# ORBYT

Technical Requirements Specification
Social Media Publishing Platform

**Version 2.0**

**January 30, 2026**

*Prepared for:* **AMITY NET AS**

*Prepared by:* **STUDIO X AS**

CONFIDENTIAL DOCUMENT

**DOCUMENT INFORMATION**

| | |
|---|---|
| Document Title | Orbyt Technical Requirements Specification |
| Version | 2.0 |
| Date | January 30, 2026 |
| Client | Amity Net AS |
| Author | Studio X AS |
| Classification | Confidential |
| Language | English |

**REVISION HISTORY**

| Version | Date | Author | Changes |
|---|---|---|---|
| 1.0 | Jan 28, 2026 | Studio X | Initial draft |
| 2.0 | Jan 30, 2026 | Studio X | Added Mux, reduced platform scope |

# 1. Executive Summary

Orbyt is a centralized social media management platform designed for influencers, solopreneurs, and SMBs. The platform enables users to publish content across multiple social media platforms from a single interface, schedule posts, and manage their digital presence through a unified "Link-in-Bio" landing page.

## 1.1 MVP Scope

- Cross-platform content publishing (Meta & YouTube)
- Video processing and management via Mux
- Content scheduling and calendar management
- Link-in-Bio personal landing page
- AI-assisted content generation
- Basic analytics dashboard

## 1.2 Target Platforms

| Platform | Type | Scope |
|---|---|---|
| **Facebook** | Publishing | Posts, images, videos |
| **Instagram** | Publishing | Posts, reels, stories (phase 2) |
| **YouTube** | Publishing | Videos, shorts, thumbnails |

