GiveMeALift CMS - Technical Documentation

Version: 1.0
Date: August 2025
Organization: GiveMeALift NGO
Developer: The Y4NN
Document Type: Software Requirements Specification (SRS)

---

1. Introduction

1.1 Purpose

This document provides a comprehensive technical overview of the proposed GiveMeALift Content Management System (CMS), a custom solution designed to replace the existing WordPress website. The CMS aims to address critical issues including poor design implementation, limited visitor/donor analytics, inadequate SEO performance, and suboptimal system performance.

1.2 Scope

The GiveMeALift CMS is a purpose-built web application that combines a Laravel backend with a modern frontend framework to deliver a high-performance, secure, and user-friendly platform for NGO operations. The system includes comprehensive content management capabilities, advanced analytics, and robust security features.

1.3 Document Overview

This documentation covers system architecture, technology stack, feature specifications, data models, security considerations, and deployment guidelines. It serves as a reference for developers, project managers, and technical stakeholders involved in the development and maintenance of the system.

---

2. System Architecture

2.1 High-Level Architecture

The GiveMeALift CMS follows a modern three-tier architecture pattern:

```
┌─────────────────┐    ┌─────────────────┐    ┌─────────────────┐
│   Presentation  │    │   Application   │    │      Data       │
│     Layer       │    │     Layer       │    │     Layer       │
│                 │    │                 │    │                 │
│  Vue.js/React   │◄──►│  Laravel API    │◄──►│  PostgreSQL     │
│  Frontend       │    │  Backend        │    │  Database       │
│                 │    │                 │    │                 │
└─────────────────┘    └─────────────────┘    └─────────────────┘
```

2.2 System Components

Frontend Layer:

- Modern JavaScript framework (Vue.js or React)
- Responsive design implementation of Figma mockups
- Progressive Web App (PWA) capabilities
- Client-side routing and state management

Backend Layer:

- PHP Laravel framework (v12+)
- RESTful API architecture
- Authentication and authorization services
- Business logic processing

Database Layer:

- PostgreSQL relational database
- Optimized schema design with advanced data types
- Data integrity constraints and ACID compliance
- Performance indexing and query optimization

2.3 Integration Points

- **External APIs:** Payment gateways, email services, social media platforms
- **Third-party Services:** Google Analytics, SEO tools, CDN services
- **Admin Dashboard:** Real-time data synchronization with frontend

---

3. Frontend Technology

3.1 Technology Selection

Selected Framework: Vue.js 3 with Composition API

Key Advantages:

- Seamless Laravel ecosystem integration
- Intuitive template syntax and gentle learning curve
- Optimal bundle size for NGO website performance
- Excellent TypeScript support for scalability
- Strong French developer community support
- Built-in state management with Pinia

3.2 Figma Design Integration

Implementation Strategy:

1. **Design System Translation:** Convert Figma design tokens (colors, typography, spacing) into CSS custom properties
2. **Component Library:** Build reusable Vue/React components matching Figma components exactly
3. **Responsive Implementation:** Implement mobile-first responsive design based on Figma breakpoints
4. **Asset Optimization:** Export and optimize images, icons, and graphics from Figma for web delivery

Key Integration Features:

- Pixel-perfect design implementation
- Consistent spacing and typography system
- Optimized image delivery and lazy loading
- Cross-browser compatibility testing

3.3 Performance Optimization

- Code splitting and lazy loading
- Image optimization and WebP format support
- CSS purging and minification
- Service worker implementation for offline functionality

---

4. Backend Technology

4.1 Laravel Framework Architecture

Version: Laravel 10.x (LTS)

Core Components:

- **Eloquent ORM:** Database abstraction and relationship management

- **Artisan CLI:** Development tools and automation
- **Middleware Pipeline:** Request processing and security
- **Service Container:** Dependency injection and IoC
- **Task Scheduling:** Automated maintenance and reporting

4.2 API Design

RESTful Architecture with Laravel API Resources:

