MELODAI PROJECT DOCUMENTATION

MelodAI Installation Guide

Complete setup for the MelodAI web platform (Laravel 13 + Inertia) and the Expo 56 React Native mobile app — local development, shared hosting, VPS, API providers, and production mobile builds.

Generated 2026-06-04 · melodai-project documentation

Table of contents

PART 1

MelodAI Installation Guide

Laravel 13 + Inertia web platform and Expo 56 React Native mobile app. Local development, cPanel shared hosting, VPS, API keys, PHP extensions, queues, and production APK builds.

Overview

MelodAI is an AI music and media SaaS platform built with , , and a companion mobile app (`melodai-app`) using and React Native.

The web app includes an installation wizard at `/install` , a shared login at `/login` (admins and users), and an admin panel at `/admin` for users with the admin role. It supports HTTP cron endpoints, database queue workers, and integrations for Suno, OpenAI, xAI, FAL, Stripe, Flutterwave, Revolut, Cryptomus, Cloudflare R2, YouTube OAuth, and more.

Technology stack

- PHP 8.3+, Laravel 13, MySQL/MariaDB, database queues (Horizon optional on Linux VPS)
- Inertia.js, Vite, Tailwind CSS v4
- Expo 56, React Native 0.85, Expo Router, TypeScript
- FFmpeg (lyrics video and audio processing)
- Local disk or Cloudflare R2 (recommended for production)

Documentation guides

Local development

XAMPP/WAMP/Laragon, php.ini extensions, Composer, Node, migrations, queue worker, installer wizard.

Shared hosting (cPanel)

Hostinger/cPanel deploy, document root, HTTP cron URLs, PHP queue crons, SSL.

VPS deployment

Ubuntu, Nginx, PHP-FPM 8.3, Supervisor, Reverb, FFmpeg, production hardening.

API keys & providers

Step-by-step registration for every key in `.env.example`.

Expo 56 mobile app

Windows, macOS, Linux dev setup, Expo Go, production API URL, EAS builds, APK/AAB release.

Quick start (local)

1. Install PHP 8.3+, Composer 2, Node.js 20 LTS, MySQL, and FFmpeg.
2. Clone the project, copy `.env.example` to `.env`, enable required PHP extensions.
3. Run `composer install`, `npm install`, `npm run build`, and migrations.
4. Start dev stack: `composer run dev` (app, queue worker, and Vite).
5. Complete the `/install` wizard.
6. Mobile: set `EXPO_PUBLIC_API_URL` in `melodai-app/.env` and run `npx expo start`.

Pre-launch checklist

Application

- [ ] `APP_ENV=production`, `APP_DEBUG=false`, `APP_URL=https://yourdomain.com`
- [ ] Admin → Settings → General → matches production HTTPS (used for cron URLs)
- [ ] `APP_KEY` is set; never commit `.env`
- [ ] Suno, R2 storage, and payment keys configured and tested

Operations

- [ ] HTTP cron jobs copied from Admin → Cron & Queue
- [ ] Queue workers running (Supervisor on VPS or cPanel PHP cron)
- [ ] `php artisan storage:link`; writable `storage` and `bootstrap/cache`
- [ ] Mobile app built with production `EXPO_PUBLIC_API_URL`

Post-launch verification

- [ ] Homepage, login, registration, and admin panel load over HTTPS
- [ ] Music generation end-to-end (queue processes job, user notified)
- [ ] Stripe, Flutterwave, Revolut (sandbox), and/or Cryptomus test payment, then one live payment per enabled provider
- [ ] Mobile app login against production API; YouTube connect if enabled
- [ ] Review `storage/logs/laravel.log` and Admin → Queue Monitor

PART 2

Local Development (Windows / macOS / Linux)

Install MelodAI on XAMPP, WAMP, Laragon, or native PHP. Includes php.ini extensions, database setup, installer wizard, queues, and FFmpeg.

Requirements

- (CLI and web server must use the same version)
-
- and npm
- or MariaDB 10.6+
- (required for lyrics video and audio pipelines)
- Optional: Redis (not required; MelodAI uses database queues by default)

On Windows, XAMPP is the most common stack. Laragon and WAMP work the same way with different folder paths.

