# YTEMpire Week 0 Execution Plan

## Leadership Team

### Role: CEO

#### Task 1: Strategic Vision Alignment [P0]

**Description**: Conduct kickoff meeting to align all team leads on MVP vision, success metrics, and 3-month roadmap. **Steps**:

1. Prepare vision deck with target metrics (10 beta users, 95% automation, <$3/video)
2. Schedule 2-hour all-hands meeting for Day 1 morning
3. Present roadmap phases and critical milestones
4. Document Q&A and concerns raised **Duration**: 4 hours **Dependencies**: None **Deliverable**: Vision deck, recorded meeting, alignment document

#### Task 2: Resource Allocation Finalization [P0]

**Description**: Approve and allocate $200K budget across teams with clear spending authority. **Steps**:

1. Review budget proposals from CTO and VP of AI
2. Allocate infrastructure budget ($50K), AI services ($30K), tools ($20K), reserve ($100K)
3. Set up approval workflows in finance system
4. Communicate spending limits to team leads **Duration**: 3 hours **Dependencies**: Budget proposals from technical leads **Deliverable**: Budget allocation spreadsheet with approval matrix

#### Task 3: Stakeholder Communication Plan [P1]

**Description**: Establish weekly reporting cadence and success metrics dashboard. **Steps**:

1. Design executive dashboard template
2. Set up weekly stakeholder update meetings
3. Create investor update template
4. Define escalation paths for blockers **Duration**: 2 hours **Dependencies**: Success metrics from Product Owner **Deliverable**: Communication calendar and dashboard template

### Role: CTO (Technical Director)

#### Task 1: Technical Architecture Documentation [P0]

**Description**: Create comprehensive technical architecture blueprint for all teams to follow. **Steps**:

1. Document microservices boundaries and API contracts

2. Define data flow between services (Frontend → Backend → AI → Database)
3. Specify technology stack versions and dependencies
4. Create deployment architecture (Docker, local vs cloud split) **Duration**: 6 hours **Dependencies**: Input from VP of AI on ML pipeline requirements **Deliverable**: 20-page architecture document in Confluence

**Task 2: Development Environment Standardization [P0]**

**Description**: Set up standardized development environment configuration for all engineers. **Steps**:

1. Create Docker Compose template with all services
2. Write setup scripts for MacOS, Windows, Linux
3. Configure IDE settings and extensions package
4. Test complete setup on fresh machine **Duration**: 4 hours **Dependencies**: None **Deliverable**: GitHub repo with setup scripts and docker-compose.yml

**Task 3: CI/CD Pipeline Foundation [P1]**

**Description**: Establish basic CI/CD pipeline for automated testing and deployment. **Steps**:

1. Set up GitHub Actions workflows for main repository
2. Configure automated testing on PR creation
3. Create staging deployment pipeline
4. Document deployment procedures **Duration**: 4 hours **Dependencies**: Repository structure created **Deliverable**: Working CI/CD pipeline with test deployment

**Task 4: Security Baseline Implementation [P1]**

**Description**: Establish security standards and initial implementations across infrastructure. **Steps**:

1. Configure secrets management (HashiCorp Vault or AWS Secrets Manager)
2. Set up SSL certificates for development domains
3. Implement basic API rate limiting rules
4. Create security checklist for code reviews **Duration**: 3 hours **Dependencies**: Infrastructure provisioning **Deliverable**: Security configuration and documentation

## Role: VP of AI

**Task 1: AI Service Integration Planning [P0]**

**Description**: Design and document AI service architecture including cost optimization strategies. **Steps**:

1. Map out AI pipeline: trend detection → content generation → quality assurance

2. Calculate API costs per video (target <$1.50 AI costs)
3. Design fallback chains (GPT-4 → GPT-3.5 → Claude)
4. Document prompt templates and optimization strategies **Duration**: 4 hours **Dependencies**: None **Deliverable**: AI architecture document with cost models

