Late Content Poster - Complete Documentation

Table of Contents

1. Overview

What is Late Content Poster?

Late Content Poster is a comprehensive social media management platform that enables users to schedule, automate, and manage content across multiple social media platforms. The application integrates with Dropbox for media storage, uses AI for content generation, and supports both manual and automated posting workflows.

Key Capabilities

- **Multi-Platform Support**: Instagram, Facebook, LinkedIn, Twitter, Threads, TikTok, Bluesky, YouTube
- **AI Content Generation**: Automated caption and hashtag generation using Abacus AI
- **Dropbox Integration**: Direct media sourcing from Dropbox folders
- **Automated Posting Series**: Schedule recurring posts from Dropbox folders
- **Bulk Scheduling**: Schedule multiple posts at once via CSV workflow
- **Template System**: Create and reuse custom graphic templates
- **Multi-Company Management**: Switch between different business accounts
- **Rate Limit Tracking**: Real-time monitoring of platform posting limits

Supported Platforms

Platform	Features	Media Types	Rate Limits
Instagram	Posts, Stories	Image, Video	8/day via Late API
Facebook	Posts	Image, Video	8/day via Late API
LinkedIn	Posts	Image, Video	8/day via Late API
Twitter	Tweets	Image, Video	17/day (separate API)
Threads	Posts	Image, Video	8/day via Late API
TikTok	Videos	Video only	8/day via Late API
Bluesky	Posts	Image, Video	8/day via Late API
YouTube	Videos	Video only	8/day via Late API

2. Architecture

System Components

```
┌─────────────────────────────────────────────────────────────┐
│                     Frontend (Next.js)                       │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐     │
│  │  Dashboard   │  │  Bulk CSV UI │  │  Templates   │     │
│  └──────────────┘  └──────────────┘  └──────────────┘     │
└─────────────────────────────────────────────────────────────┘
                            ↓
┌─────────────────────────────────────────────────────────────┐
│                   API Layer (Next.js API)                    │
│  ┌──────────┐  ┌────────────┐  ┌──────────┐  ┌──────────┐  │
│  │  Posts   │  │   Series   │  │ Profiles │  │Templates │  │
│  └──────────┘  └────────────┘  └──────────┘  └──────────┘  │
└─────────────────────────────────────────────────────────────┘
                            ↓
┌─────────────────────────────────────────────────────────────┐
│                    Business Logic Layer                      │
│  ┌──────────────────────┐  ┌─────────────────────────┐      │
│  │ Series Processor     │  │ Content Generator       │      │
│  │ (Dropbox Auto-Post)  │  │ (AI Integration)        │      │
│  └──────────────────────┘  └─────────────────────────┘      │
└─────────────────────────────────────────────────────────────┘
                            ↓
┌─────────────────────────────────────────────────────────────┐
│                  External Integrations                       │
│  ┌─────────┐  ┌──────────┐  ┌─────────┐  ┌──────────┐     │
│  │ Late API│  │ Dropbox  │  │ Twitter │  │Abacus AI │     │
│  └─────────┘  └──────────┘  └─────────┘  └──────────┘     │
└─────────────────────────────────────────────────────────────┘
                            ↓
┌─────────────────────────────────────────────────────────────┐
│                   Data & Storage Layer                       │
│  ┌─────────────┐  ┌─────────────┐  ┌─────────────┐        │
│  │ PostgreSQL  │  │   AWS S3    │  │  Temp Files │        │
│  │ (Prisma)    │  │  (Media)    │  │(Rate Limits)│        │
│  └─────────────┘  └─────────────┘  └─────────────┘        │
└─────────────────────────────────────────────────────────────┘
```

Technology Stack

Frontend
- Next.js 14.2.28
- React 18.2.0
- TypeScript 5.2.2
- Tailwind CSS 3.3.3
- Radix UI Components

Backend
- Next.js API Routes
- NextAuth.js 4.24.11 (Authentication)
- Prisma 6.7.0 (ORM)
- PostgreSQL (Database)

External Services
- Late API (Multi-platform posting)

- Dropbox API (Media storage)
- Twitter API v2 (Direct Twitter posting)
- Abacus AI (Content generation)
- AWS S3 (Temporary media storage)

Media Processing
- FFmpeg (Video compression)
- Sharp (Image compression)
- Tesseract.js (OCR for video text extraction)

---

3. Features

3.1 Content Journey (Manual Posting)

The Content Journey is a 6-step wizard for creating and posting content:

Step 1: Choose Template
- Select from existing templates or skip
- Preview template before use

Step 2: Upload or Select Media
- Upload from device
- Select from Dropbox
- Generate with DALL-E (optional)

Step 3: Analyze Images
- AI vision analysis of uploaded images
- Extract key elements for content generation

Step 4: Generate Content
- AI-generated captions and hashtags
- Customizable prompts
- Platform-specific optimization

