Technical Documentation

Bethel Family Choir Website

Complete developer & content editor reference for the Bethel Family Choir web platform.

VERSION	STACK	COMPANY	DEVELOPER	EMAIL	LOCATION
1.0.0	PHP · MySQL · JS	Tris Tech Hub	Shimo Sezerano Second Adelin	sezeranoshimo@gmail.com	Rwanda, Africa

TABLE OF CONTENTS

01 System Overview

The **Bethel Family Choir** website is a full-stack content management platform built for a Christian choir organization based in **Huye, Rwanda**. The choir was founded in **2005** at Groupe Scolaire Officiel de Butare (Indatwa n' Inkesha School).

Key Features

FEATURE	DESCRIPTION
Public Website	Responsive multi-page site with hero sliders, gallery, history, services, and contact info
Admin Dashboard	Secure login with full CRUD operations for all website content
Media Management	Upload and manage images and YouTube video embeds
Dynamic Content	All content loaded via AJAX — no page refreshes needed

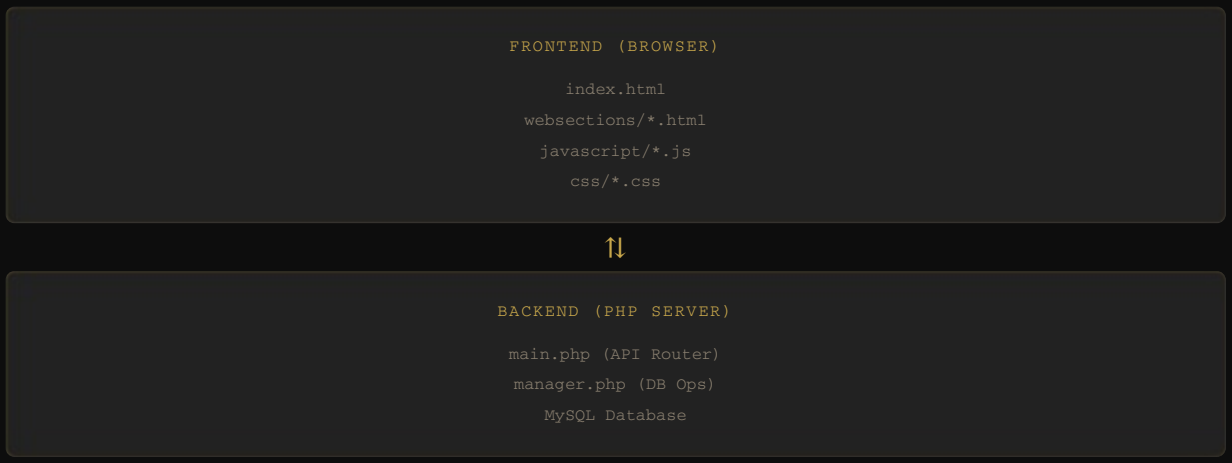
Responsive Design	Optimized for desktop, tablet, and mobile

Technology Stack

LAYER	TECHNOLOGY	VERSION
Frontend	HTML5, CSS3, JavaScript (ES6+ Modules)	—
Backend	PHP	8.0+
Database	MySQL	5.7+
Icons	Font Awesome	6.5.2
Fonts	Playfair Display, Inter (Google Fonts)	—
API Style	RESTful over AJAX/Fetch	—

02 Architecture

High-Level Architecture



Data Flow

```
User Action (click / form submit)
    |
    ▼
JavaScript Event Handler
    |  fetch() API call
    ▼
admin/backend/main.php    ◄── HTTP Request ($_GET['action'])
    |
    ▼
manager.php               ◄── SQL Query (prepared statements)
    |
    ▼
MySQL Database
    |
    ▼
JSON Response  →  DOM Update (UI refresh)
```

Directory Structure

```
bethel-family-choir/
├── index.html                # Main homepage
│
├── admin/                    # Admin panel
│   ├── login.html
│   ├── dashboard.html
│   ├── login.css
│   ├── css/
│   │   └── admin.css
│   ├── javascript/
│   │   ├── dashboard.js      # 650+ lines
│   │   └── login.js
│   └── backend/
│       ├── api.js            # Frontend API client
│       ├── main.php          # API router 500+ lines
│       └── manager.php       # DB operations 800+ lines
│
├── websections/              # Public page sections
│   ├── about.html
│   ├── service.html
│   ├── gallery.html
│   └── history.html
│
├── css/
│   ├── index.css             # Main import file
│   ├── initial.css           # Core styles ~1000 lines
│   └── initial2.css
│
├── javascript/
│   ├── main.js               # Navigation & utilities
│   ├── home.js
│   ├── about.js
│   ├── service.js
│   ├── gallery.js
│   ├── history.js
│   └── api/
│       └── renderApi.js      # API calls & rendering
│
└── media/
    ├── icons/
    ├── placeholders/
    └── [uploaded content]
```

03 JavaScript Modules

main.js — Core Navigation

Handles the mobile navigation drawer and core utility event listeners used across all pages.