**Task 2: Model API Access Setup [P0]**

**Description**: Establish access to all required AI services and validate functionality. **Steps**:

1. Create OpenAI organization account and obtain API keys
2. Set up ElevenLabs account for voice synthesis
3. Configure Stable Diffusion API access
4. Test each API with sample requests and measure latency **Duration**: 3 hours **Dependencies**: Budget approval from CEO **Deliverable**: API credentials in secrets manager, test results document

**Task 3: Data Pipeline Architecture [P1]**

**Description**: Design data flow for ML model training and inference pipelines. **Steps**:

1. Define feature engineering pipeline structure
2. Design model versioning and deployment strategy
3. Create monitoring and logging architecture
4. Specify data storage requirements (vector DB, training data) **Duration**: 4 hours **Dependencies**: Database schema from Backend Lead **Deliverable**: Data pipeline architecture diagram and specifications

**Task 4: Cost Tracking Framework [P1]**

**Description**: Implement granular cost tracking for all AI operations. **Steps**:

1. Create cost tracking database schema
2. Implement API call logging with cost calculation
3. Design cost dashboard mockup
4. Set up alerts for cost overruns (>$3/video) **Duration**: 3 hours **Dependencies**: Database setup from Data Engineer **Deliverable**: Cost tracking system design and initial implementation

## Role: Product Owner

**Task 1: User Story Mapping [P0]**

**Description**: Create comprehensive user story map for MVP features with clear priorities. **Steps**:

1. Define primary user personas (content creators, agencies)

2. Map user journey from onboarding to first automated video
3. Break down epics into sprint-sized stories
4. Prioritize stories using MoSCoW method **Duration**: 4 hours **Dependencies**: Vision alignment from CEO **Deliverable**: User story map in Jira with 3-month backlog

#### Task 2: Success Metrics Definition [P0]

**Description**: Define and document all success metrics for MVP launch. **Steps**:

1. Define quantitative metrics (videos/day, automation %, cost/video)
2. Create user satisfaction metrics and feedback loops
3. Establish technical performance KPIs
4. Design metrics dashboard wireframe **Duration**: 3 hours **Dependencies**: Strategic vision from CEO **Deliverable**: Metrics framework document and dashboard design

#### Task 3: Beta User Recruitment Plan [P2]

**Description**: Develop strategy for recruiting and onboarding 10 beta users. **Steps**:

1. Define ideal beta user profile
2. Create recruitment messaging and channels
3. Design onboarding flow and materials
4. Set up feedback collection system **Duration**: 3 hours **Dependencies**: MVP timeline from CTO **Deliverable**: Beta recruitment plan and materials

## Technical Team (Under CTO)

### Role: Backend Team Lead

#### Task 1: API Service Scaffolding [P0]

**Description**: Create initial FastAPI project structure with basic endpoints and documentation. **Steps**:

1. Initialize FastAPI project with proper folder structure
2. Implement authentication/authorization skeleton
3. Create OpenAPI documentation configuration
4. Set up basic health check and monitoring endpoints **Duration**: 4 hours **Dependencies**: Architecture documentation from CTO **Deliverable**: GitHub repository with working API skeleton

#### Task 2: Database Schema Design [P0]

**Description**: Design and implement initial PostgreSQL schema for core entities. **Steps**:

1. Create ERD for users, channels, videos, analytics entities

2. Write migration scripts using Alembic
3. Implement connection pooling configuration
4. Create seed data for development **Duration**: 4 hours **Dependencies**: Requirements from Product Owner **Deliverable**: Database schema documentation and migration scripts

**Task 3: Message Queue Setup [P1]**

**Description**: Configure Redis and Celery for asynchronous task processing. **Steps**:

1. Set up Redis server configuration
2. Implement Celery worker structure
3. Create task routing configuration
4. Test with sample async tasks **Duration**: 3 hours **Dependencies**: Docker environment from DevOps **Deliverable**: Working message queue with example tasks

