Full-Stack Video Transcoding Platform - Technical Documentation

i. System Architecture

Application Review

VideoTranscode Pro is a cloud-native, scalable video processing platform that enables users to upload video files and transcode them into multiple formats and resolutions asynchronously. The system follows microservices architecture with clear separation of concerns between frontend, backend, and processing workers.

Technology Stack

Frontend Layer

- **Framework**: React 18+ with TypeScript
- **State Management**: Redux Toolkit with RTK Query
- **Styling**: Tailwind CSS + Material-UI components
- **File Upload**: Axios with progress tracking, Resumable.js for large files
- **Real-time Updates**: Socket.IO client for progress notifications
- **Build Tool**: Vite for faster builds and HMR
- **Testing**: Jest + React Testing Library + Cypress E2E

Backend Layer

- **API Framework**: FastAPI (Python 3.9+)
- **Authentication**: JWT with OAuth2.0 flows
- **ORM**: SQLAlchemy 2.0 with async support
- **Database**: PostgreSQL 14 with TimescaleDB for metrics
- **Message Broker**: RabbitMQ 3.11 with management plugin
- **Cache**: Redis 7 for session storage and rate limiting
- **Object Storage**: AWS S3/MinIO for video storage

Processing Layer

- **Transcoding Engine**: FFmpeg 6.0 with NVIDIA GPU support (if available)
- **Workers**: Python Celery workers or custom Python workers
- **Video Analysis**: OpenCV for thumbnail generation
- **File Handling**: Watchdog for file system monitoring

Infrastructure

- **Containerization**: Docker 20.10+ with Docker Compose
- **Orchestration**: Kubernetes (production) / Docker Swarm (staging)
- **Load Balancer**: NGINX Plus with WebSocket support
- **CDN**: CloudFront/Akamai for content delivery
- **Monitoring Stack**: Prometheus + Grafana + ELK Stack
- **CI/CD**: GitHub Actions + ArgoCD for GitOps

Third-Party Integrations

- **Payment Processing**: Stripe for subscription billing
- **Email Service**: SendGrid/Mailgun for notifications
- **Analytics**: Mixpanel/Amplitude for user behavior
- **Error Tracking**: Sentry for real-time error monitoring
- **CDN**: AWS CloudFront for global content distribution
- **Storage**: AWS S3 with lifecycle policies

