Laravel Event Site Implementation Plan

This implementation plan outlines the steps for creating a Laravel backend system to showcase yearly events. The system will allow an admin user to update event content every year, while previous year's content will be stored and accessible for reference. The implementation plan includes database design, model relationships, admin dashboard setup, API routes, and frontend integration.

Step

1. Events Table: Holds the main event details, including event_id, year, created_at, and updated_at.
2. Details Tables: Separate tables for each event detail (e.g., summary, themes, programme, speakers).

Each table has the columns: id, event_id (foreign key), detail-specific column (e.g., summary, theme), created_at, updated_at.

3. Example tables: EventSummary, EventTheme, EventProgramme, EventResource, Speaker, FAQ, Media, Sponsor, Gallery, Attendance.

Step 2: Create Migration Files

Create migration files for the events and details tables using the following artisan commands:

- php artisan make:migration create_events_table
- php artisan make:migration create_event_summaries_table
- ...

Then run: php artisan migrate

Step 3: Model Relationships

1. Event Model: Define relationships for each of the details tables (one-to-many).

2. Example relationship in Event Model: public function summaries() { return $this->hasMany(EventSummary::class); }

3. Define inverse relationships in the detail models (e.g., EventSummary).

Step 4: Admin Dashboard for Content Management

1. Use Laravel Sanctum or Passport for API-based authentication.

2. Implement CRUD functionality for the admin to manage content (summary, themes, etc.).

3. Use Laravel Nova or custom views for the admin dashboard.

4. Allow the admin to update the current year's event details.

Step 5: API Routes

1. Create public API routes to fetch event data using Laravel's resource controllers.

2. Example route: Route::resource('events', EventController::class);

3. Secure admin routes using API authentication.

Step 6: Frontend Implementation

1. Display the current event's details on the homepage.

2. Allow users to browse past events by year (using a dropdown or list).

3. Fetch past events using API calls.

Step 7: Media and File Handling

1. Use Laravel's file storage system for media uploads (e.g., images, PDFs, videos).

2. Store files in separate directories for each event.

3. Example code for file upload: $filePath = $request->file('image')->store('events/'.$event->id);

Step 8: Testing and Finalization

1. Test CRUD operations, file uploads, and API routes.

2. Deploy the application to your server or cloud provider.