API Endpoint Structure:

```
Base URL: https://api.givemealift.org/v1/

Authentication Endpoints:
├── POST /auth/login
├── POST /auth/logout
├── POST /auth/refresh
└── GET  /auth/user

Content Management Endpoints:
├── GET    /events                  (List events with pagination)
├── POST   /events                  (Create new event)
├── GET    /events/{id}             (Get specific event)
├── PUT    /events/{id}             (Update event)
├── DELETE /events/{id}             (Delete event)
├── GET    /testimonials            (List testimonials)
├── POST   /testimonials            (Create testimonial)
├── PUT    /testimonials/{id}       (Update testimonial)
├── GET    /gallery/albums          (List gallery albums)
├── POST   /gallery/albums          (Create album)
├── GET    /media                   (List media files)
└── POST   /media/upload            (Upload media)

Analytics Endpoints:
├── GET /analytics/visitors         (Visitor statistics)
├── GET /analytics/donors           (Donor analytics)
├── GET /analytics/dashboard        (Dashboard summary)
└── GET /analytics/reports          (Generate reports)
```

API Response Format:

```
{
  "success": true,
  "data": {
    // Response data
  },
  "meta": {
    "pagination": {
      "current_page": 1,
      "total_pages": 10,
      "per_page": 15,
      "total": 150
    }
  },
  "message": "Operation successful"
}
```

Error Response Format:

```
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "The given data was invalid.",
    "details": {
      "title": ["The title field is required."],
      "email": ["The email must be a valid email address."]
    }
  }
}
```

Authentication:

- Laravel Sanctum for SPA authentication
- Bearer token authentication for API requests
- Role-based access control (RBAC) middleware
- Rate limiting: 60 requests per minute for authenticated users
- API versioning strategy (v1, v2) for backward compatibility

4.3 Database Integration

- **PostgreSQL Advantages:** Advanced data types (JSON, Arrays), full-text search, and superior performance
- **Query Optimization:** Eloquent query optimization with PostgreSQL-specific features
- **Migration System:** Version-controlled database schema management with PostgreSQL support
- **Seeding:** Automated test data generation with PostgreSQL data types

- **Backup Strategy:** Automated daily database backups using pg_dump

---

5. Admin Dashboard Features

5.1 Content Management

5.1.1 Media Management

Core Functionality:

- Upload, organize, and manage digital assets
- Image optimization and multiple format support
- Bulk upload capabilities with progress tracking
- Folder structure organization
- Search and filtering capabilities
- Usage tracking across content

Features:

- Image resizing and cropping tools
- Alt text and SEO metadata management
- CDN integration for global delivery
- Storage quota monitoring

5.1.2 Events Management

CRUD Operations:

- Create, read, update, and delete event records
- Rich text editor for event descriptions
- Image gallery attachment
- Event categorization and tagging
- Publication scheduling

Advanced Features:

- Event calendar view
- Registration tracking
- Email notifications
- Social media integration
- Analytics tracking

5.1.3 Testimonials Management

Content Features:

- Rich text testimonial content
- Author information and photos
- Star rating system
- Publication status controls
- Featured testimonial selection

Display Options:

- Multiple layout templates
- Carousel and grid display modes
- Social proof integration
- Automated testimonial requests

5.1.4 Gallery Management

Organization Features:

- Album creation and management
- Bulk image operations
- Drag-and-drop reordering
- Image metadata management
- Lightbox gallery displays

5.2 User Management

5.2.1 Role-Based Access Control

User Roles:

- **Super Admin:** Full system access and user management
- **Content Manager:** Content creation and editing privileges
- **Editor:** Content editing and publishing rights
- **Viewer:** Read-only access to dashboard analytics

5.2.2 Permission System

- Granular permission assignment
- Module-based access control
- Activity logging and audit trails
- Password policy enforcement
- Two-factor authentication support

5.3 Visitor/Donor Overview

5.3.1 Analytics Dashboard

Visitor Metrics:

- Real-time visitor tracking
- Geographic distribution maps
- Traffic source analysis
- Page performance metrics
- User behavior flow

Donor Analytics:

- Donation tracking and trends
- Donor lifetime value calculations
- Campaign performance metrics
- Retention rate analysis
- Monthly recurring donation tracking

5.3.2 Reporting Features

- Customizable date range reports
- Export functionality (PDF, CSV, Excel)
- Automated weekly/monthly reports
- Goal tracking and progress visualization
- Comparative analysis tools