FUNCTION	DESCRIPTION

`openNav()`	Opens the mobile navigation drawer
`closeNav()`	Closes the mobile navigation drawer
Event Listeners	Attaches click handlers for nav toggle buttons

home.js — Homepage Functionality

FUNCTION	DESCRIPTION
`showSlides()`	Rotates hero slider images every 7 seconds automatically
`animateCounter(element, target)`	Animates stat counters from 0 up to their target number
`initStatsObserver()`	Uses `IntersectionObserver` to trigger counters when visible in viewport
`renderHomePage()`	Main init function — calls all home page data loaders

> NOTE
>
> The stats counters display: **5+ songs**, **60+ members**, **200+ alumni**. These are animated only once per page load when the section scrolls into view.

gallery.js — Gallery & Lightbox

FUNCTION	DESCRIPTION
`openViewer(index)`	Opens image lightbox at the specified array index
`showImage(index)`	Displays image with fade-in transition
`closeViewer()`	Closes lightbox with fade-out effect
`nextImage()`	Navigate to next image (wraps around at end)
`prevImage()`	Navigate to previous image (wraps around)
`renderGallery_2()`	Fetches images from API and renders the gallery grid

The lightbox supports **keyboard navigation**: `Escape` to close, `←` / `→` arrow keys to navigate, and clicking outside the image also closes the viewer.

renderApi.js — Central API Layer

This is the main bridge between the frontend pages and the PHP backend. All data fetching goes through this file.

DATA FETCHING FUNCTIONS

FUNCTION	API ENDPOINT	DESCRIPTION
`Slideshow()`	`Slideshow`	Loads hero background images for root-level pages
`Slideshow_2()`	`Slideshow`	Same but with adjusted relative path for subdirectories

`flyerRender()`	`flyer`	Loads service flyer image; shows/hides based on status flag
`HeroContent(page)`	`CTA`	Gets hero heading and caption text for a given page
`BibleVerse()`	`Scripture`	Gets the current daily scripture verse
`renderGallery()`	`picture`	Gets first 4 gallery images for homepage preview
`PresidentMessage()`	`PresidentWord`	Gets the president's message content
`renderVideoGallery()`	`song`	Gets YouTube video embed links
`BethelCommittee()`	`Committee`	Gets the most recent committee data
`CommitteeSlideshow()`	`Committee`	Gets all committees for the history page slideshow
`createYearDetails()`	`AnnualAchievement`	Gets yearly achievements for history timeline

PAGE RENDER FUNCTIONS

FUNCTION	CALLS
`renderHomePage()`	Slideshow, flyerRender, HeroContent, BibleVerse, renderGallery, PresidentMessage
`renderServicesPage()`	Slideshow_2, HeroContent, renderVideoGallery
`renderaboutPage()`	Slideshow_2, HeroContent, BethelCommittee
`renderGalleryPage()`	Slideshow_2, HeroContent
`renderHistoryPage()`	Slideshow_2, HeroContent, CommitteeSlideshow, createYearDetails

HELPER FUNCTION

```
highlightByPattern(originalText)

// Splits text by spaces, then wraps every other word in <span> tags
// Used for visual alternating-color effects in hero headings
// Example: "Glorifying God" → "Glorifying <span>God</span>"
```

api.js — Admin API Client

HTTP request handler used exclusively in the admin panel.

FUNCTION	METHOD	PURPOSE
`fetchData(resource, id)`	`GET`	Generic data retrieval
`postData(resource, payload, id)`	`POST`	Create or update records
`putData(resource, id, payload)`	`POST`	Update specific records
`deleteData(resource, id)`	`DELETE`	Remove records
`showMessageBox(title, message, isConfirm)`	—	Display modal dialog / confirmation

API BASE URL

`http://localhost:3000/admin/backend/main.php` — update this in `api.js` when deploying to production.

dashboard.js — Admin Panel

Manages all admin dashboard interactions including panel switching, form submissions, and data rendering.