ii. Database Schema

Core Tables

```sql
-- Users and Authentication
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    username VARCHAR(100) UNIQUE,
    password_hash VARCHAR(255),
    is_active BOOLEAN DEFAULT true,
    is_verified BOOLEAN DEFAULT false,
    tier VARCHAR(20) DEFAULT 'free', -- free, pro, enterprise
    storage_limit BIGINT DEFAULT 1073741824, -- 1GB in bytes
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    last_login TIMESTAMP WITH TIME ZONE
);

-- User sessions for JWT invalidation
CREATE TABLE user_sessions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    jwt_id VARCHAR(100) UNIQUE NOT NULL,
    device_info JSONB,
    ip_address INET,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP WITH TIME ZONE NOT NULL,
    revoked_at TIMESTAMP WITH TIME ZONE
);

-- Video uploads
CREATE TABLE videos (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    original_filename VARCHAR(500) NOT NULL,
    file_size BIGINT NOT NULL,
    duration DECIMAL(10,3), -- in seconds
    original_format VARCHAR(10),
    resolution VARCHAR(20),
    bitrate INTEGER,
    storage_path VARCHAR(1000) NOT NULL,
    thumbnail_path VARCHAR(1000),
    metadata JSONB,
    upload_status VARCHAR(20) DEFAULT 'pending', -- pending, uploading, uploaded, failed
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    uploaded_at TIMESTAMP WITH TIME ZONE
);

-- Transcoding jobs
CREATE TABLE transcoding_jobs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    video_id UUID REFERENCES videos(id) ON DELETE CASCADE,
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    status VARCHAR(20) DEFAULT 'queued', -- queued, processing, completed, failed
    source_format VARCHAR(10),
    target_format VARCHAR(10) NOT NULL,
    quality_preset VARCHAR(20), -- 360p, 720p, 1080p, 4k
    ffmpeg_params JSONB,
    priority INTEGER DEFAULT 5, -- 1=highest, 10=lowest
    queued_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    started_at TIMESTAMP WITH TIME ZONE,
    completed_at TIMESTAMP WITH TIME ZONE,
    error_message TEXT,
    output_path VARCHAR(1000),
    output_size BIGINT,
    processing_time INTEGER -- in seconds
);

-- Processed videos
CREATE TABLE processed_videos (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    job_id UUID REFERENCES transcoding_jobs(id) ON DELETE CASCADE,
    video_id UUID REFERENCES videos(id) ON DELETE CASCADE,
    format VARCHAR(10) NOT NULL,
    resolution VARCHAR(20),
    bitrate INTEGER,
    file_size BIGINT,
    storage_path VARCHAR(1000) NOT NULL,
    playback_url VARCHAR(1000),
    is_default BOOLEAN DEFAULT false,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- User subscriptions
```

```sql
CREATE TABLE subscriptions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE UNIQUE,
    stripe_subscription_id VARCHAR(100) UNIQUE,
    stripe_customer_id VARCHAR(100),
    plan VARCHAR(50) NOT NULL,
    status VARCHAR(20) DEFAULT 'active',
    current_period_start TIMESTAMP WITH TIME ZONE,
    current_period_end TIMESTAMP WITH TIME ZONE,
    cancel_at_period_end BOOLEAN DEFAULT false,
    canceled_at TIMESTAMP WITH TIME ZONE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- API usage tracking
CREATE TABLE api_usage (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    endpoint VARCHAR(100) NOT NULL,
    method VARCHAR(10) NOT NULL,
    status_code INTEGER,
    request_size BIGINT,
    response_size BIGINT,
    processing_time INTEGER,
    timestamp TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Indexes for performance
CREATE INDEX idx_videos_user_id ON videos(user_id);
CREATE INDEX idx_videos_upload_status ON videos(upload_status);
CREATE INDEX idx_transcoding_jobs_status ON transcoding_jobs(status);
CREATE INDEX idx_transcoding_jobs_user_id ON transcoding_jobs(user_id);
CREATE INDEX idx_processed_videos_video_id ON processed_videos(video_id);
CREATE INDEX idx_api_usage_user_timestamp ON api_usage(user_id, timestamp DESC);
CREATE INDEX idx_transcoding_jobs_priority_status ON transcoding_jobs(priority, status);

-- TimescaleDB hypertable for metrics
SELECT create_hypertable('api_usage', 'timestamp');
```

Relationship Summary

```
users (1) ──┐
            ├── (many) videos
            ├── (many) transcoding_jobs
            ├── (1) subscriptions
            └── (many) api_usage

videos (1) ──┐
             ├── (many) transcoding_jobs
             └── (many) processed_videos

transcoding_jobs (1) ─── (1) processed_videos
```

iii. API Specifications

Authentication

- **Method**: JWT (JSON Web Tokens) with Bearer scheme
- **Token Type**: Access Token (15min expiry) + Refresh Token (7 days)
- **OAuth2 Flows**: Password Grant, Refresh Token Grant
- **Rate Limiting**: 100 requests/minute per user, 1000 requests/minute per IP

Base URL

- **Production**: `https://api.videotranscode.com/v1`
- **Staging**: `https://staging-api.videotranscode.com/v1`
- **Local**: `http://localhost:8000/v1`

Core Endpoints

Authentication

```
POST /auth/register
POST /auth/login
POST /auth/refresh
POST /auth/logout
DELETE /auth/sessions/{session_id}
GET  /auth/me
```

Video Management

```
POST   /videos/upload            # Initiate upload with presigned URL
PUT    /videos/{id}/chunk        # Upload chunk (resumable)
POST   /videos/{id}/complete     # Complete upload
GET    /videos                   # List user videos
GET    /videos/{id}              # Get video details
DELETE /videos/{id}              # Delete video
POST   /videos/{id}/transcode    # Request transcoding
GET    /videos/{id}/jobs         # Get transcoding jobs for video
```

Transcoding Jobs

```
GET    /jobs                     # List user's jobs
GET    /jobs/{id}                # Get job details
GET    /jobs/{id}/progress       # WebSocket/SSE for progress
POST   /jobs/{id}/cancel         # Cancel job
GET    /jobs/{id}/outputs        # List processed outputs
```