Step 5: Select Profile & Platforms
- Choose business profile (Basketball Factory, Rise as One, etc.)
- Select target platforms (Instagram, Facebook, LinkedIn, etc.)
- Platform requirements validation

Step 6: Schedule & Review
- Immediate posting or scheduling
- Timezone-aware scheduling
- Final preview before posting

3.2 Dropbox Auto-Posting Series

Automated posting from Dropbox folders with intelligent scheduling:

Configuration
- Series name and description
- Dropbox folder selection
- AI prompt for content generation
- Target platforms and profile

- Schedule settings (days, time, timezone)
- Loop and auto-delete options

How It Works
1. User creates a series and selects Dropbox folder
2. System lists all media files in folder
3. First post is pre-scheduled immediately in Late API
4. Daemon monitors post status (every 5 minutes)
5. When current post publishes, next post is scheduled automatically
6. Process repeats until all files are posted

7-Layer Protection System
1. **Next Schedule Time Calculation**: Automatic timezone-aware scheduling
2. **Strict Time Validation**: Only processes when scheduled time arrives
3. **Duplicate Prevention Lock**: 50-minute cooldown between runs
4. **Atomic Processing Flag**: Prevents concurrent processing
5. **Rate Limit Pre-Check**: Verifies posting capacity before processing
6. **Platform Availability Pre-Check**: Validates Dropbox and Late API connections
7. **Strict Time Window Enforcement**: Logs timing drift and warnings

3.3 Bulk CSV Scheduling

Schedule multiple posts from a Dropbox folder in one workflow:

5-Step Workflow

Step 1: Select Media from Dropbox
- Browse and select Dropbox folder
- Preview available media files
- File count display

Step 2: Generate AI Content
- Provide AI prompt for all files
- Batch AI content generation
- Real-time progress tracking
- Preview generated content for each file
- Video OCR text extraction

Step 3: Select Profile & Platforms
- Choose business profile
- Select target platforms

Step 4: Configure Scheduling
- Set start date and time
- Choose days of week for posting
- Select timezone

Step 5: Results
- View success/failure counts
- Detailed results per post
- Error messages for failed posts

Batched Processing
- Processes in batches of 10 posts

- 5-second delay between posts
- 10-second delay between batches
- Prevents API rate limit errors
- Real-time progress updates

3.4 Template System

Create and manage reusable graphic templates:

Template Creation
- Upload background image
- Add text fields with customization:
- Font family and size
- Text color
- Alignment (left, center, right)
- Positioning (x, y coordinates)
- Add image overlays
- Preview in real-time

Template Usage
- Select template in Content Journey
- Fill in dynamic fields
- Generate final graphic
- Post to social media

Template Library
- User templates (private)
- Public templates (shared)
- Category filtering
- Search functionality

3.5 Company Management

Multi-company support with isolated workspaces:

Features
- Create unlimited companies
- Switch between companies
- Each company has:
- Separate profiles
- Independent platform connections
- Isolated templates
- Separate scheduled posts
- Custom brand colors
- Independent prompt library

Company Switcher
- Dropdown in sidebar
- Search companies
- Create new companies
- Role-based access (Owner, Admin, Member)

3.6 Rate Limit Tracking

Late API Rate Limits
- 8 posts/day per platform per profile
- Rolling 24-hour window
- Real-time tracking and display
- Status levels: Good, Warning, Critical
- Reset time display

Twitter Rate Limits
- 17 tweets/day
- Separate tracking from Late API
- Reset time display
- Warning banners

UI Components
- Dashboard banners (collapsible)
- Detailed per-platform breakdown
- Color-coded status indicators
- Reset countdown timers

3.7 AI Content Generation

AI Vision Analysis
- Analyzes images for objects, text, themes
- Extracts context for content generation
- Supports multiple image types

Video OCR
- Extracts text from video frames using Tesseract.js
- Multi-frame extraction (25%, 50%, 75%)
- Selects best frame with most text
- Generates content based on extracted quotes

Content Generation
- AI-powered captions and hashtags
- Platform-specific optimization
- Character limit enforcement (e.g., Threads 500 chars)
- Plain text output (no markdown)
- Unique content for each post

Models Used
- `gpt-4o-mini`: Primary model for content generation
- `gpt-4o`: Used for complex vision analysis

---

4. Setup & Installation

4.1 Prerequisites

- Node.js 22.14.0 or higher
- PostgreSQL database
- Dropbox account and App credentials

- Late API account and API key
- Twitter Developer account (for Twitter posting)
- Abacus AI account and API key