PANEL SWITCHING

```
show_div2(panelId)  // Switches the visible admin panel
```

ADMIN PANELS

PANEL ID	DESCRIPTION
`intro`	Welcome dashboard with overview stats
`HomeCTA`	Edit hero section heading and caption per page
`scriptureOfDay`	Manage the daily scripture verse
`presidentMessage`	Edit president's message and photo
`servicesbody`	Upload and manage the service flyer
`bethel_songs`	Manage YouTube video song links
`picture`	Upload and manage gallery images
`memories`	Create memory albums
`updates`	Post news and updates
`B_commit`	Manage choir committee members and eras
`annual_achivements`	Add/edit yearly achievement records
`remarkable`	Manage special event highlights

04 CSS Styling

File Structure

FILE	PURPOSE
`css/index.css`	Entry point — imports `initial.css`
`css/initial.css`	Core styles (~1000 lines) — all public pages
`css/initial2.css`	Additional/supplementary styles

`admin/login.css`	Admin login page styles
`admin/css/admin.css`	Admin dashboard styles

Color Palette

Primary Black
#0F0F0F

Secondary Gold
#FFD700

Accent Olive
#8F9779

Text Light
#FFFFFF

Text Dark
#222222

Background Dark
#0F0F0F

```css
/* CSS Custom Properties — defined in :root */
--color-primary-black:    #0F0F0F;   /* Deep Charcoal */
--color-secondary-gold:   #FFD700;   /* Gold accent */
--color-accent-olive:     #8F9779;   /* Olive green */
--color-text-light:       #FFFFFF;
--color-text-dark:        #222222;
--color-background-dark:  #0F0F0F;
```

Typography

USE	FONT	TYPE
Headings	Playfair Display	Serif
Body Text	Inter	Sans-serif

CSS Sections in initial.css

#	SECTION	WHAT IT COVERS
1	Root Variables	Color palette and global CSS custom properties

2	Navbar	Fixed header, logo, links, mobile drawer
3	Hero Section	Full-screen background slider and overlay text
4	Content Sections	Cards, grids, content layout
5	Gallery	Image grid layout and lightbox overlay
6	Footer	Contact information, links, copyright

05 Pages & Sections

FILE	URL	PAGE NAME	JS LOADED
`index.html`	`/`	Home	main.js, home.js, renderApi.js
`websections/about.html`	`/websections/about.html`	About	main.js, about.js, renderApi.js
`websections/service.html`	`/websections/service.html`	Services	main.js, service.js, renderApi.js
`websections/gallery.html`	`/websections/gallery.html`	Gallery	main.js, gallery.js, renderApi.js
`websections/history.html`	`/websections/history.html`	History	main.js, history.js, renderApi.js
`admin/login.html`	`/admin/login.html`	Admin Login	login.js
`admin/dashboard.html`	`/admin/dashboard.html`	Admin Dashboard	dashboard.js, api.js

Home Page Sections

The homepage (`index.html`) is the main entry point and loads the following dynamic content sections via API:

SECTION	CONTENT SOURCE	NOTES
Hero Slider	Slideshow API	Images with auto-rotation every 7 seconds
Hero CTA Text	CTA API (page=Home)	Editable heading & caption
Daily Scripture	Scripture API	Bible verse displayed below hero
Gallery Preview	Picture API (first 4)	Links to full gallery page
President's Message	PresidentWord API	Photo + message text
Stats Counter	Static / animated JS	Members, songs, alumni counts
Service Flyer	Flyer API	Only shown if status = active

06 API Router — main.php

Located at `admin/backend/main.php`, this is the single entry point for all API requests. It reads the `action` query parameter and routes to the appropriate database operation.

Request Routing

ACTION	ROUTES TO
`login`	User::login()
`logout`	User::logout()
`CTA`	Manager::getCTA() / updateCTA()
`Scripture`	Manager::getScripture() / updateScripture()
`PresidentWord`	Manager::getWord() / updateWord()
`song`	Manager::getSong() / addSong() / updateSong() / deleteSong()
`picture`	Manager::getPicture() / addPicture() / updatePicture() / deletePicture()
`Slideshow`	Manager::getSlideshowImages() / updateSlideshowImage()
`Committee`	Manager::getCommittee() / addCommittee() / updateCommittee() / deleteCommittee()
`AnnualAchievement`	Manager::getAnnualAchievements() / add / update / delete
`flyer`	Manager::getFlyer() / updateFlyer()

Code Structure

```php
// 1. Start session
session_start();

// 2. CORS headers
header('Access-Control-Allow-Origin: *');
header('Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS');

// 3. File upload handler
function uploadImage($files) { /* validates & moves file to media/ */ }

// 4. Database connection
$conn = new mysqli($servername, $username, $password, $dbname);

// 5. Action router
switch ($action) {
    case 'login':  /* ... */
    case 'CTA':    /* ... */
    // etc.
}

// 6. JSON output
echo json_encode($response);
```

Key Features

The router handles **CORS preflight** (OPTIONS method), **file uploads** with unique filename generation via `uniqid()`, and wraps all operations in try-catch blocks that return structured JSON error responses.