Subscription & Billing

```
GET    /subscription             # Get current subscription
POST   /subscription/upgrade     # Upgrade subscription
POST   /subscription/cancel      # Cancel subscription
GET    /subscription/invoices    # List invoices
GET    /billing/portal           # Generate billing portal URL
```

System Status

```
GET    /health                   # Health check
GET    /metrics                  # Prometheus metrics
GET    /status                   # System status
```

Example Request/Response

```
POST /videos/upload
Content-Type: application/json
Authorization: Bearer {jwt_token}

{
  "filename": "vacation.mp4",
  "file_size": 104857600,  # 100MB
  "content_type": "video/mp4"
}

Response:
{
  "video_id": "550e8400-e29b-41d4-a716-446655440000",
  "upload_id": "unique_upload_id",
  "chunk_size": 5242880,  # 5MB
  "upload_urls": [
    "https://storage.s3.amazonaws.com/presigned-url-1",
    "https://storage.s3.amazonaws.com/presigned-url-2"
  ]
}
```

iv. Integration Requirements

Payment Integration (Stripe)

- **Purpose**: Subscription management for different tiers
- **Features Required**:
  - Monthly/Annual subscription plans
  - Free trial (14 days)
  - Upgrade/downgrade functionality
  - Invoice generation and email delivery
  - Failed payment handling with retry logic
  - Webhook handling for subscription events
- **Security**: PCI DSS Level 1 compliance required
- **Implementation**:
  - Use Stripe Elements for secure card capture
  - Never store card details in our database

    - Use Stripe webhooks to sync subscription status
    - Implement idempotency keys for webhook handling

Email API Integration (SendGrid)

- **Purpose**: User notifications and transactional emails
- **Email Types Required**:
    - Welcome email with verification link
    - Email verification
    - Password reset
    - Video processing complete
    - Video processing failed
    - Storage quota warnings
    - Subscription receipts
    - Account security alerts
- **Template Management**: Use SendGrid dynamic templates
- **Metrics**: Track open rates, click-through rates, bounce rates

Push Notifications (Optional)

- **Purpose**: Real-time updates for mobile/web clients
- **Channels**:
    - WebSocket for real-time progress updates
    - Firebase Cloud Messaging for mobile apps
    - Web Push API for browser notifications
- **Events to Notify**:
    - Upload completed
    - Transcoding started
    - Transcoding progress (every 10%)
    - Transcoding completed/failed
    - Storage limit approaching (80%, 90%, 100%)

v. Data Synchronization

Real-time Requirements

- **Upload Progress**: Real-time percentage updates during file upload
- **Transcoding Progress**: Live updates on processing status (10% increments)

- **Job Status Changes**: Immediate notification on job state changes
- **Storage Updates**: Real-time quota updates
- **Implementation**:
    - WebSocket connections for persistent bidirectional communication
    - Server-Sent Events (SSE) for simpler progress updates
    - Redis Pub/Sub for internal event propagation
    - Message queue for guaranteed delivery of critical updates

Offline Support

- **Upload Resumption**: Support for resuming interrupted uploads
- **Job Queue Management**: Queue jobs when offline, sync when back online
- **Local Cache**: Cache recent videos and jobs
- **Implementation**:
    - Use IndexedDB for storing upload metadata and chunks
    - Implement service worker for background sync
    - Store pending jobs in local storage with timestamps
    - Conflict resolution with server-side timestamps (Last-Write-Wins)

Background Sync

- **Service Worker**: For handling uploads in background
- **Sync Strategies**:
    - Immediate sync for small files (<50MB)
    - Delayed sync for large files
    - Periodic sync every 15 minutes for job status checks
- **Retry Logic**: Exponential backoff with jitter for failed syncs
- **Battery Optimization**: Defer large syncs when on battery power

vi. Security & Compliance

Authentication & Authorization

- **Multi-factor Authentication**: Optional for enterprise users
- **OAuth2.0 Providers**: Google, GitHub, Microsoft
- **Role-Based Access Control (RBAC)**:
    - User: Can manage own videos
    - Admin: Can manage all users and system settings

    - Transcoder: Service role for processing only
- **API Keys**: For programmatic access with scoped permissions
- **Session Management**: JWT with rotating refresh tokens, session invalidation