---

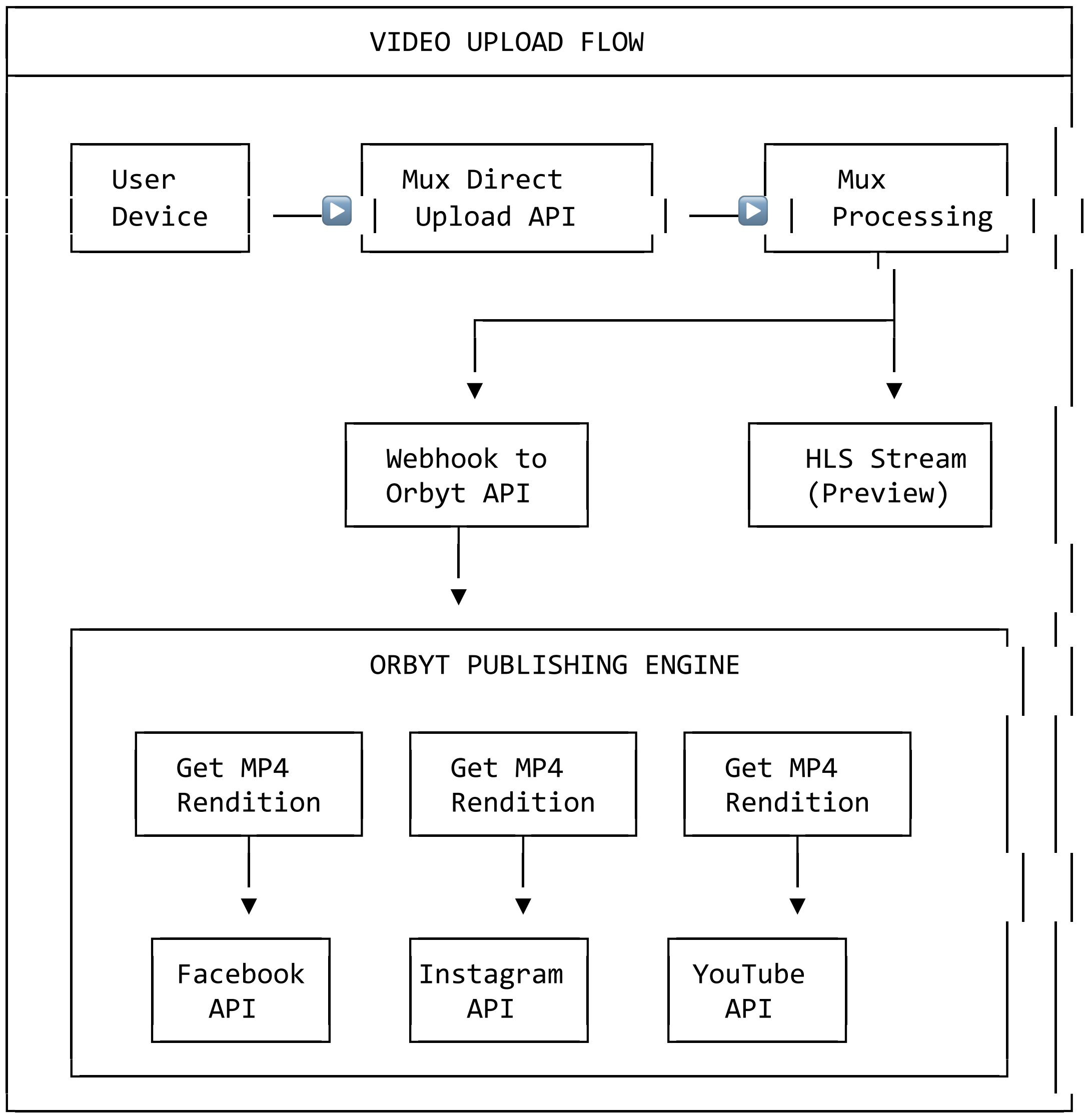
# 2. Video Infrastructure - Mux

## 2.1 Why Mux?

Mux provides enterprise-grade video infrastructure that eliminates the need to build custom video processing pipelines. For Orbyt, this means:

- **Instant transcoding** to platform-specific formats
- **Adaptive streaming** for in-app preview
- **Automatic thumbnails** for calendar views
- **Resumable uploads** for large video files
- **Analytics** on video engagement

## 2.2 Mux Architecture



## 2.3 Mux Integration Points

| Feature | Mux Capability | Orbyt Usage |
|---|---|---|
| **Upload** | Direct Upload API | Client uploads directly to Mux |
| **Processing** | Auto-transcoding | Multiple formats generated |
| **Preview** | HLS Playback | In-app video preview |
| **Thumbnails** | Image API | Calendar thumbnails, post previews |
| **Static Files** | MP4 Renditions | Download for platform upload |
| **Webhooks** | video.asset.ready | Trigger post-processing |

## 2.4 Mux API Integration

```
// Mux Service Interface
interface MuxVideoService {
  // Create direct upload URL for client
  createUpload(options: {
    corsOrigin: string;
    newAssetSettings: {
      playbackPolicy: 'public' | 'signed';
      mp4Support: 'standard' | 'capped-1080p';
    };
  }): Promise<{
    id: string;
    url: string;  // Client uploads here
  }>;

  // Get asset status and details
  getAsset(assetId: string): Promise<{
    id: string;
    status: 'preparing' | 'ready' | 'errored';
    duration: number;
    aspectRatio: string;
    playbackIds: Array<{ id: string; policy: string }>;
    staticRenditions: {
      status: 'preparing' | 'ready';
      files: Array<{ name: string; width: number; height: number }>;
    };
  }>;

  // Get playback URL for preview
  getPlaybackUrl(playbackId: string): string;
  // Returns: https://stream.mux.com/{playbackId}.m3u8

  // Get thumbnail
  getThumbnailUrl(playbackId: string, options?: {
    time?: number;
    width?: number;
    height?: number;
  }): string;
  // Returns: https://image.mux.com/{playbackId}/thumbnail.jpg

  // Get static MP4 for platform upload
  getStaticRenditionUrl(playbackId: string, quality: 'high' | 'medium' | 'low
'): string;
  // Returns: https://stream.mux.com/{playbackId}/{quality}.mp4

  // Delete asset
  deleteAsset(assetId: string): Promise<void>;
}
```

### 2.5 Mux Webhooks

Configure webhooks to receive asset status updates:

```
// Webhook payload: video.asset.ready
{
  "type": "video.asset.ready",
  "data": {
    "id": "asset_id",
    "status": "ready",
    "playback_ids": [{ "id": "playback_id", "policy": "public" }],
    "static_renditions": {
      "status": "ready",
      "files": [
        { "name": "high.mp4", "width": 1920, "height": 1080 },
        { "name": "medium.mp4", "width": 1280, "height": 720 }
      ]
    }
  }
}
```

### 2.6 Mux Pricing Estimate

| Component | Rate | Estimate (100 users, 50 videos/month) |
|---|---|---|
| Encoding | $0.015/min | 50 videos × 3 min = $22.50/mo |
| Storage | $0.007/GB/mo | 50GB = $3.50/mo |
| Streaming | $0.00096/min | 500 min preview = $0.48/mo |
| **Total** | | **~$27/month** |

