MyRefera Backend Documentation

Table of Contents

Project Overview

MyRefera Backend is a comprehensive Node.js/Express.js API that powers the MyRefera influencer marketing platform. The backend provides RESTful APIs for user management, campaign management, viral video processing, AI-powered script tailoring, and transcript generation.

Key Features:

- **User Management**: Multi-role authentication (Brands, Creators, Admins)

- **Campaign Management**: Complete campaign lifecycle management
- **Viral Video Processing**: Video transcript generation and AI tailoring
- **AI Integration**: Advanced AI-powered script customization
- **Database Management**: PostgreSQL with Sequelize ORM
- **Real-time Features**: Socket.io integration for messaging

Technology Stack

Core Technologies

- **Runtime**: Node.js
- **Framework**: Express.js
- **Database**: PostgreSQL
- **ORM**: Sequelize
- **Authentication**: JWT (JSON Web Tokens)
- **Validation**: Zod
- **Language**: TypeScript

Key Dependencies

```
{  "dependencies": {  "express": "^4.18.2",  "sequelize": "^6.35.0",  "pg": "^8.11.3",  "jsonwebtoken": "^9.0.2",  "zod": "^3.22.4",  "socket.io": "^4.7.4",  "bcryptjs": "^2.4.3",  "cors": "^2.8.5",  "helmet": "^7.1.0",  "dotenv": "^16.3.1"  }}
```

Database Schema

New Tables Added

Video Transcripts Table

```
CREATE TABLE video_transcripts (
    id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL,
    user_role VARCHAR(10) NOT NULL CHECK (user_role IN ('brand', 'creato
r')),
    video_url VARCHAR(2048) NOT NULL,
    video_title VARCHAR(500),
    platform VARCHAR(50) DEFAULT 'tiktok',
    transcript_text TEXT NOT NULL,
    language VARCHAR(10) DEFAULT 'en',
    duration INTEGER,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE);
```

Tailored Scripts Table

```
CREATE TABLE tailored_scripts (
    id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL,
    user_role VARCHAR(10) NOT NULL CHECK (user_role IN ('brand', 'creato
r')),
    video_url VARCHAR(2048) NOT NULL,
    transcript_id INTEGER,
    product_name VARCHAR(200),
    product_description TEXT,
    product_url VARCHAR(2048),
    brand_tone VARCHAR(50),
    target_audience VARCHAR(200),
    tailored_script TEXT,
    section_breakdown JSONB,
    sutherland_alchemy JSONB,
    hormozi_value_stack JSONB,
```

```
    confidence DECIMAL(3,2),
    processing_time DECIMAL(8,3),
    word_count INTEGER,
    estimated_read_time VARCHAR(20),
    api_version VARCHAR(20),
    model_used VARCHAR(50),
    improvement_areas JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (transcript_id) REFERENCES video_transcripts(id) ON DELET
E CASCADE);
```

API Endpoints

Transcript Endpoints

POST /api/transcripts

Purpose: Create a new video transcript

Request Body:

```
{
  videoUrl: string;           // Required: Video URL (max 2048 chars)  videoTitle?:
string;        // Optional: Video title (max 500 chars)  platform?: string;          // O
ptional: Platform (default: 'tiktok')  transcriptText: string;     // Required: Transc
ript text (max 100,000 chars)  language?: string;          // Optional: Language c
ode (default: 'en')  duration?: number;          // Optional: Duration in seconds}
```

Response:

```
{
  success: boolean;  data?: {
    id: number;    videoUrl: string;    videoTitle?: string;    transcriptText: string;
```

```
language: string;    duration?: number;    generatedAt: string;  };  error?: strin
g;}
```

GET /api/transcripts

Purpose: Get all transcripts for authenticated user

Query Parameters:

- `limit`: Number of results (1-100, default: 50)
- `offset`: Number of results to skip (default: 0)

Response:

```
{
  success: boolean;  data?: {
    transcripts: Array<{
      id: number;      videoUrl: string;      videoTitle?: string;      transcriptText: stri
ng;      language: string;      duration?: number;      generatedAt: string;    }>;    t
otal: number;  };  error?: string;}
```

GET /api/transcripts/:videoUrl

Purpose: Get transcript by video URL (URL-encoded)