4.2 Environment Variables

Create a `.env` file in the `nextjs_space` directory:

```
# Database
DATABASE_URL="postgresql://user:password@host:port/database"

# NextAuth
NEXTAUTH_SECRET="your-nextauth-secret"
NEXTAUTH_URL="http://localhost:3000"

# Dropbox
DROPBOX_ACCESS_TOKEN="your-dropbox-access-token"
DROPBOX_APP_KEY="your-dropbox-app-key"
DROPBOX_APP_SECRET="your-dropbox-app-secret"
DROPBOX_REFRESH_TOKEN="your-dropbox-refresh-token"

# Late API
LATE_API_KEY="your-late-api-key"

# Abacus AI
ABACUSAI_API_KEY="your-abacus-ai-api-key"

# AWS S3 (for media storage)
AWS_BUCKET_NAME="your-s3-bucket-name"
AWS_FOLDER_PREFIX="your-folder-prefix/"
AWS_REGION="us-west-2"
```

4.3 Installation Steps

```
# 1. Navigate to project directory
cd /home/ubuntu/late_content_poster/nextjs_space

# 2. Install dependencies
yarn install

# 3. Generate Prisma client
yarn prisma generate

# 4. Run database migrations
yarn prisma migrate deploy

# 5. Seed the database (creates test user)
yarn prisma db seed

# 6. Build the application
yarn build

# 7. Start the production server
yarn start
```

4.4 Initial Configuration

1. Sign In
- Default test user: `john@doe.com` / `password123`
- Admin user: `admin@example.com` / `admin123`

2. Configure Dropbox
- Obtain Dropbox App Key and Secret
- Generate OAuth refresh token
- Add credentials to `.env`

3. Configure Late API
- Create Late API account at https://getlate.dev
- Generate API key
- Add to `.env`
- Connect social media accounts in Late dashboard

4. Configure Twitter (Optional)
- Create Twitter Developer app
- Generate API keys and access tokens
- Store in `abacusai_auth_secrets.json`

5. Configure Abacus AI
- Create Abacus AI account
- Generate API key
- Add to `.env`

6. Set Up Daemon (Optional)
- Configure cron job to run every 5 minutes:
`bash`
`  */5 * * * * cd /home/ubuntu/late_content_poster/nextjs_space && npx tsx scripts/process_scheduled_series.ts >> /tmp/series-processor.log 2>&1`

7. Configure Webhooks (Optional)
- In Late API dashboard, add webhook URL:
`https://your-domain.com/api/webhooks/late`
- Subscribe to events: `post.published`, `post.failed`, `post.draft`, `post.deleted`

---

5. User Guide

5.1 Creating Your First Post

1. **Navigate to Dashboard**
   - Click “Content Journey” in sidebar

2. **Select Template (Optional)**
   - Choose a template or click “Skip”

3. **Upload Media**
   - Click “Upload” or “Select from Dropbox”
   - Choose image or video

4. **Analyze Images**
   - Review AI analysis results
   - Click “Next”

5. **Generate Content**
   - Review AI-generated caption and hashtags
   - Edit if needed
   - Click “Next”

6. **Select Profile & Platforms**
   - Choose business profile (e.g., “Basketball Factory”)
   - Select platforms (Instagram, Facebook, etc.)
   - Click “Next”

7. **Schedule or Post**
   - **Immediate**: Click “Post Now”
   - **Scheduled**: Select date/time, click “Schedule”

5.2 Creating a Dropbox Auto-Posting Series

1. **Navigate to Post Series**
   - Click “Post” in sidebar
   - Click “+ New Series”

2. **Basic Information**
   - Enter series name (e.g., “Daily Motivational Quotes”)
   - Add description (optional)

3. **Select Dropbox Folder**
   - Click “Select Dropbox Folder”
   - Navigate to folder with media files
   - Click “Select Current Folder”

4. **Configure AI Prompt**
   - Enter instructions for AI content generation
   - Example: “Generate motivational captions based on the quote in the image”

5. **Select Profile & Platforms**
   - Choose profile
   - Select platforms (Instagram, Facebook, LinkedIn, etc.)

6. **Configure Schedule**
   - **Start Date**: When to begin posting
   - **Days of Week**: Which days to post (e.g., Mon-Fri)
   - **Time**: Post time (e.g., 7:00 AM)
   - **Timezone**: Your timezone (e.g., America/New_York)

7. **Advanced Options**
   - **Loop**: Restart from first file after last
   - **Auto-Post**: Enable automatic posting
   - **Delete After Posting**: Remove from Dropbox after posting

8. **Create Series**
   - Click "Create Series"
   - First post is immediately scheduled in Late API

9. **Monitor Progress**
   - Check "Scheduled Posts" in Late API dashboard
   - View daemon logs for processing status

5.3 Bulk CSV Scheduling

1. **Navigate to Bulk CSV**
   - Click "Bulk Schedule CSV" in sidebar

2. **Step 1: Select Media**
   - Click "Select Dropbox Folder"
   - Choose folder with media files
   - Review file count

3. **Step 2: Generate AI Content**
   - Enter AI prompt for content generation
   - Click "Generate AI Content"
   - Wait for progress bar to complete
   - Review generated content for each file
   - Regenerate if needed