*Scales linearly with usage. First 100GB encoding free for new accounts.*

---

## 3. Platform Integrations

### 3.1 Meta Platform (Facebook & Instagram)

#### *3.1.1 API Overview*

- **API:** Meta Graph API v19.0+
- **Auth:** OAuth 2.0 with Facebook Login
- **Documentation:** https://developers.facebook.com/docs/graph-api

#### *3.1.2 Required Permissions*

**Facebook Pages:**

```
pages_manage_posts      - Publish content to Pages
pages_read_engagement   - Read comments, reactions
```

```
pages_show_list         - List user's pages
pages_read_user_content - Read page content
```

**Instagram Professional Accounts:**

```
instagram_basic             - Read profile info
instagram_content_publish   - Publish posts and reels
instagram_manage_comments   - Read/reply to comments
instagram_manage_insights   - Access analytics
```

### *3.1.3 Key Limitations & Constraints*

| Constraint | Details | Impact |
|---|---|---|
| **App Review Required** | Must pass Meta app review for production | 2-4 weeks review time |
| **Business Accounts Only** | Instagram requires Professional/Business account | User onboarding requirement |
| **Rate Limits** | 200 calls/user/hour for Graph API | Implement queuing system |
| **Image Specs** | JPEG, PNG (max 8MB) | File validation required |
| **Video Specs** | MP4, max 1GB, H.264 codec | Mux handles transcoding |
| **Caption Limits** | Instagram: 2,200 chars, Facebook: 63,206 chars | UI validation |
| **Hashtag Limits** | Instagram: max 30 hashtags | UI counter |
| **No Native Scheduling** | Must implement server-side scheduling | Cron jobs required |

### *3.1.4 Video Requirements for Meta*

| Platform | Max Duration | Max Size | Aspect Ratio | Format |
|---|---|---|---|---|
| Facebook Feed | 240 min | 4GB | 16:9, 1:1, 4:5 | MP4, MOV |
| Facebook Reels | 90 sec | 1GB | 9:16 | MP4 |
| Instagram Feed | 60 min | 3.6GB | 16:9, 1:1, 4:5 | MP4, MOV |
| Instagram Reels | 90 sec | 1GB | 9:16 | MP4 |

### *3.1.5 App Review Requirements*

To publish to production, Meta requires: 1. Privacy Policy URL 2. Terms of Service URL 3. App Icon (1024x1024) 4. Detailed use case description 5. Screencast video demonstrating functionality 6. Data handling documentation 7. Business verification (for extended permissions)

#### *3.1.6 Implementation Checklist*

- ☐ Create Meta Developer App
- ☐ Configure Facebook Login product
- ☐ Implement OAuth flow with required scopes
- ☐ Handle token refresh (tokens expire in 60 days)
- ☐ Build content validation layer
- ☐ Implement rate limiting queue
- ☐ Create webhook handlers for engagement data
- ☐ Submit for App Review
- ☐ Business Verification (if required)

---

## 3.2 YouTube Integration

#### *3.2.1 API Overview*

- **API:** YouTube Data API v3
- **Auth:** OAuth 2.0 with Google Sign-In
- **Documentation:** https://developers.google.com/youtube/v3

#### *3.2.2 Required Scopes*

```
youtube.readonly             - Read channel info
youtube.upload               - Upload videos
youtube.force-ssl            - Required for all API calls
yt-analytics.readonly        - Access analytics (optional)
```

#### *3.2.3 Key Limitations & Constraints*

| Constraint | Details | Impact |
|---|---|---|
| **Quota System** | 10,000 units/day default | Request increase early |
| **Upload Costs** | Video upload = 1,600 units | ~6 videos/day max default |
| **Verification Required** | OAuth consent verification | 2-6 weeks for approval |
| **Video Size** | Max 256GB or 12 hours | Usually not an issue |
| **Thumbnail Upload** | 2MB max, 1280x720 recommended | Mux generates thumbnails |
| **Title Length** | Max 100 characters | UI validation |
| **Description** | Max 5,000 characters | UI validation |
| **Tags** | Max 500 characters total | UI counter |

#### *3.2.4 YouTube Video Specifications*

| Type | Duration | Aspect Ratio | Resolution |
|---|---|---|---|
| Regular Video | Up to 12 hours | 16:9 | Up to 8K |
| YouTube Shorts | ≤60 seconds | 9:16 | 1080x1920 |