Step 1: Enable Required PHP Extensions (php.ini)

Edit the `php.ini` used by both Apache and CLI. Typical paths:

- Windows XAMPP: `C:\xampp\php\php.ini`
- Windows WAMP: `C:\wamp64\bin\php\php8.3.x\php.ini`
- macOS Homebrew: `/opt/homebrew/etc/php/8.3/php.ini`
- Linux: `/etc/php/8.3/apache2/php.ini` and `/etc/php/8.3/cli/php.ini`

(remove leading `;` if commented):

- `bcmath`, `ctype`, `curl`, `dom`, `fileinfo`, `gd`, `intl`, `mbstring`, `mysql`, `openssl`, `pcntl`, `pdo_mysql`, `posix`, `tokenizer`, `xml`, `zip`

- `exif`, `iconv`, `sodium`
- `redis` only if you switch `QUEUE_CONNECTION` / `CACHE_STORE` to Redis

Also verify these php.ini values for uploads and long jobs:

```
upload_max_filesize = 64M
```

```
post_max_size = 64M
```

```
max_execution_time = 300
```

```
memory_limit = 512M
```

Windows/XAMPP SSL fix (if OpenAI, Suno, or R2 return cURL error 60):

```
curl.cainfo = C:\xampp\php\extras\ssl\cacert.pem
```

```
openssl.cafile = C:\xampp\php\extras\ssl\cacert.pem
```

Restart Apache, then verify extensions:

```
php -m
```

```
php --ini
```

Step 2: Prepare Project and .env

Place the project in your web root, for example `C:\xampp\htdocs\melodai`.

```
cd C:\xampp\htdocs\melodai
```

```
copy .env.example .env
```

```
composer install
```

```
php artisan key:generate
```

Minimum local `.env` values:

```
APP_ENV=local
```

```
APP_DEBUG=true
```

```
APP_URL=http://localhost/melodai/public
```

```
SKIP_INSTALL_WIZARD=false
```

```
QUEUE_CONNECTION=database
```

```
MELODAI_QUEUE_AFTER_RESPONSE=true
```

```
REDIS_CLIENT=predis
```

On XAMPP, use `OPENAI_SSL_VERIFY=false` and `CLOUDFLARE_R2_SSL_VERIFY=false` only if CA bundle is missing. Prefer fixing `curl.cainfo` instead.

Step 3: Database Setup

Create a MySQL database (phpMyAdmin or CLI), then set:

```
DB_CONNECTION=mysql
```

```
DB_HOST=127.0.0.1
```

```
DB_PORT=3306
```

```
DB_DATABASE=melodai
```

```
DB_USERNAME=root
```

```
DB_PASSWORD=
```

Run migrations and seeders:

```
php artisan migrate
```

```
php artisan db:seed
```

```
php artisan storage:link
```

Step 4: Build Frontend Assets

```
npm install
```

```
npm run build
```

For active development with hot reload, use `npm run dev` instead of build-only.

Step 5: Run the Application

Recommended — starts web server, queue worker, and Vite together:

```
composer run dev
```

Alternative manual terminals:

```
php artisan serve
```

```
php artisan melodai:work --tries=1
```

```
npm run dev
```

- Website (artisan serve): `http://127.0.0.1:8000`
- XAMPP Apache: `http://localhost/melodai/public`
- Login (users and admins): `/login` — admin accounts are redirected to `/admin` after sign-in

Horizon requires Linux/macOS with `pcntl`. On Windows/XAMPP always use `php artisan melodai:work` or `composer run dev`.

Step 6: Installation Wizard

On first visit, open `/install` and complete all steps (database, site URL, admin account, and integrations).

After installation, configure providers in (AI, Mail, Payment, YouTube, Security).

Step 7: FFmpeg (Local)

Install FFmpeg and set paths in `.env`:

```
FFMPEG_BINARIES=C:\ffmpeg\bin\ffmpeg.exe
```

```
FFPROBE_BINARIES=C:\ffmpeg\bin\ffprobe.exe
```

Linux/macOS example:

```
FFMPEG_BINARIES=/usr/bin/ffmpeg
```

Troubleshooting (Local)

- run `npm run build` or `npm run dev`.
- keep `melodai:work` or `composer run dev` running.
- set CA bundle in php.ini or temporarily `OPENAI_SSL_VERIFY=false`.
- ensure `storage` and `bootstrap/cache` are writable.
- confirm DB connection works and migrations completed.

PART 3

Shared Hosting Deployment (cPanel / Hostinger)

Deploy MelodAI on cPanel: PHP 8.3 extensions, document root, Hostinger package, HTTP cron, queue workers, SSL, and production .env.