Data Encryption

- **In Transit**: TLS 1.3 for all communications
- **At Rest**:
    - Database: PostgreSQL native encryption + application-level encryption for sensitive data
    - Storage: AES-256 server-side encryption on S3/MinIO
    - Secrets: HashiCorp Vault for secret management
- **Key Management**: AWS KMS or HashiCorp Vault for key rotation
- **Encryption Standards**: FIPS 140-2 validated modules where applicable

GDPR Compliance

- **Data Protection Officer**: Appointed team member
- **Data Processing Agreement**: Available for all customers
- **Privacy by Design**: Default privacy settings
- **User Rights**:
    - Right to access all personal data
    - Right to rectification
    - Right to erasure (video deletion within 24 hours)
    - Right to data portability (export in JSON format)
    - Right to object to processing
- **Data Retention Policy**:
    - Videos: Deleted after 30 days of inactivity for free tier, 90 days for paid
    - Logs: Retained for 90 days, anonymized after 30 days
    - Backups: Encrypted and retained for 30 days

PCI Compliance

- **Payment Data**: Never stored, processed through Stripe Elements
- **Network Security**: Segmented network, WAF protection
- **Vulnerability Management**: Quarterly penetration testing
- **Access Controls**: Multi-factor authentication for admin access
- **Audit Logging**: All payment-related actions logged

vii. Performance Requirements

Response Time Targets (P95)

- **API Responses**: < 200ms for read operations, < 500ms for write operations
- **Video Upload Start**: < 100ms for presigned URL generation
- **Transcoding Queue Time**: < 5 seconds for job acceptance
- **Video Playback Start**: < 2 seconds for initial buffering
- **Dashboard Load**: < 1.5 seconds for full page load

Scalability Targets

- **Concurrent Users**: 10,000 concurrent authenticated users
- **Video Uploads**: 100 concurrent uploads (max 10GB each)
- **Transcoding Jobs**: 500 concurrent transcoding jobs
- **Throughput**: 1TB/hour video processing capacity
- **Storage**: Petabyte-scale object storage

Database Optimization

- **Connection Pooling**: PgBouncer for PostgreSQL connection management
- **Read Replicas**: 3 read replicas for geographic distribution
- **Sharding Strategy**: User-based sharding for videos table
- **Indexing**: Composite indexes for common query patterns
- **Caching Strategy**:
  - Redis cache for user sessions (TTL: 15 minutes)
  - CDN for processed videos
  - Application cache for metadata
- **Query Optimization**:
  - Use EXPLAIN ANALYZE for slow queries
  - Implement query timeouts (30 seconds max)
  - Use materialized views for complex reports

viii. Monitoring & Logging

Application Monitoring

- **Health Checks**: HTTP, TCP, and gRPC health endpoints

- **Metrics Collection**:
  - System: CPU, memory, disk I/O, network
  - Application: Request rate, error rate, latency
  - Business: Uploads per hour, transcode success rate
- **Alerting Rules**:
  - Critical: Service down, database unavailable
  - Warning: High error rate (>5%), high latency (>1s)
  - Info: Storage >80%, queue length >1000
- **Tools**: Prometheus for metrics, Grafana for visualization

Logging Standards

- **Log Levels**: DEBUG, INFO, WARN, ERROR, CRITICAL
- **Structured Logging**: JSON format with consistent schema

```json
{
  "timestamp": "2024-01-15T10:30:00Z",
  "level": "INFO",
  "service": "transcoder-api",
  "trace_id": "abc-123-def-456",
  "span_id": "span-789",
  "user_id": "user-123",
  "request_id": "req-456",
  "endpoint": "/api/videos/upload",
  "duration_ms": 245,
  "message": "Video upload completed successfully"
}
```

- **Log Retention**:
  - Application logs: 30 days in Elasticsearch
  - Access logs: 90 days compressed
  - Audit logs: 1 year immutable storage
- **Distributed Tracing**: OpenTelemetry for cross-service tracing
- **Error Tracking**: Sentry for real-time error aggregation

ix. Testing Requirements

Unit Testing

- **Coverage Target**: 80% code coverage minimum
- **Framework**: pytest for Python, Jest for JavaScript
- **Mocking**: unittest.mock for external dependencies
- **Test Data**: Factory Boy for test data generation
- **Focus Areas**:
    - Business logic validation
    - Utility functions
    - Model serialization/deserialization
    - Security functions