07 Database Operations — manager.php

Located at `admin/backend/manager.php`. Contains two classes: **User** and **Manager**.

User Class

METHOD	PURPOSE
`register(email, passkey)`	Create a new admin account
`login(email, passkey)`	Authenticate user and set PHP session
`logout()`	Destroy the session
`changePassword(userId, current, new)`	Update password with verification
`changeEmail(userId, password, newEmail)`	Update email after password confirmation

Manager Class

CATEGORY	METHOD	SQL
CTA	`getCTA(page)`	SELECT from cta
	`updateCTA(data)`	UPDATE cta
Scripture	`getScripture(id)`	SELECT from scriptureofday
	`updateScripture(data)`	UPDATE scriptureofday
President	`getWord(id)`	SELECT from presidentsword
	`updateWord(data)`	UPDATE presidentsword
Songs	`getSong / addSong / updateSong / deleteSong`	Full CRUD on songs table
Pictures	`getPicture / addPicture / updatePicture / deletePicture`	Full CRUD on pictures table
	`updateSlideshowImage(id, status)`	UPDATE pictures SET slideshow
Committee	`getAllCommittees()`	JOIN bethelcommitte + bethelcommittemember
	`addCommittee / updateCommittee / deleteCommittee`	Transactional CRUD
Achievements	`getAnnualAchievements / add / update / delete`	Full CRUD on annualachievements
Flyer	`getFlyer / updateFlyer`	SELECT/UPDATE flyer table

Slideshow	`getSlideshowImages()`	SELECT pictures WHERE slideshow=1

08 API Reference

All requests go to `admin/backend/main.php` with a `?action=` query parameter.

Authentication

POST `?ACTION=LOGIN`

```
// Request body
email=admin@example.com&password=password123

// Response
{
  "success": true,
  "message": "Login successful",
  "data": { "user_id": 1, "email": "admin@example.com" }
}
```

POST `?ACTION=LOGOUT`

```
{
  "success": true,
  "message": "Logged out successfully"
}
```

Content Endpoints

GET `?ACTION=CTA&ID=HOME`

```
{
  "success": true,
  "data": {
    "id": 1,
    "page": "Home",
    "heading": "Welcome to Bethel Family Choir",
    "caption": "Glorifying God through music since 2005"
  }
}
```

GET `?ACTION=PICTURE`

```
{
  "success": true,
  "data": [
    {
      "id": 1,
      "caption": "Choir performance",
      "link": "media/abc123image.jpg",
      "slideshow": 1,
      "created_at": "2024-01-15 10:30:00"
    }
  ]
```

```
}
```

`GET` `?ACTION=COMMITTEE`

```
{
  "success": true,
  "data": [{
    "commit_id": 1,
    "era": "2024-2025",
    "ecree": "Bethel niferme",
    "picture": "media/committee2024.jpg",
    "members": [
      { "names": "John Doe", "post": "President" },
      { "names": "Jane Smith", "post": "Vice President" }
    ]
  }]
}
```

All endpoints follow the same pattern: `{ "success": true/false, "data": [...], "message": "..." }`. Failed operations return `"success": false` with an error message.

09 Database Schema

Database name: `bethelfinal`

Tables Overview

TABLE	DESCRIPTION	KEY COLUMNS
`users`	Admin accounts	id, email, passkey
`cta`	Hero section text per page	page, heading, caption
`scriptureofday`	Daily Bible verse	title, content
`presidentsword`	President's message	name, message, image
`pictures`	Gallery images	link, caption, slideshow
`songs`	YouTube video links	title, link
`bethelcommitte`	Committee eras	era, ecree, picture
`bethelcommittemember`	Members per committee	commit_id (FK), names, post
`annualachievements`	Yearly achievements	year, summary
`flyer`	Service flyer	link, status

Key SQL Definitions

```
CREATE TABLE pictures (
    id          INT PRIMARY KEY AUTO_INCREMENT,
    caption     VARCHAR(255),
```

```
    link        VARCHAR(255),
    slideshow   TINYINT DEFAULT 0,   -- 1 = appears in hero slider
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE bethelcommittemember (
    id         INT PRIMARY KEY AUTO_INCREMENT,
    commit_id INT,
    names      VARCHAR(255),
    post       VARCHAR(255),
    FOREIGN KEY (commit_id) REFERENCES bethelcommitte(commit_id) ON DELETE CASCADE
);
```

SLIDESHOW FLAG

The `pictures.slideshow` column controls whether an image appears in the hero slider (`1`) or only in the gallery (`0`). This can be toggled from the admin panel.