Hosting Requirements

- PHP (8.2 minimum if host does not offer 8.3 yet)
- MySQL 8 / MariaDB, SSH or Terminal access (strongly recommended)
- Composer available via SSH or upload `vendor` from local build
- Ability to set document root to Laravel `public` folder
- Cron jobs (required for scheduler and queue processing)

Step 1: PHP Version and Extensions (cPanel)

In cPanel open or :

- Select PHP for your domain.
- Enable: `bcmath`, `ctype`, `curl`, `fileinfo`, `gd`, `intl`, `mbstring`, `mysql`, `pcntl`, `pdo_mysql`, `posix`, `openssl`, `tokenizer`, `xml`, `zip`, `exif`.
- Set `memory_limit` ≥ 256M, `max_execution_time` ≥ 300.

Verify from Terminal:

```
/usr/local/bin/php -m
```

Step 2: Build and Upload Hostinger Package

On your local machine (with Composer and Node installed), create the hosting archive:

```
composer run package-hostinger
```

Upload `melodai-hostinger.zip` via cPanel File Manager. Do zip `vendor` or `node_modules` manually — large extracts often cause HTTP 500 on shared hosts.

1. Extract zip into `public_html` (or a folder above it).
2. Run `composer install --no-dev --optimize-autoloader` on the server after extract.

Step 3: Point Document Root to /public

1. cPanel → → edit your domain.
2. Set document root to `/home/USERNAME/public_html/public` (adjust path to match extract location).
3. If the host cannot change docroot, copy only `public/*` into `public_html` and update `index.php` paths to the Laravel root one level up.

Step 4: Configure Production .env

```
APP_ENV=production
```

```
APP_DEBUG=false
```

```
APP_URL=https://melodai.site
```

```
QUEUE_CONNECTION=database
```

```
FILESYSTEM_DISK=cloudflare
```

Set database credentials from cPanel MySQL Databases:

```
DB_DATABASE=cpuser_melodai
```

```
DB_USERNAME=cpuser_dbuser
```

```
DB_PASSWORD=strong_password
```

In , set to your HTTPS domain. Cron HTTP URLs are generated from this value.

Step 5: Run Artisan Commands

From cPanel Terminal (use full PHP path):

```
/usr/local/bin/php /home/USERNAME/public_html/artisan key:generate --force
```

```
/usr/local/bin/php /home/USERNAME/public_html/artisan migrate --force
```

```
/usr/local/bin/php /home/USERNAME/public_html/artisan storage:link
```

```
/usr/local/bin/php /home/USERNAME/public_html/artisan config:cache
```

```
/usr/local/bin/php /home/USERNAME/public_html/artisan route:cache
```

```
/usr/local/bin/php /home/USERNAME/public_html/artisan view:cache
```

Complete `/install` if this is a fresh server, or ensure installer was completed once.

Step 6: Cron Jobs and Queue Workers

Open on your live site and copy the exact commands shown (they include your token and server paths).

```
curl -s "https://yourdomain.com/cron/schedule?token=YOUR_TOKEN" > /dev/null 2>&1
```

```
/usr/local/bin/php /home/USERNAME/public_html/artisan queue:work database --queue=default --sleep=3 --tries=3 --timeout=60 --stop-when-empty > /dev/null 2>&1
```

Never use `melodai.site/public_html/artisan` as the command — that is a hostname fragment, not a file path. Use `/usr/local/bin/php` + absolute path to `artisan`.

Suggested schedules:

- Scheduler URL: every minute (`* * * * *`)
- Default queue: every minute
- Music generation queue: every 2 minutes (`*/2 * * * *`)
- Video processing queue: every 2 minutes

Step 7: SSL and SMTP

Enable AutoSSL in cPanel, then force HTTPS in `APP_URL` and Site URL.

Configure mail in or `.env`:

```
MAIL_MAILER=smtp
```

```
MAIL_HOST=mail.yourdomain.com
```

```
MAIL_PORT=465
```

```
MAIL_USERNAME=info@yourdomain.com
```

```
MAIL_PASSWORD=mailbox_password
```

Troubleshooting (Shared Hosting)

- copy fresh URLs from Admin → Cron & Queue; ensure identical token on all jobs.
- run `composer install --no-dev` on server; check `storage/logs/laravel.log`.
- verify queue cron commands use full PHP + artisan paths.
- ask host to enable FFmpeg or use VPS for video features.

PART 4

VPS Deployment (Ubuntu + Nginx + PHP 8.3)

Production VPS setup: PHP-FPM extensions, Nginx, SSL, Supervisor queue workers, Horizon, optional Reverb, FFmpeg, and deployment commands.