### *3.2.5 Quota Considerations*

| Operation | Quota Cost |
|---|---|
| videos.insert (upload) | 1,600 units |
| videos.update | 50 units |
| channels.list | 1 unit |
| search.list | 100 units |

**Recommendation:** Apply for quota increase during development phase (typically granted within 1-2 weeks).

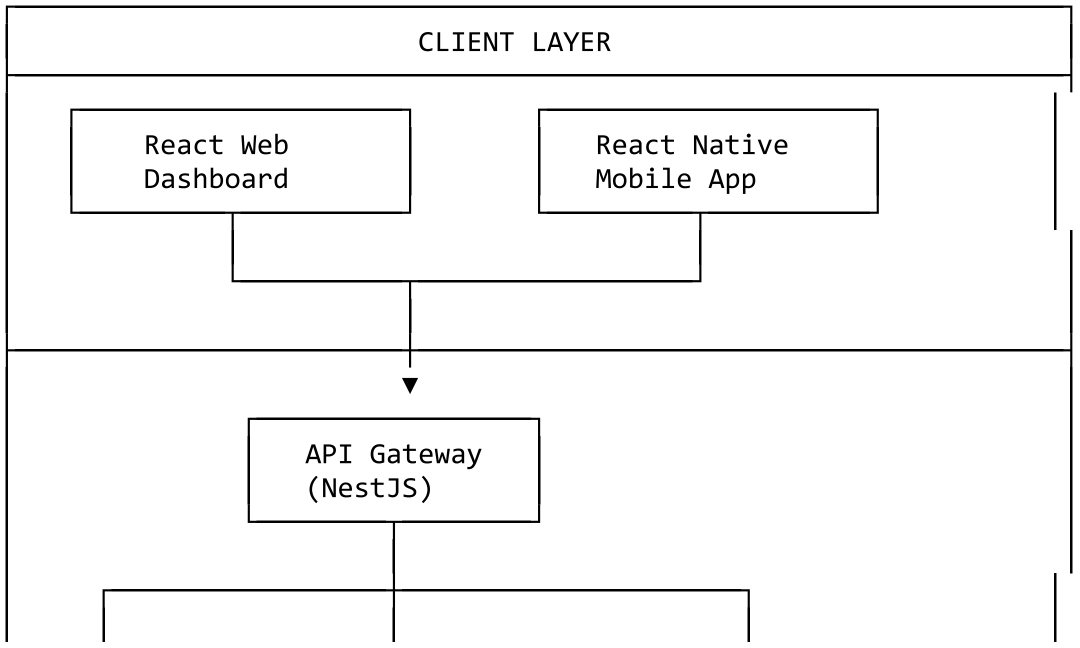
### *3.2.6 Implementation Checklist*

- ☐ Create Google Cloud Project
- ☐ Enable YouTube Data API v3
- ☐ Configure OAuth consent screen
- ☐ Implement Google Sign-In flow
- ☐ Use Mux MP4 renditions for upload
- ☐ Handle quota errors gracefully
- ☐ Implement video status polling (processing state)
- ☐ Submit OAuth consent for verification
- ☐ Request quota increase

---

## 4. Technical Architecture

### 4.1 System Overview



```
│     ▼                  ▼                    ▼              │
│  ┌────────┐   ┌─────────────┐    ┌──────────────┐          │
│  │  Auth  │   │    Posts    │    │  Scheduling  │          │
│  │Service │   │   Service   │    │   Service    │          │
│  └────────┘   └──────┬──────┘    └──────────────┘          │
│                      │                                     │
├──────────────────────┼─────────────────────────────────────┤
│                      ▼                                     │
│          ┌──────────────────────┐                          │
│          │   Publishing Engine  │                          │
│          └──────────┬───────────┘                          │
│                     │                                      │
│     ┌───────────────┼──────────────────┐                   │
│     │               │                  │                   │
│     ▼               ▼                  ▼                   │
│  ┌──────┐      ┌───────┐       ┌───────────┐               │
│  │ Mux  │      │ Meta  │       │  YouTube  │               │
│  │Video │      │ API   │       │   API     │               │
│  └──────┘      └───────┘       └───────────┘               │
│                                                            │
└────────────────────────────────────────────────────────────┘
```