5.4 Message View

5.4.1 Contact Management

Message Processing:

- Centralized inbox for all website inquiries
- Message categorization and labeling
- Response tracking and templates
- Priority assignment system
- Automated acknowledgment emails

Communication Features:

- Internal message notes and collaboration
- Email integration and threading
- Response time tracking
- Follow-up reminders
- Archive and search functionality

6. Data Model & Class Diagram

6.1 Design Rationale

The data model is designed with scalability, performance, and maintainability in mind. The following decisions were made to support the system's goals:

- **PostgreSQL as a Strategic Choice:** PostgreSQL was selected for its advanced features like `JSONB`, `INET`, and custom `ENUM` types.
    - `JSONB` provides flexibility for storing semi-structured data like metadata (`meta_data`, `settings`) without compromising indexing and query performance.
    - Custom `ENUM` types (`event_status_enum`, `payment_status_enum`, etc.) enforce data integrity at the database level, ensuring that status fields contain only valid values.
- **Dedicated Analytics Tables:** `VisitorAnalytic` and `DonationAnalytic` are kept separate from operational tables. This separation prevents performance degradation on the main application tables during high-volume analytics writes and allows for independent scaling and optimization of analytics data.
- **Normalized Core Entities:** Core entities like `User`, `Event`, and `GalleryAlbum` are normalized to reduce data redundancy and improve data integrity. Foreign key constraints are used to maintain referential integrity.
- **Soft Deletes and Status Tracking:** The use of status fields (e.g., `event_status_enum`) and boolean flags (`is_active`) allows for soft deletes and content lifecycle management (e.g., drafts, published, archived) without permanently losing data.