Step 1: Provision and Secure VPS

- Ubuntu 22.04 or 24.04 LTS
- Firewall: allow SSH (22), HTTP (80), HTTPS (443)
- Deploy user with sudo (non-root for daily tasks)

```
sudo ufw allow OpenSSH && sudo ufw allow 'Nginx Full' && sudo ufw enable
```

PHP 8.3 Extensions (VPS php.ini)

Install PHP-FPM and extensions:

```
sudo apt install -y php8.3-fpm php8.3-cli php8.3-mysql php8.3-curl php8.3-xml ph
p8.3-mbstring php8.3-zip php8.3-gd php8.3-bcmath php8.3-intl php8.3-fileinfo php
8.3-redis php8.3-pcntl
```

Edit `/etc/php/8.3/fpm/php.ini` and `/etc/php/8.3/cli/php.ini`:

- `memory_limit = 512M`
- `upload_max_filesize = 64M`
- `max_execution_time = 300`
- Ensure `opcache` is enabled for production

```
sudo systemctl restart php8.3-fpm
```

Step 2: Install Nginx, MySQL, Node, Composer, FFmpeg

```
sudo apt update && sudo apt install -y nginx mysql-server redis-server supervisor git unzip curl ffmpeg
```

```
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash - && sudo apt install -y nodejs
```

```
curl -sS https://getcomposer.org/installer | php && sudo mv composer.phar /usr/local/bin/composer
```

Create database:

```
CREATE DATABASE melodai_live CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

```
CREATE USER 'melodai'@'localhost' IDENTIFIED BY 'strong_password';
```

```
GRANT ALL PRIVILEGES ON melodai_live.* TO 'melodai'@'localhost'; FLUSH PRIVILEGES;
```

Step 3: Deploy MelodAI

```
cd /var/www && sudo git clone <repo-url> melodai
```

```
cd /var/www/melodai && cp .env.example .env
```

```
composer install --no-dev --optimize-autoloader
```

```
npm ci && npm run build
```

```
php artisan key:generate --force
```

```
php artisan migrate --force
```

```
php artisan storage:link
```

```
php artisan config:cache && php artisan route:cache && php artisan view:cache
```

```
sudo chown -R www-data:www-data /var/www/melodai/storage /var/www/melodai/bootstrap/cache
```

Production `.env` highlights:

```
APP_ENV=production
```

```
APP_DEBUG=false
```

```
QUEUE_CONNECTION=database
```

```
REDIS_CLIENT=phpredis
```

```
FFMPEG_BINARIES=/usr/bin/ffmpeg
```

```
FFPROBE_BINARIES=/usr/bin/ffprobe
```

Step 4: Nginx Virtual Host and SSL

Point domain A record to VPS IP. Nginx root must be `/var/www/melodai/public`.

```
sudo certbot --nginx -d yourdomain.com -d www.yourdomain.com
```

Step 5: Supervisor Queue Workers

On Linux VPS you can use Horizon (requires Redis) or database workers via Supervisor.

(`/etc/supervisor/conf.d/melodai-worker.conf`):

```
[program:melodai-default]
command=php /var/www/melodai/artisan queue:work database --queue=default --sleep
=3 --tries=3 --timeout=900
autostart=true
autorestart=true
user=www-data
numprocs=1
```

Add separate programs for `music-generation` and `video-processing,video-clips` queues.

```
sudo supervisorctl reread && sudo supervisorctl update
```

```
php artisan queue:restart
```

After each deploy, run `php artisan queue:restart` so workers reload code.

Step 6: Laravel Reverb (Optional Real-Time)

Reverb provides WebSocket broadcasting. It works on self-hosted VPS (not typical shared hosting). Generate keys:

```
php artisan tinker --execute "echo 'KEY=' . bin2hex(random_bytes(16)) . PHP_EOL
. 'SECRET=' . bin2hex(random_bytes(32));"
```

Set in `.env`:

```
BROADCAST_CONNECTION=reverb
```

```
REVERB_APP_ID=melodai
```

```
REVERB_HOST=your-vps-ip-or-ws.yourdomain.com
```

```
REVERB_PORT=8080
```

```
REVERB_SCHEME=https
```

Run Reverb via Supervisor and open firewall port if needed. Configure Nginx reverse proxy for WSS in production.

Step 7: Laravel Scheduler