## Role: API Developer Engineer

**Task 1: Authentication Service Implementation [P1]**

**Description**: Build JWT-based authentication system with role-based access control. **Steps**:

1. Implement user registration endpoint
2. Create login/logout functionality with JWT tokens
3. Add role-based middleware for route protection
4. Write unit tests for auth endpoints **Duration**: 4 hours **Dependencies**: Database schema from Backend Lead **Deliverable**: Working authentication endpoints with tests

**Task 2: YouTube API Integration Setup [P1]**

**Description**: Establish YouTube Data API v3 integration for channel management. **Steps**:

1. Set up Google Cloud project and enable YouTube APIs
2. Implement OAuth 2.0 flow for YouTube authorization
3. Create wrapper functions for common YouTube operations
4. Test quota usage and implement rate limiting **Duration**: 4 hours **Dependencies**: API credentials from VP of AI **Deliverable**: YouTube API client library with auth flow

**Task 3: Error Handling Framework [P2]**

**Description**: Implement comprehensive error handling and logging system. **Steps**:

1. Create custom exception classes
2. Implement global error handler middleware

3. Set up structured logging with correlation IDs
4. Configure error reporting to monitoring service **Duration**: 3 hours **Dependencies**: Monitoring setup from DevOps **Deliverable**: Error handling framework with documentation

## Role: Data Pipeline Engineer

### Task 1: ETL Pipeline Architecture [P0]

**Description**: Design and implement basic ETL pipeline for YouTube analytics data. **Steps**:

1. Create Apache Airflow DAG structure
2. Implement YouTube Analytics data extraction tasks
3. Design data transformation logic
4. Set up PostgreSQL loading procedures **Duration**: 4 hours **Dependencies**: Database schema from Backend Lead **Deliverable**: Working ETL pipeline for analytics data

### Task 2: Real-time Event Streaming Setup [P1]

**Description**: Configure Kafka or Redis Streams for real-time event processing. **Steps**:

1. Set up Kafka/Redis Streams infrastructure
2. Create event producer clients
3. Implement consumer groups for different event types
4. Test with sample event flow **Duration**: 4 hours **Dependencies**: Docker environment from DevOps **Deliverable**: Event streaming system with example producers/consumers

### Task 3: Data Quality Framework [P2]

**Description**: Implement data validation and quality monitoring system. **Steps**:

1. Define data quality rules and constraints
2. Implement validation functions for incoming data
3. Create data quality dashboard queries
4. Set up alerts for data anomalies **Duration**: 3 hours **Dependencies**: ETL pipeline completion **Deliverable**: Data quality monitoring system

## Role: Integration Specialist

### Task 1: n8n Workflow Engine Setup [P0]

**Description**: Deploy and configure n8n for workflow automation. **Steps**:

1. Deploy n8n using Docker with PostgreSQL backend
2. Configure webhook endpoints for external triggers

3. Create example workflow for video processing
4. Set up credential management for integrations **Duration**: 4 hours **Dependencies**: Docker environment from DevOps **Deliverable**: Working n8n instance with example workflows

**Task 2: Third-party API Integration Framework [P1]**

**Description**: Create standardized framework for integrating external APIs. **Steps**:

1. Design abstraction layer for external API calls
2. Implement retry logic and circuit breakers
3. Create rate limiting and quota management
4. Add response caching mechanism **Duration**: 3 hours **Dependencies**: API structure from Backend Lead **Deliverable**: Integration framework with documentation

**Task 3: Webhook Management System [P2]**

**Description**: Build system for managing incoming and outgoing webhooks. **Steps**:

1. Create webhook endpoint registration system
2. Implement webhook signature verification
3. Add webhook retry logic for failures
4. Create webhook event log and monitoring **Duration**: 3 hours **Dependencies**: API framework from Backend Lead **Deliverable**: Webhook management system

## Role: Frontend Team Lead

**Task 1: React Application Scaffolding [P0]**

**Description**: Initialize React application with TypeScript and core dependencies. **Steps**:

1. Create React app with Vite and TypeScript configuration
2. Set up Material-UI theme and component library
3. Configure ESLint, Prettier, and Husky
4. Implement basic routing with React Router **Duration**: 3 hours **Dependencies**: None **Deliverable**: React application repository with development setup