6.2 UML Class Diagram

```
┌─────────────────────────────────────┐
│                User                 │
├─────────────────────────────────────┤
│ - id: SERIAL (PK)                   │
│ - name: VARCHAR(255)                │
│ - email: VARCHAR(255) UNIQUE        │
│ - email_verified_at: TIMESTAMP      │
│ - password: VARCHAR(255)            │
│ - role: ENUM(admin,editor,viewer)   │
│ - is_active: BOOLEAN                │
│ - last_login_at: TIMESTAMP          │
│ - created_at: TIMESTAMP             │
│ - updated_at: TIMESTAMP             │
└─────────────────────────────────────┘
              │ 1
              │ creates/updates
              │ *
┌─────────────────────────────────────┐
│               Event                 │
├─────────────────────────────────────┤
│ - id: SERIAL (PK)                   │
│ - title: VARCHAR(255)               │
│ - slug: VARCHAR(255) UNIQUE         │
│ - description: TEXT                 │
│ - short_description: VARCHAR(500)   │
│ - featured_image_id: INTEGER (FK)   │
│ - start_date: TIMESTAMP             │
│ - end_date: TIMESTAMP               │
│ - location: VARCHAR(255)            │
│ - category_id: INTEGER (FK)         │
│ - status: event_status_enum         │
│ - meta_data: JSONB                  │
│ - created_by: INTEGER (FK)          │
│ - updated_by: INTEGER (FK)          │
│ - created_at: TIMESTAMP             │
│ - updated_at: TIMESTAMP             │
└─────────────────────────────────────┘
              │ *
              │
              │ *
┌─────────────────────────────────────┐
│            Category                 │
├─────────────────────────────────────┤
│ - id: SERIAL (PK)                   │
│ - name: VARCHAR(100)                │
│ - slug: VARCHAR(100) UNIQUE         │
│ - description: TEXT                 │
│ - color: VARCHAR(7)                 │
│ - icon: VARCHAR(50)                 │
│ - created_at: TIMESTAMP             │
│ - updated_at: TIMESTAMP             │
└─────────────────────────────────────┘

┌─────────────────────────────────────┐
│            Testimonial              │
├─────────────────────────────────────┤
│ - id: SERIAL (PK)                   │
│ - name: VARCHAR(100)                │
│ - email: VARCHAR(255)               │
│ - position: VARCHAR(100)            │
│ - organization: VARCHAR(100)        │
│ - content: TEXT                     │
│ - rating: SMALLINT CHECK (1-5)      │
│ - avatar_id: INTEGER (FK)           │
│ - is_featured: BOOLEAN DEFAULT false│
│ - status: testimonial_status_enum   │
│ - display_order: INTEGER            │
│ - approved_by: INTEGER (FK)         │
│ - created_at: TIMESTAMP             │
│ - updated_at: TIMESTAMP             │
└─────────────────────────────────────┘

┌─────────────────────────────────────┐
│            GalleryAlbum             │
├─────────────────────────────────────┤
│ - id: SERIAL (PK)                   │
│ - title: VARCHAR(255)               │
│ - description: TEXT                 │
│ - slug: VARCHAR(255) UNIQUE         │
│ - cover_image_id: INTEGER (FK)      │
│ - is_featured: BOOLEAN              │
│ - display_order: INTEGER            │
│ - settings: JSONB                   │
│ - created_at: TIMESTAMP             │
│ - updated_at: TIMESTAMP             │
└─────────────────────────────────────┘
              │ 1
              │
              │ *
┌─────────────────────────────────────┐
│           GalleryImage              │
├─────────────────────────────────────┤
│ - id: SERIAL (PK)                   │
│ - album_id: INTEGER (FK)            │
│ - media_id: INTEGER (FK)            │
│ - caption: TEXT                     │
│ - alt_text: VARCHAR(255)            │
│ - display_order: INTEGER            │
│ - metadata: JSONB                   │
│ - created_at: TIMESTAMP             │
│ - updated_at: TIMESTAMP             │
└─────────────────────────────────────┘
              │ *
              │
              │ 1
┌─────────────────────────────────────┐
│              Media                  │
├─────────────────────────────────────┤
│ - id: SERIAL (PK)                   │
│ - filename: VARCHAR(255)            │
│ - original_name: VARCHAR(255)       │
│ - mime_type: VARCHAR(100)           │
│ - size: BIGINT                      │
│ - dimensions: JSONB                 │
│ - alt_text: VARCHAR(255)            │
│ - caption: TEXT                     │
│ - folder_id: INTEGER (FK)           │
│ - storage_path: VARCHAR(500)        │
│ - cdn_url: VARCHAR(500)             │
│ - created_at: TIMESTAMP             │
│ - updated_at: TIMESTAMP             │
└─────────────────────────────────────┘

┌─────────────────────────────────────┐
│          VisitorAnalytic            │
├─────────────────────────────────────┤
│ - id: SERIAL (PK)                   │
│ - session_id: VARCHAR(255)          │
│ - ip_address: INET                  │
│ - user_agent: TEXT                  │
│ - page_url: VARCHAR(500)            │
│ - referrer: VARCHAR(500)            │
│ - country_code: CHAR(2)             │
│ - city: VARCHAR(100)                │
│ - device_type: device_enum          │
│ - visit_duration: INTEGER           │
│ - page_views: INTEGER               │
│ - bounce_rate: DECIMAL(5,2)         │
│ - visited_at: TIMESTAMP             │
└─────────────────────────────────────┘

┌─────────────────────────────────────┐
│         DonationAnalytic            │
```

```
├────────────────────────────────────┤
│ - id: SERIAL (PK)                  │
│ - donor_name: VARCHAR(255)         │
│ - email: VARCHAR(255)              │
│ - phone: VARCHAR(20)               │
│ - amount: DECIMAL(10,2)            │
│ - currency: CHAR(3)                │
│ - donation_type: donation_type_enum │
│ - campaign_id: INTEGER (FK)        │
│ - payment_method: VARCHAR(50)      │
│ - payment_reference: VARCHAR(100)  │
│ - status: payment_status_enum      │
│ - donor_metadata: JSONB            │
│ - created_at: TIMESTAMP            │
│ - updated_at: TIMESTAMP            │
└────────────────────────────────────┘
```

PostgreSQL Custom Types:

```
-- Custom ENUM types for better data integrity
CREATE TYPE event_status_enum AS ENUM ('draft', 'published', 'archived', 'cancelle
CREATE TYPE testimonial_status_enum AS ENUM ('pending', 'approved', 'rejected');
CREATE TYPE device_enum AS ENUM ('desktop', 'mobile', 'tablet');
CREATE TYPE donation_type_enum AS ENUM ('one_time', 'monthly', 'yearly');
CREATE TYPE payment_status_enum AS ENUM ('pending', 'completed', 'failed', 'refun
```