```
* * * * * cd /var/www/melodai && php artisan schedule:run >> /dev/null 2>&1
```

Alternatively use HTTP cron URLs from Admin → Cron & Queue if you prefer curl-based scheduling.

Troubleshooting (VPS)

- 502 Bad Gateway: check `php8.3-fpm` status and Nginx `fastcgi_pass` socket path.
- Permission denied: fix ownership to `www-data`.
- Video processing fails: verify `ffmpeg -version` and `FFMPEG_BINARIES` path.
- Queue not working: ensure Redis and `pcntl` extension are installed on Linux.

PART 5

API Provider Registration & Environment Keys

How to register each platform, retrieve credentials, and map them to MelodAI .env variables and Admin settings.

Standard Setup Workflow

1. Create provider account and complete verification/billing where required.
2. Generate API keys, OAuth client IDs, or webhook secrets in the provider dashboard.
3. Add values to production `.env` and local `.env` (never commit secrets).
4. Mirror credentials in when the panel supports runtime overrides (Mail, Payment, AI, YouTube).
5. Test in sandbox/test mode, then switch to live keys.
6. Run `php artisan optimize:clear` then `php artisan config:cache` on the server.

OpenAI

Portal: platform.openai.com/api-keys

1. Sign up and add a payment method under Billing.
2. Create a new secret API key (copy immediately — shown once).
3. Optional: set organization ID if you use multiple orgs.

```
OPENAI_API_KEY=sk-proj-...
```

```
OPENAI_ORGANIZATION=
```

```
OPENAI_IMAGE_MODEL=gpt-image-1
```

```
OPENAI_SSL_VERIFY=true
```

Used for images, text, and related AI features configured in Admin → Settings → AI.

xAI (Grok)

Portal: console.x.ai

1. Create account and open API keys section.
2. Generate API key and copy to `XAI_API_KEY`.

```
XAI_API_KEY=xai-...
```

```
XAI_BASE_URL=https://api.x.ai/v1
```

```
XAI_IMAGE_MODEL=grok-imagine-image-quality
```

```
XAI_VIDEO_MODEL=grok-imagine-video
```

FAL.ai & Suno API

FAL.ai (video models: Kling, Veo, etc.)

Portal: fal.ai/dashboard/keys

```
FAL_KEY=...
```

Suno (music generation)

1. Register at your Suno API provider (e.g. sunoapi.org) and create an API key.
2. — Suno does not issue a webhook signing secret. You create a long random string and use it on both sides below.
3. Set `SUNO_WEBHOOK_SECRET` in `.env` to that value (required in production).
4. In the Suno provider dashboard, set the callback/webhook URL to the same secret as the `token` query parameter:

```
https://yourdomain.com/webhooks/suno?token=YOUR_SUNO_WEBHOOK_SECRET
```

Generate a secret locally (example):

```
php -r "echo bin2hex(random_bytes(32));"
```

```
SUNO_API_BASE_URL=https://api.sunoapi.org
```

```
SUNO_API_KEY=...
```

```
SUNO_WEBHOOK_SECRET=paste_the_same_random_string_here
```

`SUNO_WEBHOOK_SECRET` is required in production. MelodAI appends it to Suno callback URLs as `?token=...` so only requests with your secret are accepted — this is your secret, not one from Suno.

YouTube OAuth (Publishing)

Portal: Google Cloud Console → Credentials

1. Create a project (or select existing).
2. Enable .
3. Configure OAuth consent screen (External, add test users during development).
4. Create → type .
5. Add authorized redirect URI exactly: `https://yourdomain.com/dashboard/youtube/callback`
6. Copy Client ID and Client Secret.

```
YOUTUBE_CLIENT_ID=....apps.googleusercontent.com
```

```
YOUTUBE_CLIENT_SECRET=GOCSPX-...
```

```
YOUTUBE_REDIRECT_URI=https://yourdomain.com/dashboard/youtube/callback
```

For Expo dev against a LAN API, set `YOUTUBE_REDIRECT_URI` to your production HTTPS callback while the mobile app uses the server OAuth flow. Also configure in Admin → Settings → YouTube.

Cloudflare R2 Storage

Portal: Cloudflare Dashboard → R2

1. Create an R2 bucket (e.g. `melodai-storage`).
2. Create R2 API token with Object Read & Write on that bucket.
3. Copy Access Key ID, Secret Access Key, and account endpoint URL.
4. Optionally connect a custom domain for public CDN URL.

```
FILESYSTEM_DISK=cloudflare
```

```
CLOUDFLARE_R2_KEY=...
```

```
CLOUDFLARE_R2_SECRET=...
```

```
CLOUDFLARE_R2_REGION=auto
```

```
CLOUDFLARE_R2_BUCKET=melodai-storage
```

```
CLOUDFLARE_R2_ENDPOINT=https://<account-id>.r2.cloudflarestorage.com
```

```
CLOUDFLARE_R2_URL=https://cdn.yourdomain.com
```

Stripe (Laravel Cashier)

Portal: dashboard.stripe.com/apikeys

1. Create Stripe account and complete business verification.
2. Copy Publishable key and Secret key (test mode first).
3. Developers → Webhooks → Add endpoint: `https://yourdomain.com/stripe/webhook` (Cashier default).
4. Select events Cashier needs (checkout, customer subscription, invoice).
5. Copy webhook signing secret.