Integration Testing

- **Test Environment**: Docker Compose with all services
- **Database**: Testcontainers for PostgreSQL
- **Message Queue**: Embedded RabbitMQ for testing
- **Storage**: Local MinIO instance
- **Focus Areas**:
    - API endpoint integration
    - File upload/download workflows
    - Transcoding pipeline integration
    - Authentication flows

Performance Testing

- **Load Testing**: Locust for simulating user load
- **Stress Testing**: Determine breaking points
- **Endurance Testing**: 24-hour sustained load
- **Scalability Testing**: Horizontal scaling validation
- **Metrics to Measure**:
    - Throughput (requests/second)
    - Response time percentiles (P50, P95, P99)
    - Error rate under load
    - Resource utilization (CPU, memory, I/O)

E2E Testing

- **Framework**: Cypress with TypeScript
- **Coverage**:
  - User registration and login
  - Video upload and processing
  - Subscription management
  - Error scenarios
- **Cross-browser Testing**: Chrome, Firefox, Safari
- **Mobile Testing**: Responsive design validation
- **Accessibility Testing**: WCAG 2.1 AA compliance

x. Deployment & CI/CD

Deployment Strategy

- **Environment Strategy**:
  - Development: Feature branches, automatic deployment
  - Staging: Pre-production with production-like configuration
  - Production: Blue-green deployment with canary releases
- **Infrastructure as Code**: Terraform for cloud resources
- **Configuration Management**: Ansible for server configuration
- **Secrets Management**: HashiCorp Vault with dynamic secrets

Database Migrations

- **Migration Tool**: Alembic for SQLAlchemy
- **Version Control**: All migrations in source control
- **Rollback Strategy**: Every migration has a downgrade path
- **Zero-downtime Migrations**:
  - Add columns as nullable first
  - Backfill data in background
  - Add constraints after backfill
  - Use triggers for complex migrations
- **Migration Validation**: Automated schema validation in CI

Feature Flags

- **Framework**: Unleash or Flagsmith
- **Flag Types**:
  - Release flags: Gradual feature rollouts
  - Operational flags: Kill switches for problematic features
  - Permission flags: Feature access control
  - Experiment flags: A/B testing
- **Flag Lifecycle**:
  - Created with default false
  - Enabled for internal testing
  - Gradually rolled out to users
  - Monitored for performance impact
  - Removed after full rollout

CI/CD Pipeline

```yaml
# GitHub Actions Pipeline
name: CI/CD Pipeline

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    services:
      postgres: ...
      redis: ...
      rabbitmq: ...
    steps:
      - Run unit tests
      - Run integration tests
      - Run security scans (SAST, SCA)

  build:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - Build Docker images
      - Push to registry with version tags
      - Generate deployment manifests

  deploy-staging:
    needs: build
    if: github.ref == 'refs/heads/develop'
    runs-on: ubuntu-latest
    steps:
      - Deploy to staging
      - Run smoke tests
      - Run performance tests

  deploy-production:
    needs: [build, deploy-staging]
    if: github.ref == 'refs/heads/main'
    runs-on: ubuntu-latest
    steps:
      - Manual approval gate
      - Blue-green deployment
      - Run canary analysis
      - Update feature flags
```

xi. Edge Cases & Error Handling

Domain-Related Edge Cases

1. **Large File Uploads (>10GB)**:
   - Implement chunked upload with resume capability
   - Server-side validation of file type and size
   - Timeout handling with exponential backoff
2. **Unsupported Video Formats**:
   - Pre-upload format validation
   - Graceful error messages with supported formats
   - Option to suggest conversion tools
3. **Transcoding Failures**:
   - Retry with different parameters (max 3 attempts)
   - Fallback to lower quality preset
   - Detailed error logging with FFmpeg output
   - User notification with troubleshooting steps
4. **Storage Quota Exceeded**:
   - Pre-check before accepting upload
   - Automatic cleanup of oldest processed videos
   - Upgrade prompts with storage options
5. **Network Interruptions**:
   - Resumable uploads with checksum validation
   - Queue persistence for transcoding jobs
   - Automatic reconnection for WebSocket