4. **Step 3: Select Profile & Platforms**
   - Choose business profile
   - Select platforms

5. **Step 4: Configure Scheduling**
   - Set start date and time
   - Choose days of week
   - Select timezone

6. **Step 5: Complete & Schedule**
   - Click "Complete & Schedule"
   - Wait for batched processing
   - Review results

5.4 Managing Companies

Create a New Company
1. Click company dropdown (top-left)
2. Click "Create Company"
3. Enter company name and description
4. Click "Create Company"

Switch Companies
1. Click company dropdown
2. Select desired company
3. Dashboard refreshes with company data

Configure Company Branding
1. Navigate to Settings
2. Click "Branding" tab

3. Set primary, secondary, accent colors
4. Upload logo (optional)
5. Click “Save”

5.5 Monitoring Rate Limits

Dashboard Banners
- **Late API Banner**: Shows rate limit status for all platforms
- **Twitter Banner**: Shows Twitter-specific rate limits
- Banners appear only when approaching or at limits

Detailed View
- Click “Expand” on banner to see per-platform breakdown
- View remaining posts for each platform
- See reset times for each platform

Planning Posts
- Use different profiles to increase limits
- Schedule posts across multiple days
- Monitor rolling 24-hour window

---

6. API Documentation

6.1 Authentication

All API routes require authentication via NextAuth session:

```
const session = await getServerSession(authOptions)
if (!session?.user?.email) {
  return NextResponse.json({ error: 'Unauthorized' }, { status: 401 })
}
```

6.2 Core Endpoints

Posts

GET /api/posts
- Fetch user’s posts
- Query params: `status`, `limit`, `offset`
- Returns: Array of post objects with analytics

POST /api/posts
- Create new post
- Body: `content`, `platforms`, `mediaUrls`, `profileId`, `scheduledAt`, `timezone`
- Returns: Created post object

POST /api/late/post
- Post to Late API (multi-platform)
- Body: `text`, `platforms`, `mediaUrls`, `profileId`, `scheduledFor`, `timezone`
- Returns: `success`, `latePostId`, `twitterResult`

POST /api/twitter/post
- Post to Twitter directly

- Body: `text`, `mediaUrls`
- Returns: `success`, `tweetId`

Series

GET /api/series
- Fetch user's post series
- Returns: Array of series objects

POST /api/series
- Create new series
- Body: `name`, `description`, `dropboxFolderId`, `dropboxFolderPath`, `prompt`, `profileId`, `platforms`, `daysOfWeek`, `timeOfDay`, `startDate`, `timezone`, `autoPost`, `loopEnabled`
- Returns: Created series object

PATCH /api/series/[id]
- Update existing series
- Body: Same as POST
- Returns: Updated series object

DELETE /api/series/[id]
- Delete series
- Returns: Success message

POST /api/series/process
- Process scheduled series (daemon endpoint)
- Returns: Array of processing results

POST /api/series/[id]/bulk-schedule
- Bulk schedule all files in series' Dropbox folder
- Returns: Streaming progress updates

Dropbox

GET /api/dropbox/folders
- List Dropbox folders
- Query params: `parentPath`
- Returns: Array of folders with file counts

GET /api/dropbox/files
- List files in Dropbox folder
- Query params: `path`
- Returns: Array of media files

GET /api/dropbox/download
- Download file from Dropbox
- Query params: `path`
- Returns: File buffer

AI Content Generation

POST /api/generate-content
- Generate AI content
- Body: `contentType`, `platform`, `topic`, `tone`, `mediaUrls`
- Returns: Streaming AI-generated content

POST /api/bulk-csv/analyze-file
- Analyze media file and generate content
- Body: `filePath`, `aiPrompt`, `platforms`
- Returns: Generated content

POST /api/bulk-csv/generate
- Generate bulk CSV and schedule posts
- Body: `dropboxPath`, `profileId`, `platforms`, `scheduleConfig`, `generatedContent`
- Returns: Success message and Late API response

Templates

GET /api/templates
- Fetch user's templates
- Returns: Array of template objects

POST /api/templates
- Create new template
- Body: `name`, `category`, `imageUrl`, `fields`
- Returns: Created template object

GET /api/templates/[id]
- Fetch specific template
- Returns: Template object with `isOwner` flag

PUT /api/templates/[id]
- Update template
- Body: `name`, `category`, `imageUrl`, `fields`
- Returns: Updated template object

DELETE /api/templates/[id]
- Delete template
- Returns: Success message

Companies

GET /api/companies
- Fetch user's companies
- Returns: Array of companies with `selectedCompanyId`

POST /api/companies
- Create new company
- Body: `name`, `description`
- Returns: Created company object

POST /api/companies/switch
- Switch active company
- Body: `companyId`
- Returns: Success message

Rate Limits

GET /api/late/rate-limit
- Get Late API rate limit status
- Returns: Per-profile platform statuses with reset times