```
STRIPE_KEY=pk_live_...
```

```
STRIPE_SECRET=sk_live_...
```

```
STRIPE_WEBHOOK_SECRET=whsec_...
```

```
STRIPE_CURRENCY=usd
```

Also enable Stripe in Admin → Settings → Payment.

Flutterwave

Portal: dashboard.flutterwave.com

1. Complete merchant verification.
2. Copy Public Key, Secret Key, and Encryption Key.
3. Set webhook URL and copy webhook hash.

```
FLUTTERWAVE_PUBLIC_KEY=FLWPUBK-...
```

```
FLUTTERWAVE_SECRET_KEY=FLWSECK-...
```

```
FLUTTERWAVE_ENCRYPTION_KEY=...
```

```
FLUTTERWAVE_WEBHOOK_HASH=...
```

Enable Flutterwave in .

Revolut Merchant API

Portal: Revolut Merchant API (Business account → Merchant API). MelodAI uses API version `2026-04-20` for orders, webhooks, and refunds.

1. Create or open a Revolut Business merchant account and complete onboarding.
2. Generate API keys (public + secret). Start with .
3. Create a webhook endpoint on Revolut pointing to your MelodAI URL (HTTPS in production).
4. Copy the webhook (`wsk_...`).
5. Add keys to `.env` or (admin values override .env when saved).
6. Turn on and save. Revolut appears at checkout only when enabled and a secret key is configured.

Webhook & return URLs

```
https://yourdomain.com/webhooks/revolut
```

```
https://yourdomain.com/dashboard/billing/payment/revolut/return
```

The return URL runs after hosted checkout. Fulfillment also runs from webhooks when the order completes.

Environment variables

```
REVOLUT_PUBLIC_KEY=pk_...
```

```
REVOLUT_SECRET_KEY=sk_...
```

```
REVOLUT_WEBHOOK_SIGNING_SECRET=wsk_...
```

```
REVOLUT_CURRENCY=USD
```

```
REVOLUT_ENVIRONMENT=sandbox
```

Set `REVOLUT_ENVIRONMENT=production` (or in admin) for live payments. Sandbox: `https://sandbox-merchant.revolut.com`; production: `https://merchant.revolut.com`.

Subscriptions and credit packs use one-shot Revolut orders, not Revolut Subscription Plans. Refunds: Admin → Billing → Invoices when the invoice allows refunds.

Cryptomus (crypto payments)

Portal: cryptomus.com → Business → Merchant → API integration.

1. Register a Cryptomus merchant account and complete verification.
2. Copy and .
3. Set the payment webhook URL in Cryptomus to your MelodAI endpoint (HTTPS).
4. Add credentials to `.env` or .
5. Enable . Checkout shows Cryptomus when enabled and both UUID and payment key are set.

Webhook & return URLs

```
https://yourdomain.com/webhooks/cryptomus
```

```
https://yourdomain.com/dashboard/billing/payment/cryptomus/return
```

Cryptomus signs webhooks with an MD5 signature from your payment key; MelodAI verifies before fulfilling invoices or subscriptions.

Environment variables

```
CRYPTOMUS_MERCHANT_UUID=...
```

```
CRYPTOMUS_PAYMENT_KEY=...
```

```
CRYPTOMUS_CURRENCY=USD
```

```
CRYPTOMUS_INVOICE_LIFETIME=7200
```

require a destination crypto wallet address. In , enter the refund wallet when refunding Cryptomus payments.

Payment providers — activation checklist

MelodAI supports Stripe (Cashier), Flutterwave, Revolut, and Cryptomus. Configure in
or `.env`; admin-saved values override .env when present.

1. Set `APP_URL` and to your public HTTPS domain.
2. Register webhook URLs at each provider (see sections above).
3. Enable each provider toggle and save required keys.
4. Run sandbox/test checkout for subscriptions and credit packs from .
5. Confirm invoice creation and entitlements (credits or active subscription).
6. For production, switch Revolut to production, Stripe to live keys, and Cryptomus/Flutterwave to live credentials.

Admin refunds

- payment intent refund when allowed.
- transaction refund API.
- order refund API.
- refund with admin-supplied wallet address; entitlements reversed on success.

Social Login (Google, GitHub, Discord)

Google

1. Google Cloud Console → OAuth client (Web).
2. Redirect URI: `https://yourdomain.com/auth/google/callback`