6.3 Entity Specifications

6.3.1 Events Entity

```
Events Table:
├── id (Primary Key, Auto-increment)
├── title (VARCHAR(255), Required)
├── slug (VARCHAR(255), Unique, SEO-friendly)
├── description (TEXT, Rich content)
├── short_description (VARCHAR(500))
├── featured_image (VARCHAR(255), Foreign key to media)
├── start_date (DATETIME, Required)
├── end_date (DATETIME, Nullable)
├── location (VARCHAR(255))
├── category_id (Foreign Key to categories)
├── status (ENUM: draft, published, archived)
├── meta_title (VARCHAR(60), SEO)
├── meta_description (VARCHAR(160), SEO)
├── created_at (TIMESTAMP)
├── updated_at (TIMESTAMP)
├── created_by (Foreign Key to users)
└── updated_by (Foreign Key to users)

Relationships:
├── belongsToMany(images) - Gallery images
├── belongsTo(category) - Event category
├── belongsTo(creator) - User who created
└── hasMany(registrations) - Event registrations
```

6.3.2 Testimonials Entity

```
Testimonials Table:
├── id (Primary Key, Auto-increment)
├── name (VARCHAR(100), Required)
├── email (VARCHAR(255), Nullable)
├── position (VARCHAR(100), Nullable)
├── organization (VARCHAR(100), Nullable)
├── content (TEXT, Required)
├── rating (TINYINT, 1-5 scale)
├── avatar (VARCHAR(255), Foreign key to media)
├── is_featured (BOOLEAN, Default: false)
├── status (ENUM: pending, approved, rejected)
├── display_order (INTEGER, For sorting)
├── created_at (TIMESTAMP)
├── updated_at (TIMESTAMP)
└── approved_by (Foreign Key to users)

Relationships:
├── belongsTo(approver) - Admin who approved
└── belongsTo(avatarImage) - Profile image
```

6.3.3 Gallery Entity

```
Gallery Albums Table:
├── id (Primary Key, Auto-increment)
├── title (VARCHAR(255), Required)
├── description (TEXT, Nullable)
├── slug (VARCHAR(255), Unique)
├── cover_image (Foreign Key to media)
├── is_featured (BOOLEAN, Default: false)
├── display_order (INTEGER)
├── created_at (TIMESTAMP)
└── updated_at (TIMESTAMP)

Gallery Images Table:
├── id (Primary Key, Auto-increment)
├── album_id (Foreign Key to albums)
├── image_id (Foreign Key to media)
├── caption (TEXT, Nullable)
├── alt_text (VARCHAR(255), SEO)
├── display_order (INTEGER)
├── created_at (TIMESTAMP)
└── updated_at (TIMESTAMP)

Relationships:
├── Album hasMany(images)
├── Image belongsTo(album)
└── Image belongsTo(mediaFile)
```

6.4 Supporting Entities

6.4.1 Media Management

```
Media Table:
├── id (Primary Key)
├── filename (VARCHAR(255))
├── original_name (VARCHAR(255))
├── mime_type (VARCHAR(100))
├── size (BIGINT, Bytes)
├── dimensions (JSON, Width/Height)
├── alt_text (VARCHAR(255))
├── caption (TEXT)
├── folder_id (Foreign Key, Nullable)
├── created_at (TIMESTAMP)
└── updated_at (TIMESTAMP)
```

6.4.2 Analytics Tracking

```
Visitor Analytics Table:
├── id (Primary Key)
├── session_id (VARCHAR(255))
├── ip_address (VARCHAR(45))
├── user_agent (TEXT)
├── page_url (VARCHAR(500))
├── referrer (VARCHAR(500))
├── country (VARCHAR(100))
├── city (VARCHAR(100))
├── visit_duration (INTEGER, Seconds)
└── visited_at (TIMESTAMP)

Donation Analytics Table:
├── id (Primary Key)
├── donor_name (VARCHAR(255))
├── email (VARCHAR(255))
├── amount (DECIMAL(10,2))
├── currency (CHAR(3))
├── donation_type (ENUM: one-time, recurring)
├── campaign_id (Foreign Key, Nullable)
├── payment_method (VARCHAR(50))
├── status (ENUM: pending, completed, failed)
└── created_at (TIMESTAMP)
```