GET /api/twitter/rate-limit
- Get Twitter rate limit status
- Returns: Remaining tweets, reset time, status level

Webhooks

POST /api/webhooks/late
- Receive Late API webhook notifications
- Body: `event`, `post` (from Late API)
- Action: Triggers next post in series
- Returns: Success message

6.3 Response Formats

Success Response

```
{
  "success": true,
  "data": { /* response data */ },
  "message": "Operation completed successfully"
}
```

Error Response

```
{
  "error": "Error message",
  "status": 400,
  "details": { /* additional error details */ }
}
```

7. Technical Implementation

7.1 Database Schema

Key Models

```
model User {
  id                String    @id @default(cuid())
  email             String    @unique
  name              String?
  password          String
  selectedCompanyId String?
  companies         CompanyMember[]
  createdAt         DateTime  @default(now())
}

model Company {
  id          String          @id @default(cuid())
  name        String
  description String?
  logoUrl     String?
  ownerId     String
  isActive    Boolean         @default(true)
  members     CompanyMember[]
  profiles    Profile[]
  posts       Post[]
  templates   Template[]
  series      PostSeries[]
  createdAt   DateTime        @default(now())
}

model Profile {
  id               String            @id @default(cuid())
  name             String
  companyId        String
  company          Company           @relation(fields: [companyId], references: [id], o
nDelete: Cascade)
  lateProfileId    String?           // Late API profile ID
  platformSettings PlatformSetting[]
  series           PostSeries[]
  createdAt        DateTime          @default(now())
}

model PlatformSetting {
  id          String   @id @default(cuid())
  profileId   String
  companyId   String
  platform    String   // instagram, facebook, linkedin, twitter, etc.
  isConnected Boolean  @default(false)
  platformId  String?  // Account ID in Late API
  profile     Profile  @relation(fields: [profileId], references: [id], onDelete: Cas-
cade)
  createdAt   DateTime @default(now())
}

model PostSeries {
  id                 String    @id @default(cuid())
  name               String
  description        String?
  companyId          String
  profileId          String
  dropboxFolderId    String?
  dropboxFolderPath  String?
  prompt             String?   // AI prompt for content generation
  daysOfWeek         String[]  // ["MONDAY", "WEDNESDAY", "FRIDAY"]
  timeOfDay          String    // "07:00"
  startDate          DateTime
  timezone           String    @default("America/New_York")
  currentFileIndex   Int        @default(1)
  nextScheduledAt    DateTime?
  lastProcessedAt    DateTime?
  currentLatePostId  String?    // ID of current post in Late API
  status             String     @default("ACTIVE") // ACTIVE, PAUSED, COMPLETED
  autoPost           Boolean    @default(true)
  loopEnabled        Boolean    @default(false)
  deleteAfterPosting Boolean    @default(false)
  isProcessing       Boolean    @default(false) // Atomic lock
  company            Company    @relation(fields: [companyId], references: [id], onDele
te: Cascade)
  profile            Profile    @relation(fields: [profileId], references: [id], onDele
te: Cascade)
  createdAt          DateTime   @default(now())
}

model Post {
  id          String    @id @default(cuid())
  userId      String
  companyId   String
  content     String
  platforms   String[]  // ["instagram", "facebook", "linkedin"]
  mediaUrls   String[]
  status      String    @default("draft") // draft, scheduled, published, failed
  scheduledAt DateTime?
  postedAt    DateTime?
  company     Company   @relation(fields: [companyId], references: [id], onDelete: Cas
cade)
  createdAt   DateTime  @default(now())
}

model Template {
  id          String   @id @default(cuid())
  userId      String
  companyId   String
  name        String
  category    String
  imageUrl    String
  fields      Json     // Array of field definitions
  isPublic    Boolean  @default(false)
  company     Company  @relation(fields: [companyId], references: [id], onDelete: Cas-
cade)
  createdAt   DateTime @default(now())
}
```