```
GOOGLE_CLIENT_ID=...
```

```
GOOGLE_CLIENT_SECRET=...
```

```
GOOGLE_REDIRECT_URI=${APP_URL}/auth/google/callback
```

GitHub

github.com/settings/developers → OAuth App → Authorization callback URL
`https://yourdomain.com/auth/github/callback`

```
GITHUB_CLIENT_ID=...
```

```
GITHUB_CLIENT_SECRET=...
```

Discord

discord.com/developers/applications → OAuth2 → Redirect
`https://yourdomain.com/auth/discord/callback`

```
DISCORD_CLIENT_ID=...
```

```
DISCORD_CLIENT_SECRET=...
```

Leave blank to hide social login buttons (`FEATURE_SOCIAL_LOGIN=true` still requires keys).

Cloudflare Turnstile (Captcha)

Portal: Cloudflare Dashboard → Turnstile → Add site widget for your domain.

```
TURNSTILE_SITE_KEY=0x4AAAA...
```

```
TURNSTILE_SECRET_KEY=0x4AAAA...
```

Prefer Admin → Settings → Security for runtime updates.

SMTP Email

Use your hosting mailbox or transactional provider (Brevo, Mailgun, Amazon SES).

```
MAIL_MAILER=smtp
```

```
MAIL_HOST=mail.yourdomain.com
```

```
MAIL_PORT=587
```

```
MAIL_USERNAME=info@yourdomain.com
```

```
MAIL_PASSWORD=...
```

```
MAIL_FROM_ADDRESS=info@yourdomain.com
```

Test from Admin → Settings → Mail → Send test email.

Pusher & Reverb (Real-Time)

Pusher (managed)

dashboard.pusher.com → Create app → copy app id, key, secret, cluster.

```
BROADCAST_CONNECTION=pusher
```

```
PUSHER_APP_ID=...
```

```
PUSHER_APP_KEY=...
```

```
PUSHER_APP_SECRET=...
```

```
PUSHER_APP_CLUSTER=mt1
```

Reverb (self-hosted VPS)

See VPS documentation for generating `REVERB_APP_KEY` and `REVERB_APP_SECRET`.

Full .env Reference (MelodAI)

Core application and infrastructure variables from `.env.example`:

Variable	Purpose
`APP_NAME`	Application display name
`APP_ENV`	local | production
`APP_DEBUG`	true locally, false in production
`APP_URL`	Public HTTPS base URL of Laravel app
`CRON_SECRET_TOKEN`	Optional fallback for HTTP /cron/* endpoints
`DB_*`	MySQL connection credentials
`QUEUE_CONNECTION`	database (default) or redis with Horizon on VPS
`CACHE_STORE`	database or redis
`SANCTUM_TOKEN_EXPIRATION`	Mobile API token lifetime in minutes
`FILESYSTEM_DISK`	local or cloudflare (R2)
`FFMPEG_BINARIES / FFPROBE_BINARIES`	Paths to FFmpeg binaries
`SUPER_ADMIN_EMAIL`	Email for super admin notifications
`TELESCOPE_ENABLED`	false in production

Feature flags:

Variable	Purpose
`FEATURE_BLOG`	Enable blog module
`FEATURE_SOCIAL_LOGIN`	Show OAuth login buttons when keys exist

Variable	Purpose
`FEATURE_TWO_FACTOR`	Enable 2FA for users
`DEFAULT_CREDITS_FREE`	Free credits for new users

Admin Settings Mapping

- Site name, Site URL (cron base), contact emails
- OpenAI, xAI, Suno, FAL defaults
- SMTP host, port, credentials (overrides .env at runtime)
- Stripe, Flutterwave, Revolut, Cryptomus toggles and keys
- OAuth client and redirect URI
- Turnstile keys
- HTTP cron URLs and cPanel commands

PART 6

Expo 56 Mobile App (melodai-app)

Install and run the MelodAI React Native app on Windows, macOS, and Linux. Covers Expo Go, API configuration, production builds, and Android APK/AAB release.

Requirements

- and npm
- via `npx expo` (project uses Expo SDK 56)
-
- Android Studio, JDK 17, Android SDK (for emulator or local APK builds)
- Xcode 15+, CocoaPods, Apple Developer account for device/store builds
- Running MelodAI Laravel API (local XAMPP or production HTTPS server)

Project path: `melodai-app/` (sibling to Laravel root). Package: `com.melodai.app`, scheme: `melodaiapp`.

Step 1: Install Dependencies