Response: Same as POST /api/transcripts data format

DELETE /api/transcripts/:id

Purpose: Delete transcript by ID

Response:

```
{
  success: boolean;  error?: string;}
```

Tailored Script Endpoints

POST /api/tailored-scripts

Purpose: Create a new tailored script

Request Body:

```
{
  videoUrl: string;                    // Required: Video URL  transcriptId?: number;
// Optional: Associated transcript ID  productName: string;                 // Require
d: Product name (max 200 chars)  productDescription: string;          // Require
d: Product description (max 5000 chars)  productUrl?: string;                // Opti
onal: Product URL  brandTone?: string;                // Optional: Brand tone (max
50 chars)  targetAudience?: string;            // Optional: Target audience (max 2
00 chars)  // AI Response Data  tailoredScript: string;              // Required: AI-g
enerated script  sectionBreakdown: SectionBreakdown[]; // Required: Script a
nalysis  sutherlandAlchemy: SutherlandAlchemy; // Required: Value reframing
hormoziValueStack: HormoziValueStack; // Required: Value stacking  // Metad
ata  confidence: number;                // Required: AI confidence (0-1)  processin
gTime: number;              // Required: Processing time in seconds  wordCount:
number;                  // Required: Word count  estimatedReadTime: string;
// Required: Estimated read time  apiVersion?: string;                // Optional: API
version  modelUsed?: string;                // Optional: AI model used  improveme
ntAreas?: string[];         // Optional: Improvement suggestions}
```

Response:

```
{
  success: boolean;  data?: {
    id: number;    videoUrl: string;    productName: string;    generatedAt: string;
};  error?: string;}
```

GET /api/tailored-scripts

Purpose: Get all tailored scripts for authenticated user

Query Parameters: Same as transcripts (limit, offset)

Response:

```
{
  success: boolean;  data?: {
    scripts: Array<{
      id: number;      videoUrl: string;      productName: string;      productDescrip
tion: string;      tailoredScript: string;      sectionBreakdown: any[];      confiden
ce: number;      wordCount: number;      generatedAt: string;    }>;    total: num
ber;  };  error?: string;}
```

GET /api/tailored-scripts/:videoUrl

Purpose: Get tailored script by video URL with full details

Response:

```
{
  success: boolean;  data?: {
    id: number;    videoUrl: string;    productName: string;    productDescription:
string;    tailoredScript: string;    sectionBreakdown: any[];    sutherlandAlchem
y: any;    hormoziValueStack: any;    confidence: number;    processingTime: n
umber;    wordCount: number;    estimatedReadTime: string;    generatedAt: str
ing;  };  error?: string;}
```

DELETE /api/tailored-scripts/:id

Purpose: Delete tailored script by ID

Response:

```
{
  success: boolean;  error?: string;}
```

Legacy AI Tailoring Endpoint

POST /ai-tailor-script

Purpose: Legacy endpoint for AI script tailoring (backward compatibility)

Authentication: Required (brand, creator roles)

Request Body: Compatible with frontend AI tailoring service

Response: AI tailoring results with psychological analysis

New Features - Transcripts & AI Tailoring

Video Transcript System

Features

- **Multi-platform Support**: TikTok, Instagram, YouTube
- **Language Support**: Multi-language transcript storage
- **User Association**: Transcripts linked to specific users
- **Duplicate Prevention**: Automatic update of existing transcripts
- **Soft Deletes**: Paranoid deletion with timestamps

Data Flow

1. **Frontend Request**: User requests transcript for video
2. **External API Call**: Call to transcript generation service
3. **Database Storage**: Store transcript with metadata
4. **Response**: Return transcript data to frontend

AI Tailoring System

Advanced Psychology Integration

- **Sutherland Alchemy**: Value reframing and identity shifts
- **Hormozi Value Stack**: Comprehensive value proposition building
- **Section Breakdown**: Psychological analysis of script sections
- **Confidence Scoring**: AI confidence metrics

Data Structures

Section Breakdown

```
interface SectionBreakdown {
  sectionName: string; // "Hook", "Transformation Story", "CTA"  triggerEmo
tionalState: string; // "Curiosity Gap + Social Proof"  originalQuote: string; //
Original transcript text  rewrittenVersion: string; // AI-rewritten version  scen
eDescription: string; // Visual direction for filming  psychologicalPrinciples: st
ring[]; // Applied psychology principles  timestamp?: string; // Video timesta
mp}
```

Sutherland Alchemy

```
interface SutherlandAlchemy {
  explanation: string; // How the reframing works  valueReframing: Array<{
    original: string; // Original framing  reframed: string; // New framing
psychologyBehind: string; // Psychological explanation  }>;  identityShifts: s
tring[]; // Identity transformations}
```

Hormozi Value Stack

```
interface HormoziValueStack {
  coreOffer: string; // Main product offer  valueElements: Array<{
    element: string; // Value component name  perceivedValue: string; //
Perceived value  actualCost: string; // Actual cost  }>;  totalStack: {
    totalPerceivedValue: string; // Total perceived value  actualPrice: strin
g; // Actual price  valueMultiplier: string; // Value multiplier (e.g., "7x+ valu
e")  };  grandSlamElements: string[]; // High-impact value elements}
```

Controllers

Transcript Controller

File: `src/controllers/transcript.controller.ts`

Methods:

- `createTranscript()` - Create new transcript

- `getTranscripts()` - Get user's transcripts with pagination
- `getTranscriptByVideoUrl()` - Get specific transcript by URL
- `deleteTranscript()` - Delete transcript by ID

Features:

- Input validation using Zod schemas
- User authentication and authorization
- Error handling with appropriate HTTP status codes
- URL decoding for video URL parameters

Tailored Script Controller

File: `src/controllers/tailoredScript.controller.ts`

Methods:

- `createTailoredScript()` - Create new tailored script
- `getTailoredScripts()` - Get user's tailored scripts with pagination
- `getTailoredScriptByVideoUrl()` - Get specific script by URL
- `deleteTailoredScript()` - Delete script by ID

Features:

- Complex data validation for AI response structures
- Transcript ownership verification
- Comprehensive error handling
- Support for updating existing scripts

Services

Video Transcript Service

File: `src/features/common/videoTranscripts/videoTranscripts.service.ts`

Key Methods:

```
class VideoTranscriptService {
    async createTranscript(request, authUser, transaction?): Promise<TranscriptResponse>;    async getTranscripts(authUser, query): Promise<TranscriptsListResponse>;    async getTranscriptByVideoUrl(videoUrl, authUser): Promise<TranscriptResponse>;    async deleteTranscript(transcriptId, authUser): Promise<DeleteTranscriptResponse>;    async getTranscriptById(transcriptId, authUser): Promise<any>;}
```

Features:

- **Duplicate Handling**: Updates existing transcripts instead of creating duplicates
- **Transaction Support**: Database transaction support for data consistency
- **User Isolation**: Users can only access their own transcripts
- **Soft Deletes**: Paranoid deletion with recovery capability

Tailored Script Service

File: `src/features/common/tailoredScripts/tailoredScripts.service.ts`

Key Methods:

```
class TailoredScriptService {
    async createTailoredScript(request, authUser, transaction?): Promise<TailoredScriptResponse>;    async getTailoredScripts(authUser, query): Promise<TailoredScriptsListResponse>;    async getTailoredScriptByVideoUrl(videoUrl, authUser): Promise<TailoredScriptDetailResponse>;    async deleteTailoredScript(scriptId, authUser): Promise<{ success: boolean; error?: string }>;    async getTailoredScriptById(scriptId, authUser): Promise<any>;}
```

Features:

- **Complex Data Storage**: JSONB storage for AI analysis results
- **Transcript Integration**: Links to video transcripts when available
- **Update Logic**: Updates existing scripts instead of creating duplicates

- **Comprehensive Validation**: Validates all AI response structures

Models

Video Transcripts Model

File: `src/features/common/videoTranscripts/videoTranscripts.model.ts`

Attributes:

```
interface VideoTranscriptAttributes {
    userId: number; // User ID (foreign key)    userRole: 'brand' | 'creator'; // Us
er role    videoUrl: string; // Video URL (max 2048 chars)    videoTitle?: string;
// Video title (max 500 chars)    platform: string; // Platform (default: 'tiktok')
transcriptText: string; // Transcript content    language: string; // Language co
de (default: 'en')    duration?: number; // Duration in seconds}
```

Features:

- **Snake Case Mapping**: Automatic camelCase to snake_case conversion
- **Soft Deletes**: Paranoid deletion with timestamps
- **User Association**: Links to user profiles
- **Platform Support**: Multi-platform video support

Tailored Scripts Model

File: `src/features/common/tailoredScripts/tailoredScripts.model.ts`

Attributes:

```
interface TailoredScriptAttributes {
    userId: number; // User ID (foreign key)    userRole: 'brand' | 'creator'; // Us
er role    videoUrl: string; // Video URL (max 2048 chars)    transcriptId?: numb
er; // Associated transcript ID    productName?: string; // Product name (max 2
00 chars)    productDescription?: string; // Product description    productUrl?:
string; // Product URL (max 2048 chars)    brandTone?: string; // Brand tone
(max 50 chars)    targetAudience?: string; // Target audience (max 200 chars)
tailoredScript?: string; // AI-generated script    sectionBreakdown?: any; // JS
```

```
ONB: Script analysis    sutherlandAlchemy?: any; // JSONB: Value reframing    hormoziValueStack?: any; // JSONB: Value stacking    confidence?: number; // AI confidence (0-1)    processingTime?: number; // Processing time in seconds    wordCount?: number; // Word count    estimatedReadTime?: string; // Estimated read time    apiVersion?: string; // API version    modelUsed?: string; // AI model used    improvementAreas?: any; // JSONB: Improvement suggestions}
```

Features:

- **JSONB Storage**: Efficient storage of complex AI analysis data
- **Foreign Key Relations**: Links to transcripts and users
- **Flexible Schema**: Supports various AI response formats
- **Metadata Tracking**: Comprehensive tracking of AI processing