7.2 Dropbox Auto-Posting Flow

```
┌─────────────────────────────────────────────────────────────┐
│  1. USER CREATES SERIES                                     │
│     - Selects Dropbox folder                                │
│     - Configures AI prompt, platforms, schedule             │
│     - Clicks "Create Series"                                │
└─────────────────────────────────────────────────────────────┘
                              ↓
┌─────────────────────────────────────────────────────────────┐
│  2. IMMEDIATE PRE-SCHEDULING (scheduleFirstSeriesPost)      │
│     - Downloads first file from Dropbox                     │
│     - Generates AI content                                  │
│     - Uploads media to Late API                             │
│     - Creates scheduled post in Late API                    │
│     - Stores currentLatePostId in database                  │
└─────────────────────────────────────────────────────────────┘
                              ↓
┌─────────────────────────────────────────────────────────────┐
│  3. DAEMON MONITORING (every 5 minutes)                     │
│     - Runs /api/series/process                              │
│     - Checks series with nextScheduledAt <= now             │
│     - Validates 7-layer protection system                   │
└─────────────────────────────────────────────────────────────┘
                              ↓
┌─────────────────────────────────────────────────────────────┐
│  4. STATUS CHECK (checkLatePostStatus)                      │
│     - Queries Late API for current post status              │
│     - If status = "scheduled": Wait (do nothing)            │
│     - If status = "published"/"failed"/etc.: Schedule next  │
└─────────────────────────────────────────────────────────────┘
                              ↓
┌─────────────────────────────────────────────────────────────┐
│  5. SCHEDULE NEXT POST (processCloudStorageSeries)          │
│     - Increments currentFileIndex                           │
│     - Downloads next file from Dropbox                      │
│     - Generates AI content                                  │
│     - Uploads media to Late API                             │
│     - Calculates next scheduled date/time                   │
│     - Creates scheduled post in Late API                    │
│     - Updates currentLatePostId and nextScheduledAt         │
└─────────────────────────────────────────────────────────────┘
                              ↓
┌─────────────────────────────────────────────────────────────┐
│  6. WEBHOOK (OPTIONAL - Instant Trigger)                    │
│     - Late API sends webhook when post publishes            │
│     - POST /api/webhooks/late                               │
│     - Immediately triggers processCloudStorageSeries        │
│     - Bypasses 5-minute daemon wait                         │
└─────────────────────────────────────────────────────────────┘
                              ↓
                    (Repeat Steps 4-6)
```

7.3 Media Compression

Image Compression

- Uses `sharp` library
- Platform-specific limits:
- Late API: 10MB
- Twitter: 5MB

- Instagram: 8MB
- Resizes to max dimensions
- Iteratively reduces JPEG quality (85 → 30)
- Maintains aspect ratio

Video Compression
- Uses `fluent-ffmpeg` with `@ffmpeg-installer/ffmpeg`
- Platform-specific limits:
- Late API: 10MB
- Twitter: 512MB
- Instagram: 100MB
- Reduces bitrate: 800k video, 128k audio
- Uses H.264 codec (libx264)
- Optimized flags: `-movflags +faststart`, `-preset medium`, `-crf 23`
- Maintains aspect ratio with padding

7.4 AI Integration

Vision Analysis

```
const response = await fetch('https://apps.abacus.ai/v1/chat/completions', {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${process.env.ABACUSAI_API_KEY}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    model: 'gpt-4o',
    messages: [
      {
        role: 'user',
        content: [
          { type: 'text', text: 'Analyze this image...' },
          { type: 'image_url', image_url: { url: imageUrl } }
        ]
      }
    ]
  })
})
```

Content Generation

```
const response = await fetch('https://apps.abacus.ai/v1/chat/completions', {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${process.env.ABACUSAI_API_KEY}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    model: 'gpt-4o-mini',
    messages: [
      {
        role: 'system',
        content: 'CRITICAL FORMATTING RULES: Output ONLY plain text...'
      },
      {
        role: 'user',
        content: `Generate social media content based on: ${imageAnalysis}`
      }
    ]
  })
})
```

Video OCR

```
import { createWorker } from 'tesseract.js'

const worker = await createWorker('eng')
const { data } = await worker.recognize(frameBuffer)
const extractedText = data.text.trim()
await worker.terminate()
```

7.5 Rate Limit Tracking

Storage
- File: `/tmp/late-rate-limit.json`
- Structure:
`json`
```
 {
   "profileId": {
     "instagram": [
       { "timestamp": 1700000000000, "postId": "abc123" },
       { "timestamp": 1700003600000, "postId": "def456" }
     ]
   }
 }
```

Calculation
- Rolling 24-hour window
- Count posts where `timestamp >= now - 24h`
- Reset time: `oldestPostTimestamp + 24h`
- Status levels:
- Good: 3+ remaining
- Warning: 1-2 remaining
- Critical: 0 remaining

Recording

```
import { recordLatePost } from '@/lib/late-rate-limit'

// After successful post
await recordLatePost(platform, profileId, profileName)
```

---

8. Troubleshooting

8.1 Common Issues

Issue: Dropbox folder picker fails

Symptoms
- Error: "Failed to load folders"
- 401 Unauthorized error

Causes
- Expired Dropbox access token
- Missing Dropbox app credentials

Solutions
1. Check `.env` for valid `DROPBOX_ACCESS_TOKEN`
2. Verify `DROPBOX_REFRESH_TOKEN` is set
3. Check `DROPBOX_APP_KEY` and `DROPBOX_APP_SECRET`
4. Reconnect Dropbox via OAuth if needed

Issue: Series not posting automatically

Symptoms
- Posts don't appear in Late API "Scheduled Posts"
- Daemon logs show no activity

Causes
- Daemon not running or running hourly instead of every 5 minutes
- Webhook not configured
- Series is paused or inactive