```
cd melodai-app
```

```
npm install
```

Verify Expo version:

```
npx expo --version
```

Step 2: Configure Environment

Copy and edit `melodai-app/.env.example` → `.env`:

```
copy .env.example .env
```

```
EXPO_PUBLIC_API_URL=http://localhost/melodai/public/api/v1
```

Rules:

- Use the Laravel API base including `/api/v1` (no trailing slash).
- Physical phone on same Wi-Fi: replace `localhost` with your PC IPv4 from `ipconfig` / `ifconfig`.
- Release APK/AAB use HTTPS production URL — Expo Go can infer LAN; release builds cannot.

Examples:

```
EXPO_PUBLIC_API_URL=http://192.168.1.10/melodai/public/api/v1
```

```
EXPO_PUBLIC_API_URL=https://melodai.site/api/v1
```

After changing `.env`, restart Metro with cache clear: `npx expo start -c`. Production builds bake the URL via `app.config.js` → `extra.apiUrl`.

Step 3: Run with Expo Go

```
npx expo start
```

1. Install on your phone (SDK 56 compatible version).
2. Scan the QR code from the terminal (same network as dev machine).
3. Press `a` for Android emulator or `i` for iOS simulator (macOS).

Windows Development

- Install Android Studio → SDK Platform 34+, Build-Tools, Android Emulator.
- Set `ANDROID_HOME` (e.g. `C:\Users\You\AppData\Local\Android\Sdk`).
- Add `platform-tools` to PATH.

Start Laravel API (from project root):

```
composer run dev
```

Start Expo:

```
cd melodai-app && npx expo start
```

Run native Android project (dev client / release build):

```
npx expo run:android
```

macOS & iOS Development

```
xcode-select --install
```

```
cd melodai-app && npx expo run:ios
```

For App Store builds, configure signing in Xcode (`ios/` folder after prebuild) with your Apple Team ID.

Linux Development

- Install Node 20, Watchman (optional), Android SDK for emulator.
- Use production or LAN API URL in `.env`.

```
cd melodai-app && npx expo start
```

```
npx expo run:android
```

Production API Configuration

Before any store or APK release:

1. Deploy Laravel with HTTPS and valid SSL.
2. Confirm API works: `https://yourdomain.com/api/v1/...`
3. Set `EXPO_PUBLIC_API_URL=https://yourdomain.com/api/v1` in `melodai-app/.env`.
4. Rebuild native app (EAS or local Gradle) so env is embedded.

YouTube connect from mobile requires Google OAuth redirect registered for production HTTPS (see Providers doc).

EAS Build (Recommended for Production)

Expo Application Services builds signed APK/AAB in the cloud. Install EAS CLI and log in:

```
npm install -g eas-cli
```

```
eas login
```

```
cd melodai-app && eas build:configure
```

Set production env in `eas.json` or EAS secrets:

```
eas secret:create --name EXPO_PUBLIC_API_URL --value https://melodai.site/api/v1
```

Build Android APK (preview) or AAB (Play Store):

```
eas build --platform android --profile preview
```

```
eas build --platform android --profile production
```

iOS:

```
eas build --platform ios --profile production
```

Update `app.json` → `extra.eas.projectId` with your EAS project ID after linking.

Local Android APK (Gradle)

Generate native Android project and build release APK locally:

```
cd melodai-app
```

```
npx expo prebuild --platform android
```

```
cd android && .\gradlew assembleRelease
```

APK output (typical path):

```
melodai-app/android/app/build/outputs/apk/release/app-release.apk
```

Signing: create a keystore and configure `android/app/build.gradle` signingConfigs for Play Store uploads.

> If CMake/NDK errors occur on Windows, clean `android/.cxx` and run `gradlew clean` before rebuild.

Store Release Checklist

- [ ] Production `EXPO_PUBLIC_API_URL` baked into build
- [ ] App icons and splash assets in `assets/images/`
- [ ] Version bumped in `app.json` (`version`, Android `versionCode`)
- [ ] `usesCleartextTraffic` disabled for production HTTPS-only (update `app.json` android section)
- [ ] Privacy policy URL and store listing prepared
- [ ] Test login, registration, music generation, and profile on release build

Troubleshooting (Mobile)

- ensure production API URL is set before build; clear app data and reinstall.
- API unreachable — check URL, HTTPS cert, and firewall; on LAN use PC IP not localhost.
- register correct redirect URI in Google Cloud; use server-side connect flow.
- release build missing `EXPO_PUBLIC_API_URL` — rebuild with env set.
- `npx expo start -c`