## 4.2 Database Schema

```
-- Users and Authentication
users (
  id UUID PRIMARY KEY,
  email VARCHAR UNIQUE NOT NULL,
  name VARCHAR,
  avatar_url VARCHAR,
  created_at TIMESTAMP DEFAULT NOW()
);

oauth_tokens (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES users(id),
  platform VARCHAR NOT NULL, -- 'meta', 'youtube'
  access_token TEXT NOT NULL, -- encrypted
  refresh_token TEXT, -- encrypted
  expires_at TIMESTAMP,
  scope TEXT[],
  created_at TIMESTAMP DEFAULT NOW()
);

connected_accounts (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES users(id),
  platform VARCHAR NOT NULL,
  platform_user_id VARCHAR NOT NULL,
  username VARCHAR,
  display_name VARCHAR,
  avatar_url VARCHAR,
  account_type VARCHAR, -- 'page', 'profile', 'channel'
  metadata JSONB,
  created_at TIMESTAMP DEFAULT NOW()
);

-- Video Assets (Mux)
video_assets (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES users(id),
  mux_asset_id VARCHAR UNIQUE NOT NULL,
  mux_playback_id VARCHAR,
  status VARCHAR DEFAULT 'preparing', -- 'preparing', 'ready', 'errored'
  duration DECIMAL,
  aspect_ratio VARCHAR,
  thumbnail_url VARCHAR,
  original_filename VARCHAR,
  file_size BIGINT,
  created_at TIMESTAMP DEFAULT NOW()
);

-- Posts
posts (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES users(id),
  content TEXT,
  media_type VARCHAR, -- 'image', 'video', 'carousel'
  media_urls TEXT[],
  video_asset_id UUID REFERENCES video_assets(id),
  status VARCHAR DEFAULT 'draft', -- 'draft', 'scheduled', 'publishing', 'pub
lished', 'failed'
  scheduled_at TIMESTAMP,
  published_at TIMESTAMP,
  created_at TIMESTAMP DEFAULT NOW()
);

post_platforms (
  id UUID PRIMARY KEY,
  post_id UUID REFERENCES posts(id),
  platform VARCHAR NOT NULL, -- 'facebook', 'instagram', 'youtube'
  connected_account_id UUID REFERENCES connected_accounts(id),
  platform_post_id VARCHAR,
  platform_url VARCHAR,
  status VARCHAR DEFAULT 'pending', -- 'pending', 'published', 'failed'
  error_message TEXT,
  published_at TIMESTAMP
);

-- Link-in-Bio
```

```
link_pages (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES users(id),
  slug VARCHAR UNIQUE NOT NULL,
  display_name VARCHAR,
  bio TEXT,
  avatar_url VARCHAR,
  theme JSONB,
  created_at TIMESTAMP DEFAULT NOW()
);

link_items (
  id UUID PRIMARY KEY,
  link_page_id UUID REFERENCES link_pages(id),
  title VARCHAR NOT NULL,
  url VARCHAR NOT NULL,
  icon VARCHAR,
  position INTEGER,
  is_active BOOLEAN DEFAULT true,
  click_count INTEGER DEFAULT 0
);

-- Analytics
post_analytics (
  id UUID PRIMARY KEY,
  post_platform_id UUID REFERENCES post_platforms(id),
  impressions INTEGER,
  reach INTEGER,
  engagements INTEGER,
  likes INTEGER,
  comments INTEGER,
  shares INTEGER,
  fetched_at TIMESTAMP DEFAULT NOW()
);
```

### 4.3 API Endpoints

```
Authentication:
POST   /api/auth/register
POST   /api/auth/login
POST   /api/auth/refresh
GET    /api/auth/oauth/:platform          → Redirect to OAuth
GET    /api/auth/oauth/:platform/callback → Handle callback
DELETE /api/auth/oauth/:platform          → Disconnect

Video (Mux):
POST   /api/videos/upload-url             → Get Mux direct upload URL
GET    /api/videos/:id                    → Get video status
GET    /api/videos/:id/thumbnail          → Get thumbnail URL
DELETE /api/videos/:id                    → Delete video
```

```
Posts:
GET    /api/posts                          → List posts
POST   /api/posts                          → Create post
GET    /api/posts/:id                      → Get post
PUT    /api/posts/:id                      → Update post
DELETE /api/posts/:id                      → Delete post
POST   /api/posts/:id/publish              → Publish now
POST   /api/posts/:id/schedule             → Schedule post

Calendar:
GET    /api/calendar                       → Get scheduled posts
GET    /api/calendar/:date                 → Get posts for date

Connected Accounts:
GET    /api/accounts                       → List connected accounts
GET    /api/accounts/:platform             → Get platform accounts

Link-in-Bio:
GET    /api/link/:slug                     → Public page (no auth)
GET    /api/my-link                        → Get own link page
PUT    /api/my-link                        → Update link page
POST   /api/my-link/items                  → Add link item
PUT    /api/my-link/items/:id              → Update link item
DELETE /api/my-link/items/:id              → Delete link item

Analytics:
GET    /api/analytics/overview             → Dashboard stats
GET    /api/analytics/posts/:id            → Post analytics

Webhooks (internal):
POST   /api/webhooks/mux                   → Mux asset webhooks
```