Routes

Transcript Routes

File: `src/routes/transcript.routes.ts`

Route Configuration:

```
// All routes require authenticationrouter.use(middlewares.verifyToken(['brand', 'creator']));// Route definitionsrouter.post('/', transcriptController.createTranscript);router.get('/', transcriptController.getTranscripts);router.get('/:videoUrl', transcriptController.getTranscriptByVideoUrl);router.delete('/:id', transcriptController.deleteTranscript);
```

Tailored Script Routes

File: `src/routes/tailoredScript.routes.ts`

Route Configuration:

```
// All routes require authenticationrouter.use(middlewares.verifyToken(['brand', 'creator']));// Route definitionsrouter.post('/', tailoredScriptController.createTailoredScript);router.get('/', tailoredScriptController.getTailoredScripts);route
```

```
r.get('/:videoUrl', tailoredScriptController.getTailoredScriptByVideoUrl);router.
delete('/:id', tailoredScriptController.deleteTailoredScript);
```

Main Routes Integration

File: `src/routes/index.ts`

New Route Registrations:

```
app.use('/transcripts', transcriptRoute);app.use('/tailored-scripts', tailoredScri
ptRoute);// Legacy AI Tailoring endpointapp.post('/ai-tailor-script', middleware
s.verifyToken(['brand', 'creator']), async (req, res, next) => {
  // Dynamic import and processing});
```

Database Relations

Updated Models Configuration

File: `src/database/models.ts`

New Relations Added:

Video Transcripts Relations

```
setupRelation({
  child: { model: videoTranscripts.model, names: videoTranscripts.names },
parents: [
    { model: brands.model, names: brands.names, key: 'userId', child: 'many'
},    { model: creators.model, names: creators.names, key: 'userId', child: 'm
any' },  ],});
```

Tailored Scripts Relations

```
setupRelation({
  child: { model: tailoredScripts.model, names: tailoredScripts.names },  pare
nts: [
    { model: brands.model, names: brands.names, key: 'userId', child: 'many'
```

```
}, { model: creators.model, names: creators.names, key: 'userId', child: 'many' }, { model: videoTranscripts.model, names: videoTranscripts.names, key: 'transcriptId', child: 'many' }, ],});
```

Sequelize Helper

File: `src/helpers/sequelize.helper.new.ts`

Features:

- **Snake Case Conversion**: Automatic camelCase to snake_case field mapping
- **Model Generation**: Dynamic model creation with proper naming
- **Table Naming**: Consistent table naming conventions
- **Alias Management**: Automatic alias generation for relations

Error Handling

Validation Errors

- **Zod Validation**: Comprehensive input validation with detailed error messages
- **Type Safety**: TypeScript interfaces for all request/response structures
- **Field Validation**: Length limits, format validation, required field checks

Database Errors

- **Transaction Support**: Rollback on errors for data consistency
- **Foreign Key Constraints**: Proper relationship enforcement
- **Duplicate Handling**: Graceful handling of duplicate entries

API Errors

- **HTTP Status Codes**: Appropriate status codes for different error types
- **Error Messages**: User-friendly error messages
- **Logging**: Comprehensive error logging for debugging

Authentication & Authorization

JWT Authentication

- **Token Verification**: Middleware for token validation
- **Role-based Access**: Different access levels for brands, creators, admins
- **User Context**: Automatic user context injection in requests

Route Protection

```
// All transcript and tailored script routes require authenticationrouter.use(mid
dlewares.verifyToken(['brand', 'creator']));
```

User Isolation

- **Data Segregation**: Users can only access their own data
- **Ownership Verification**: Verification of data ownership before operations
- **Role Validation**: Role-based access control for different features

Development Setup

Prerequisites

- Node.js 18+
- PostgreSQL 13+
- npm or yarn

Installation

```
# Clone repositorygit clone <repository-url>cd my-refera-be
# Install dependenciesnpm install
# Set up environment variablescp .env.example .env
# Edit .env with your configuration# Run database migrationsnpm run migrate
# Start development servernpm run dev
```

Environment Variables

```
# DatabaseDATABASE_URL=postgresql://username:password@localhost:543
2/myrefera
DB_HOST=localhost
DB_PORT=5432
DB_NAME=myrefera
DB_USER=username
DB_PASSWORD=password
# JWTJWT_SECRET=your-jwt-secret
JWT_EXPIRES_IN=30d
# ServerPORT=8000
NODE_ENV=development
# External APIsTRANSCRIPT_API_URL=https://script-api.myrefera.com
AI_TAILORING_API_URL=https://script-api.myrefera.com
```

Available Scripts

```
npm run dev          # Start development servernpm run build        # Build for pr
oductionnpm run start        # Start production servernpm run migrate      # Run
database migrationsnpm run seed         # Seed database with test datanpm ru
n test         # Run testsnpm run lint         # Run ESLint
```