini canggih karena bisa "menempel" ke banyak model lain.
- **Penjelasan per Kolom**:
    - `documentable_id` & `documentable_type`: Dua kolom ini bekerja sama untuk menandai "dokumen ini milik siapa?". Contoh: jika `documentable_id`=10 dan `documentable_type`='Event', artinya ini adalah dokumen milik acara dengan ID 10.
    - `file_name` & `file_path`: Nama dan lokasi penyimpanan file-nya.
    - `file_type`: Jenis dokumen, misal: "KTP", "Kontrak".

b. `ActionLog` (Catatan Aktivitas / CCTV Digital)

- **Tujuan**: Untuk mencatat semua kejadian penting yang dilakukan pengguna, seperti "Si A mengubah anggaran pada jam sekian". Ini penting untuk keamanan dan audit.
- **Penjelasan per Kolom**:
    - `user_id`: Siapa yang melakukan aksi.
    - `action`: Aksi apa yang dilakukan, contoh: "HAPUS_VENDOR".
    - `loggable_id` & `loggable_type`: Mirip seperti model `Document`, ini menandai data apa yang diubah.

    - `old_values` & `new_values`: Mencatat data sebelum dan sesudah diubah.

c. `Lead` (Daftar Calon Klien)

- **Tujuan**: Untuk mencatat data orang-orang yang baru bertanya-tanya atau calon klien yang belum deal.
- **Penjelasan per Kolom**:
    - `source`: Dari mana calon klien ini tahu tentang Anda, misal: "Instagram", "Pameran".
    - `status`: Statusnya saat ini, misal: "Baru Masuk", "Sudah Dihubungi", "Deal".