---

## 5. Security Requirements

### 5.1 Data Protection

- All OAuth tokens encrypted at rest (AES-256)
- HTTPS required for all endpoints
- API keys stored in environment variables
- No sensitive data in logs
- Mux webhook signature verification

### 5.2 GDPR Compliance

- User data export functionality

- Account deletion with cascade (including Mux assets)
- Consent management for data processing
- Privacy policy integration

### 5.3 Platform Compliance

- Adhere to Meta and YouTube Terms of Service
- Implement required branding guidelines
- Handle platform-specific content policies
- Respect rate limits and quotas

---

## 6. Content Specifications

### 6.1 Supported Media Types

| Type | Formats | Max Size | Processing |
|---|---|---|---|
| Image | JPEG, PNG, WebP | 8MB | Direct upload |
| Video | MP4, MOV, AVI, MKV | 5GB | Via Mux |

### 6.2 Platform Content Matrix

| Field | Facebook | Instagram | YouTube |
|---|---|---|---|
| Caption/Title | 63,206 chars | 2,200 chars | 100 chars |
| Description | N/A | N/A | 5,000 chars |
| Hashtags | Unlimited | Max 30 | In description |
| Image Max | 8MB | 8MB | 2MB (thumbnail) |
| Video Max | 4GB | 3.6GB | 256GB |
| Video Duration | 240 min | 60 min (feed) | 12 hours |

---

## 7. Timeline & Approvals

### 7.1 Platform Approval Timeline

| Platform | Review Type | Expected Duration |
|---|---|---|
| Meta | App Review | 2-4 weeks |
| Meta | Business Verification | 1-2 weeks (if required) |
| Google | OAuth Consent Verification | 2-6 weeks |
| YouTube | Quota Increase | 1-2 weeks |

### 7.2 Development Timeline

| Phase | Duration | Deliverables |
|---|---|---|

| Phase | Duration | Deliverables |
|---|---|---|
| **Phase 1** | Weeks 1-4 | Auth, Mux integration, Meta OAuth |
| **Phase 2** | Weeks 5-8 | Publishing engine, YouTube, Scheduling |
| **Phase 3** | Weeks 9-12 | Analytics, Link-in-Bio, Polish |

## 8. Risk Assessment

### 8.1 Risk Matrix

| Risk | Likelihood | Impact | Mitigation |
|---|---|---|---|
| Meta app review rejection | Medium | High | Follow guidelines, prepare documentation |
| YouTube quota limits | Medium | Medium | Request increase early |
| Mux service outage | Low | High | Graceful degradation, status monitoring |
| API deprecation | Low | High | Abstract platform logic, monitor changelogs |
| Token expiration bugs | Medium | High | Robust refresh logic, monitoring |

## 9. Success Metrics

### 9.1 Technical KPIs

| Metric | Target |
|---|---|
| Video processing success rate | > 99% |
| Post publishing success rate | > 99% |
| API response time | < 500ms |
| System uptime | > 99.5% |
| Token refresh success rate | > 99.9% |

## Appendix A: API Quick Reference

### Mux API

```
Base URL: https://api.mux.com
Auth: Basic Auth (Token ID:Secret)
Docs: docs.mux.com
```

### Meta Graph API

```
Base URL: https://graph.facebook.com/v19.0
Auth: Bearer token
Docs: developers.facebook.com/docs/graph-api
```

### YouTube Data API

```
Base URL: https://www.googleapis.com/youtube/v3
Auth: Bearer token
Docs: developers.google.com/youtube/v3
```

---

**Document prepared by:** Studio X AS
**Version:** 2.0 (Mux Integration, Focused Platform Scope)