Data Synchronization Edge Cases

1. **Conflicting Updates**:
   - Use optimistic locking with version numbers
   - Last-write-wins with conflict resolution UI

    - Audit trail of all changes
2. **Offline Data Conflicts**:
    - Detect conflicts on sync
    - Present merge options to user
    - Preserve both versions if needed
3. **Partial Sync Failures**:
    - Atomic operations where possible
    - Compensating transactions for rollback
    - Manual resolution interface
4. **Clock Skew Issues**:
    - Use server timestamps for ordering
    - NTP synchronization across all servers
    - Logical clocks for causal ordering

xii. Future Considerations

Planned Features

1. **Q1 2024**:
    - Live streaming capabilities
    - Video editing suite (trim, merge, add subtitles)
    - Advanced analytics dashboard
2. **Q2 2024**:
    - AI-powered content moderation
    - Automated video chapter generation
    - Custom watermarking
3. **Q3 2024**:
    - Collaborative video projects
    - Real-time video collaboration tools
    - Advanced compression algorithms

Scalability Roadmap

1. **Phase 1 (Current)**:
    - Single region deployment
    - Basic load balancing
    - Manual scaling
2. **Phase 2 (Q2 2024)**:

- Multi-region deployment
- Auto-scaling groups
- Global CDN integration

3. **Phase 3 (Q4 2024)**:
    - Edge computing for transcoding
    - Serverless functions for lightweight tasks
    - Predictive scaling based on usage patterns

Technology Evolution

1. **Container Orchestration**: Migrate to Kubernetes for better resource management
2. **Service Mesh**: Implement Istio for advanced traffic management
3. **Machine Learning**: Integrate ML for smart transcoding presets
4. **WebAssembly**: Explore WASM for client-side preprocessing

xiii. Appendix

A. API Rate Limits

Endpoint Category	Rate Limit	Burst	Scope
Authentication	10/minute	20	IP
Video Upload	5/minute	10	User
Video List	60/minute	100	User
Transcoding	10/minute	20	User
Admin Endpoints	1000/minute	5000	User

B. Supported Video Formats

Format	Codec	Max Resolution	Notes
MP4	H.264, H.265	8K	Recommended format
WebM	VP9, AV1	4K	Web optimized
MOV	ProRes, H.264	8K	Professional editing

Format	Codec	Max Resolution	Notes
AVI	Various	1080p	Legacy support
MKV	Various	8K	Container format

C. Transcoding Presets

Preset	Resolution	Bitrate	Use Case
Mobile	480p	800 kbps	Smartphone viewing
Standard	720p	2500 kbps	Web streaming
HD	1080p	5000 kbps	HD playback
4K	2160p	25000 kbps	Ultra HD
Archive	Original	Lossless	Long-term storage

D. Error Codes

Code	Description	Action
1001	Invalid video format	Convert to supported format
1002	File too large	Upgrade plan or reduce quality
1003	Transcoding timeout	Retry with lower quality
1004	Insufficient storage	Free up space or upgrade
1005	Corrupted video file	Re-upload or repair file
2001	Rate limit exceeded	Wait and retry
2002	Authentication failed	Refresh token or re-login
3001	Payment required	Update payment method
4001	System maintenance	Check status page

E. Glossary

- **Transcoding**: Converting video from one format to another

- **Transmuxing**: Changing container format without re-encoding
- **Codec**: Compression/decompression algorithm
- **Container**: File format that holds video, audio, metadata
- **Bitrate**: Amount of data processed per second
- **Preset**: Predefined settings for quality/speed balance

F. Contact Information

- **Technical Support**: support@videotranscode.com
- **Security Issues**: security@videotranscode.com
- **API Support**: api-support@videotranscode.com
- **Status Page**: status.videotranscode.com
- **Documentation**: docs.videotranscode.com

G. Revision History

Version	Date	Author	Changes
1.0	2024-01-15	Technical Team	Initial release
1.1	2024-01-20	Security Team	Added compliance details
1.2	2024-01-25	DevOps Team	Updated deployment strategy

Document Control: This document is maintained by the Architecture Review Board. Last reviewed: 2024-01-15. Next review scheduled: 2024-04-15.