10 Security

CONCERN	IMPLEMENTATION
Authentication	PHP `$_SESSION` — set on login, destroyed on logout
Password Storage	`password_hash()` bcrypt + `password_verify()`
SQL Injection	Prepared statements with `$stmt->bind_param()` throughout
File Uploads	Unique filenames via `uniqid()`, type validation, moved to `media/`
Error Handling	try-catch blocks; generic messages to client, detailed `error_log()`
CORS	Currently set to `*` — restrict to your domain in production

PRODUCTION WARNING

Change `Access-Control-Allow-Origin: *` to your specific domain before going live. Example: `Access-Control-Allow-Origin: https://bethelfamilychoir.com`

11 Deployment Guide

Prerequisites

PHP 8.0+, MySQL 5.7+, Apache or Nginx web server.

Step-by-Step Setup

1. CREATE THE DATABASE

```
CREATE DATABASE bethelfinal;
USE bethelfinal;
-- Run all CREATE TABLE statements from Section 9
```

2. CONFIGURE DATABASE CONNECTION

Edit `admin/backend/main.php`:

```
$servername = "localhost";
$username   = "root";
$password   = "";          // Your MySQL password
$dbname     = "bethelfinal";
```

3. SET FILE PERMISSIONS

```
chmod 755 media/
chown www-data:www-data media/
```

4. START PHP SERVER (DEVELOPMENT)

```
cd admin/backend
php -S localhost:3000
```

5. ACCESS THE SITE

WHAT	URL
Public Website	`http://localhost/`
Admin Login	`http://localhost/admin/login.html`
Admin Dashboard	`http://localhost/admin/dashboard.html`
API Endpoint	`http://localhost:3000/admin/backend/main.php`

12 Content Editor Guide

This section is for the **website content editor** — the person responsible for keeping the site up to date. All content changes are made through the Admin Dashboard.

> HOW TO ACCESS
>
> Go to `/admin/login.html` and enter your admin email and password. You will be redirected to the dashboard.

What You Can Edit

CONTENT	DASHBOARD PANEL	HOW TO UPDATE
Hero text (heading & caption)	Home CTA	Select the page, edit text, click Save

Daily Bible verse	Scripture of the Day	Type the verse title and content, click Save
President's message	President's Message	Edit the name, message text, and upload a photo
Gallery photos	Picture Management	Upload images; toggle "Slideshow" to add to hero slider
Songs / Videos	Bethel Songs	Paste a YouTube embed URL and give it a title
Service flyer	Services Body	Upload flyer image and toggle it on/off
Committee members	B-Commit	Select or create an era, add member names and roles
Annual achievements	Annual Achievements	Add year and a summary of achievements

Adding Gallery Images

Navigate to **Picture Management** in the dashboard. Click "Add New", choose an image file from your computer, and add a caption. Once uploaded, you can toggle the **Slideshow** switch to make the image appear in the homepage hero slider.

Managing the Committee

Go to **B-Commit** in the dashboard. You can create a new committee era (e.g., "2024–2025"), upload a group photo, and add each member with their name and role/position. The most recent committee automatically appears on the About page.

> TIP FOR EDITORS
>
> Always check the public website after making changes to confirm everything looks correct. Use a different browser or incognito mode to see the site as a visitor would.

13 Troubleshooting

ISSUE	SOLUTION
500 Internal Server Error	Check PHP error logs; verify database credentials in main.php
Images not uploading	Check `media/` folder permissions; verify `php.ini` upload_max_filesize
API calls failing	Ensure PHP server is running on port 3000; verify API_BASE_URL in api.js
Database connection failed	Check MySQL credentials and that the service is running
CORS errors in browser	Update allowed origin in main.php to match your domain
Login not working	Ensure `session_start()` is at top of main.php; check password hash
Content not updating on site	Hard refresh browser (Ctrl+Shift+R); check browser console for API errors

Enable Debug Mode

```
// Add to the very top of admin/backend/main.php
ini_set('display_errors', 1);
error_reporting(E_ALL);
// ⚠ Remove this in production!
```

Performance Tips

Enable PHP OPcache on the server for faster script execution. Add database indexes on frequently queried columns like `pictures.slideshow` and `cta.page`. Compress images before uploading to keep page load times fast. For static assets (CSS, JS), consider using a CDN.

> SUPPORT
>
> For technical support, contact the development team at **Tris Tech Hub**.