7. Security Considerations

7.1 Authentication & Authorization

Multi-layer Security:

- Laravel Sanctum for secure API authentication
- CSRF protection for all forms
- Rate limiting on login attempts
- Password hashing using bcrypt
- Optional two-factor authentication (2FA)

7.2 Data Protection

Security Measures:

- Input validation and sanitization
- SQL injection prevention through Eloquent ORM
- XSS protection with content security policies

- File upload restrictions and validation
- Secure session management

7.3 Infrastructure Security

Server Security:

- HTTPS enforcement with SSL certificates
- Regular security updates and patches
- Database access restrictions
- Server firewall configuration
- Automated backup encryption

7.4 Compliance & Privacy

- GDPR compliance for donor data handling
- Privacy policy implementation
- Data retention policies
- Right to erasure functionality
- Audit logging for sensitive operations

---

8. Performance Metrics

8.1 Frontend Performance

Target Metrics:

- **First Contentful Paint (FCP):** < 1.5 seconds
- **Largest Contentful Paint (LCP):** < 2.5 seconds
- **Cumulative Layout Shift (CLS):** < 0.1
- **Time to Interactive (TTI):** < 3.5 seconds
- **Mobile Page Speed Score:** > 90

8.2 Backend Performance

Optimization Targets:

- **API Response Time:** < 200ms average
- **Database Query Time:** < 50ms average
- **Cache Hit Ratio:** > 85%
- **Server Response Time:** < 100ms
- **Concurrent User Capacity:** 1000+ users

8.3 SEO Performance

SEO Enhancements:

- Dynamic meta tag generation
- XML sitemap automation
- Schema.org structured data
- Open Graph protocol implementation
- Optimized URL structure
- Image lazy loading and optimization

8.4 Monitoring & Analytics

Performance Tracking:

- Real-time server monitoring
- Application performance monitoring (APM)
- Error tracking and alerting
- User behavior analytics
- Conversion rate tracking

---

9. Deployment

9.1 Development Environment

Local Setup:

- Docker containerization for consistent environments
- Laravel Sail for simplified development
- Automated testing pipeline
- Code quality tools (PHPStan, ESLint)

9.2 Production Deployment

Infrastructure Requirements:

- **Server Specifications:** 4GB RAM minimum, 2 CPU cores
- **Web Server:** Nginx with PHP-FPM
- **Database:** PostgreSQL 14+ with optimized configuration
- **Caching:** Redis for session and application caching
- **CDN:** CloudFlare or AWS CloudFront for asset delivery

9.3 Deployment Strategy

Continuous Integration/Deployment:

1. **Code Repository:** Git-based version control
2. **Automated Testing:** Unit, integration, and feature tests
3. **Build Pipeline:** Asset compilation and optimization
4. **Staging Environment:** Pre-production testing
5. **Production Deployment:** Zero-downtime deployment with rollback capability

9.4 Maintenance & Updates

Ongoing Maintenance:

- Regular security updates
- Database optimization
- Performance monitoring
- Backup verification
- Content audit and cleanup

Conclusion

The GiveMeALift CMS represents a comprehensive solution to address the limitations of the current WordPress implementation. By leveraging modern web technologies and best practices, the system will provide:

- **Enhanced User Experience:** Fast, responsive, and intuitive interface
- **Robust Content Management:** Streamlined content creation and management workflows
- **Advanced Analytics:** Comprehensive visitor and donor insights
- **Improved SEO Performance:** Optimized for search engine visibility
- **Scalable Architecture:** Built to grow with organizational needs
- **Security-First Approach:** Enterprise-level security measures

This technical documentation serves as the foundation for development, ensuring all stakeholders have a clear understanding of the system's capabilities and implementation approach.

Document Status: Draft v1.0
Next Review Date: September 2025
Contact: Y4NN, Software Developer