Solutions
1. Check daemon frequency:
`bash`
```
   crontab -l | grep series
```
2. Update to 5-minute frequency:
`bash`
```
   */5 * * * * cd /path/to/project && npx tsx scripts/process_scheduled_series.ts
```
3. Configure webhook in Late API dashboard
4. Check series status in database (should be "ACTIVE")

Issue: AI content generation fails

Symptoms
- Posts have no text or generic "Check out this post!" message
- Console shows "AI API failed"

Causes
- Missing `ABACUSAI_API_KEY`
- Invalid API key
- API rate limit exceeded

Solutions
1. Verify `ABACUSAI_API_KEY` in `.env`
2. Check API key validity
3. Review Abacus AI dashboard for quota

Issue: Rate limit banner not showing

Symptoms
- Dashboard doesn't show rate limit warnings
- All platforms show 0/8

Causes
- No recent posts recorded
- Rate limit file missing
- Profile not configured correctly

Solutions
1. Check `/tmp/late-rate-limit.json` exists
2. Verify profile has `lateProfileId` set
3. Post a test to record activity
4. Hard refresh browser (Ctrl+Shift+R)

Issue: Bulk CSV scheduling fails

Symptoms
- "405 Method Not Allowed" error
- Posts not appearing in Late API

Causes
- Trying to schedule posts for past dates
- Invalid platform configuration
- Missing Late API account IDs

Solutions
1. Ensure start date is in the future
2. Verify all platforms have `platformId` configured
3. Check Late API account connections

8.2 Debug Mode

Enable Verbose Logging

1. **Backend Logs**
   - Set `NODE_ENV=development` in `.env`
   - View logs: `tail -f /tmp/series-processor.log`

2. **Browser Console**
   - Open DevTools (F12)
   - Check Console tab for errors
   - Check Network tab for failed requests

3. **Database Queries**
   - Enable Prisma logging in `lib/db.ts`:
   `typescript`
   ```
     const prisma = new PrismaClient({
       log: ['query', 'info', 'warn', 'error']
     })
   ```

8.3 Performance Optimization

Slow API Responses
- Enable Redis caching for frequent queries
- Optimize Prisma queries with `include` and `select`
- Use database indexes on frequently queried fields

Large File Processing
- Increase Node.js memory limit:
`bash`
```
 NODE_OPTIONS="--max-old-space-size=4096" yarn start
```
- Use streaming for large file uploads
- Implement queue system for background processing

Database Performance
- Add indexes on foreign keys
- Use pagination for large datasets
- Archive old posts periodically

---

9. Deployment

9.1 Production Deployment

Deployed URL: https://late-content-poster-bvwoef.abacusai.app

Deployment Steps

1. **Build Application**
   `bash`
   ```
     cd /home/ubuntu/late_content_poster/nextjs_space
     yarn build
   ```

2. **Environment Variables**
   - Ensure all production environment variables are set
   - Update `NEXTAUTH_URL` to production domain

3. **Database Migration**
   `bash`
   ```
     yarn prisma migrate deploy
   ```

4. **Start Production Server**
   `bash`
   ```
     yarn start
   ```
   Or use PM2:
   `bash`

```
   pm2 start yarn --name "late-content-poster" -- start
   pm2 save
```

5. **Configure Reverse Proxy (Nginx)**
   ```nginx
   server {
   listen 80;
   server_name late-content-poster-bvwoef.abacusai.app;

   location / {
   proxy_pass http://localhost:3000;
   proxy_http_version 1.1;
   proxy_set_header Upgrade $http_upgrade;
   proxy_set_header Connection ‘upgrade’;
   proxy_set_header Host $host;
   proxy_cache_bypass $http_upgrade;
   }
   }
   ```

6. **Setup SSL Certificate (Let’s Encrypt)**
   `bash`
   ```
      sudo certbot --nginx -d late-content-poster-bvwoef.abacusai.app
   ```

7. **Configure Daemon**
   `bash`
   ```
      sudo crontab -e
   ```
   Add:
   `bash`
   ```
      */5 * * * * cd /home/ubuntu/late_content_poster/nextjs_space && npx tsx scripts/process_scheduled_series.ts >> /var/log/series-processor.log 2>&1
   ```

8. **Setup Webhooks**
   - In Late API dashboard, add webhook URL: `https://late-content-poster-bvwoef.abacusai.app/api/webhooks/late`
   - Subscribe to: `post.published`, `post.failed`, `post.draft`, `post.deleted`

9.2 Monitoring

Health Checks
- Endpoint: `GET /api/health`
- Returns: Application status, database connectivity

Log Monitoring

```
# Application logs
tail -f /var/log/late-content-poster.log

# Daemon logs
tail -f /var/log/series-processor.log

# Error logs
tail -f /var/log/late-content-poster-error.log
```

Metrics
- Posts created per day
- Series processing success rate
- API response times
- Error rates