**Task 2: State Management Architecture [P1]**

**Description**: Implement Zustand state management with proper store structure. **Steps**:

1. Design store structure for user, channels, videos domains
2. Implement authentication store with persistence
3. Create WebSocket store for real-time updates

4. Add development tools and debugging helpers **Duration**: 3 hours **Dependencies**: API contracts from Backend Lead **Deliverable**: State management layer with example stores

**Task 3: Component Library Foundation [P1]**

**Description**: Create base component library with design system. **Steps**:

1. Implement core UI components (Button, Input, Card)
2. Create layout components (Header, Sidebar, Container)
3. Build feedback components (Toast, Modal, Loading)
4. Document components in Storybook **Duration**: 4 hours **Dependencies**: Design system from UI/UX Designer **Deliverable**: Component library with Storybook documentation

## Role: React Engineer

**Task 1: Authentication UI Implementation [P1]**

**Description**: Build login, registration, and password reset interfaces. **Steps**:

1. Create login form with validation
2. Implement registration flow with error handling
3. Build password reset request and confirmation pages
4. Add JWT token management and refresh logic **Duration**: 4 hours **Dependencies**: Authentication API from API Developer **Deliverable**: Complete authentication UI flow

**Task 2: Dashboard Layout Structure [P2]**

**Description**: Implement main dashboard layout with navigation. **Steps**:

1. Create responsive dashboard shell with sidebar
2. Implement navigation menu with routing
3. Add breadcrumb navigation component
4. Build user profile dropdown menu **Duration**: 3 hours **Dependencies**: Component library from Frontend Lead **Deliverable**: Dashboard layout with navigation

## Role: Dashboard Specialist

**Task 1: Data Visualization Setup [P1]**

**Description**: Configure Recharts library and create example visualizations. **Steps**:

1. Install and configure Recharts with TypeScript
2. Create reusable chart wrapper components
3. Implement example line, bar, and pie charts

4. Add real-time data update capability **Duration**: 3 hours **Dependencies**: React app setup from Frontend Lead **Deliverable**: Chart component library with examples

**Task 2: Metrics Dashboard Wireframe [P2]**

**Description**: Build initial metrics dashboard with mock data. **Steps**:

1. Create dashboard grid layout with responsive design
2. Implement KPI cards with trend indicators
3. Add channel performance comparison charts
4. Build video analytics timeline view **Duration**: 4 hours **Dependencies**: Chart components completion **Deliverable**: Working metrics dashboard with mock data

## Role: UI/UX Designer

**Task 1: Design System Documentation [P0]**

**Description**: Create comprehensive design system for consistent UI development. **Steps**:

1. Define color palette, typography, and spacing system
2. Document component specifications and states
3. Create iconography guidelines
4. Provide Figma component library **Duration**: 4 hours **Dependencies**: Brand guidelines from CEO **Deliverable**: Design system documentation and Figma library

**Task 2: MVP User Flow Designs [P1]**

**Description**: Design complete user flows for core MVP features. **Steps**:

1. Create onboarding flow wireframes and mockups
2. Design channel management interface
3. Mock up video generation and queue interface
4. Design analytics dashboard layouts **Duration**: 6 hours **Dependencies**: User stories from Product Owner **Deliverable**: Complete Figma designs for MVP flows

## Role: Platform Ops Lead

**Task 1: Infrastructure Provisioning [P0]**

**Description**: Set up hybrid local/cloud infrastructure for development and staging. **Steps**:

1. Configure local Ryzen 9 9950X3D server with Ubuntu
2. Set up AWS/GCP cloud resources for staging
3. Configure networking and VPN access

4. Implement backup and disaster recovery plan **Duration**: 4 hours **Dependencies**: Budget approval from CEO **Deliverable**: Working infrastructure with access documentation

**Task 2: Monitoring Stack Setup [P1]**

**Description**: Deploy Prometheus, Grafana, and alerting systems. **Steps**:

1. Deploy Prometheus with service discovery
2. Configure Grafana dashboards for key metrics
3. Set up AlertManager with PagerDuty integration
4. Create runbook documentation **Duration**: 4 hours **Dependencies**: Infrastructure provisioning **Deliverable**: Monitoring stack with example dashboards

## Role: DevOps Engineer

**Task 1: Docker Environment Configuration [P0]**

**Description**: Create Docker Compose setup for entire application stack. **Steps**:

1. Write Dockerfile for each service
2. Create docker-compose.yml with all services
3. Configure networking and volume management
4. Add development vs production configurations **Duration**: 4 hours **Dependencies**: Service specifications from Backend Lead **Deliverable**: Complete Docker development environment

**Task 2: Kubernetes Preparation [P2]**

**Description**: Prepare Kubernetes manifests for future production deployment. **Steps**:

1. Create deployment manifests for each service
2. Configure service discovery and ingress
3. Set up ConfigMaps and Secrets
4. Document scaling policies **Duration**: 3 hours **Dependencies**: Docker containers built **Deliverable**: Kubernetes deployment manifests

## Role: Security Engineer

**Task 1: Security Audit Framework [P1]**

**Description**: Establish security scanning and vulnerability management. **Steps**:

1. Set up SAST tools in CI/CD pipeline
2. Configure dependency vulnerability scanning
3. Implement secrets scanning in repositories

4. Create security incident response plan **Duration**: 3 hours **Dependencies**: CI/CD pipeline from CTO **Deliverable**: Security scanning pipeline and documentation

#### Task 2: Access Control Implementation [P1]

**Description**: Configure RBAC and secure access patterns. **Steps**:

1. Design role hierarchy and permissions matrix
2. Implement API key management system
3. Configure OAuth 2.0 for third-party integrations
4. Set up audit logging for access events **Duration**: 4 hours **Dependencies**: Authentication system from API Developer **Deliverable**: Access control system with audit logs

### Role: QA Engineer

#### Task 1: Test Framework Setup [P1]

**Description**: Establish testing frameworks for backend and frontend. **Steps**:

1. Configure Jest for backend unit tests
2. Set up React Testing Library for frontend
3. Implement Cypress for E2E testing
4. Create test data factories **Duration**: 4 hours **Dependencies**: Application scaffolding from development teams **Deliverable**: Complete test framework with examples

#### Task 2: Test Environment Configuration [P2]

**Description**: Set up dedicated testing environment with data isolation. **Steps**:

1. Create isolated test database
2. Configure test API endpoints
3. Set up mock services for external APIs
4. Document test environment access **Duration**: 3 hours **Dependencies**: Infrastructure from DevOps **Deliverable**: Isolated test environment

## AI Team (Under VP of AI)

### Role: AI/ML Team Lead

#### Task 1: ML Pipeline Architecture [P0]

**Description**: Design end-to-end ML pipeline from data ingestion to model serving. **Steps**:

1. Define feature engineering pipeline
2. Design model training and validation workflow

3. Create model versioning and deployment strategy
4. Document monitoring and retraining triggers **Duration**: 4 hours **Dependencies**: Data pipeline design from VP of AI **Deliverable**: ML pipeline architecture document

#### Task 2: Model Serving Infrastructure [P1]

**Description**: Set up model serving infrastructure for inference. **Steps**:

1. Configure TorchServe or TensorFlow Serving
2. Implement model loading and caching
3. Create API endpoints for model inference
4. Set up performance monitoring **Duration**: 4 hours **Dependencies**: Infrastructure from Platform Ops **Deliverable**: Model serving infrastructure with example model

### Role: ML Engineer

#### Task 1: Trend Detection Model Setup [P1]

**Description**: Implement initial trend detection model using Prophet or similar. **Steps**:

1. Set up time series data preprocessing
2. Configure Prophet model with custom seasonality
3. Implement backtesting framework
4. Create model performance dashboard **Duration**: 4 hours **Dependencies**: Historical data from Data Engineer **Deliverable**: Working trend detection model with 70% accuracy baseline