9.3 Backup & Recovery

Database Backup

```
# Daily backup
pg_dump -h localhost -U postgres -d late_content_poster > backup_$(date +%Y%m%d).sql

# Automated backup script (cron)
0 2 * * * /path/to/backup.sh
```

Media Backup
- S3 bucket versioning enabled
- Lifecycle policy for archiving old media

Configuration Backup
- `.env` file (encrypted)
- Prisma schema
- Nginx configuration

---

10. Maintenance

10.1 Regular Tasks

Daily
- Monitor daemon logs for errors
- Check rate limit status
- Review failed posts

Weekly
- Review database size and performance
- Check for outdated dependencies
- Analyze posting metrics

Monthly
- Update dependencies
- Review and optimize queries
- Archive old posts
- Rotate logs

10.2 Dependency Updates

```
# Check for updates
yarn outdated

# Update dependencies
yarn upgrade-interactive --latest

# Test after updates
yarn test
yarn build
```

10.3 Database Maintenance

Vacuum PostgreSQL

```
VACUUM ANALYZE;
```

Archive Old Posts

```
-- Archive posts older than 6 months
INSERT INTO post_archive SELECT * FROM "Post" WHERE "createdAt" < NOW() - INTERVAL '6
months';
DELETE FROM "Post" WHERE "createdAt" < NOW() - INTERVAL '6 months';
```

Rebuild Indexes

```
REINDEX DATABASE late_content_poster;
```

10.4 Support

For issues or questions:
- **Documentation**: `/home/ubuntu/late_content_poster/COMPLETE_DOCUMENTATION.md`
- **Technical Issues**: Check troubleshooting section
- **Feature Requests**: Create GitHub issue (if applicable)

Appendix

A. Environment Variables Reference

Variable	Description	Required	Example
`DATABASE_URL`	PostgreSQL connection string	Yes	`postgresql://user:pass@localhost:5432/db`
`NEXTAUTH_SECRET`	NextAuth secret key	Yes	`your-secret-key-here`
`NEXTAUTH_URL`	Application URL	Yes	`https://your-domain.com`
`DROPBOX_ACCESS_TOKEN`	Dropbox access token	Yes	`sl.xxxxx`
`DROPBOX_APP_KEY`	Dropbox app key	Yes	`xxxxxxxxx`
`DROPBOX_APP_SECRET`	Dropbox app secret	Yes	`xxxxxxxxx`
`DROPBOX_REFRESH_TOKEN`	Dropbox refresh token	Yes	`xxxxxxxxx`
`LATE_API_KEY`	Late API key	Yes	`late_xxxxx`
`ABACUSAI_API_KEY`	Abacus AI API key	Yes	`xxxxx`
`AWS_BUCKET_NAME`	S3 bucket name	Yes	`your-bucket`
`AWS_FOLDER_PREFIX`	S3 folder prefix	No	`6788/`
`AWS_REGION`	AWS region	No	`us-west-2`

B. Prisma Commands

```bash
# Generate Prisma client
yarn prisma generate

# Create migration
yarn prisma migrate dev --name migration_name

# Deploy migrations
yarn prisma migrate deploy

# Reset database (WARNING: deletes all data)
yarn prisma migrate reset

# Seed database
yarn prisma db seed

# Open Prisma Studio
yarn prisma studio
```

C. Useful Scripts

```bash
# Test Dropbox connection
cd /home/ubuntu/late_content_poster/nextjs_space
npx tsx test_dropbox_now.ts

# Test series processing
npx tsx test_auto_scheduling.ts

# Check rate limits
npx tsx sync_late_rate_limits.ts

# Trigger daemon manually
npx tsx scripts/process_scheduled_series.ts

# Bulk schedule posts
npx tsx bulk_schedule_batched.ts
```

D. API Rate Limits

Service	Endpoint	Limit	Window
Late API	`/v1/posts`	8 posts/platform/profile	24 hours (rolling)
Twitter	`/2/tweets`	17 tweets	24 hours (rolling)
Abacus AI	`/v1/chat/completions`	1000 requests	1 minute
Dropbox	`/2/files/*`	300 requests	1 minute

E. Glossary

- **Company**: A business entity with multiple profiles

- **Profile**: A social media identity (e.g., “Basketball Factory”)
- **Platform**: A social media service (e.g., Instagram, Twitter)
- **Series**: An automated posting schedule from a Dropbox folder
- **Bulk CSV**: A workflow for scheduling multiple posts at once
- **Template**: A reusable graphic design
- **Content Journey**: A 6-step wizard for creating posts
- **Rate Limit**: Maximum posts allowed per platform per day
- **Rolling Window**: A time period that moves forward continuously (e.g., last 24 hours)
- **Daemon**: A background process that runs periodically
- **Webhook**: A real-time HTTP callback from an external service
- **OCR**: Optical Character Recognition (text extraction from images/videos)

Last Updated: November 26, 2025
Version: 1.0.0
Deployed URL: https://late-content-poster-bvwoef.abacusai.app