#### Task 2: Content Quality Scoring [P2]

**Description**: Build content quality scoring system for generated videos. **Steps**:

1. Define quality metrics (engagement, retention, CTR)
2. Implement scoring algorithm
3. Create feedback loop for model improvement
4. Set up A/B testing framework **Duration**: 3 hours **Dependencies**: Video metadata schema from Backend Lead **Deliverable**: Quality scoring system with baseline metrics

### Role: Data Engineer (AI Team)

#### Task 1: Training Data Pipeline [P0]

**Description**: Build data pipeline for ML model training datasets. **Steps**:

1. Set up data extraction from YouTube Analytics
2. Implement feature engineering transformations

3. Create data versioning system
4. Build data quality validation **Duration**: 4 hours **Dependencies**: Database access from Backend Lead **Deliverable**: Automated training data pipeline

**Task 2: Vector Database Setup [P1]**

**Description**: Configure vector database for content similarity search. **Steps**:

1. Deploy Pinecone or Weaviate instance
2. Create embedding generation pipeline
3. Implement similarity search API
4. Test with sample content embeddings **Duration**: 3 hours **Dependencies**: Infrastructure from Platform Ops **Deliverable**: Working vector database with search API

## Role: Data Engineer 2 (AI Team)

**Task 1: Real-time Feature Store [P1]**

**Description**: Implement feature store for real-time ML inference. **Steps**:

1. Set up Redis for feature caching
2. Create feature computation pipeline
3. Implement feature versioning
4. Build feature monitoring dashboard **Duration**: 4 hours **Dependencies**: Redis setup from Backend Lead **Deliverable**: Feature store with real-time serving capability

**Task 2: Model Monitoring System [P2]**

**Description**: Build monitoring system for model performance and drift. **Steps**:

1. Implement prediction logging
2. Create drift detection algorithms
3. Set up alerting for performance degradation
4. Build model performance dashboard **Duration**: 3 hours **Dependencies**: Model serving infrastructure from ML Team Lead **Deliverable**: Model monitoring system with alerts

## Role: Analytics Engineer

**Task 1: Metrics Pipeline Development [P1]**

**Description**: Build analytics pipeline for business and technical metrics. **Steps**:

1. Define core business metrics (CAC, LTV, engagement)
2. Implement metrics computation using dbt

3. Create scheduled metric updates
4. Build metrics API for dashboard consumption **Duration**: 4 hours **Dependencies**: Database schema from Backend Lead **Deliverable**: Automated metrics pipeline with API

**Task 2: Cost Analytics Framework [P1]**

**Description**: Implement detailed cost tracking and analysis system. **Steps**:

1. Create cost allocation model per video/channel
2. Build API usage tracking and cost calculation
3. Implement cost optimization recommendations
4. Create cost analytics dashboard **Duration**: 3 hours **Dependencies**: API integration from Integration Specialist **Deliverable**: Cost analytics system with dashboard

**Task 3: Reporting Infrastructure [P2]**

**Description**: Set up automated reporting for stakeholders. **Steps**:

1. Create report templates for different audiences
2. Implement scheduled report generation
3. Set up email delivery system
4. Build self-service reporting interface **Duration**: 3 hours **Dependencies**: Metrics pipeline completion **Deliverable**: Automated reporting system

## Daily Standup Schedule

### Day 1 (Monday)

- 9:00 AM: All-hands kickoff meeting (CEO leading)
- 11:00 AM: Technical team sync (CTO leading)
- 2:00 PM: AI team sync (VP of AI leading)
- 4:00 PM: End-of-day progress check

### Day 2 (Tuesday)

- 9:00 AM: Daily standup (15 min)
- 10:00 AM: Backend team working session
- 2:00 PM: Frontend team working session
- 4:00 PM: Integration testing sync

### Day 3 (Wednesday)

- 9:00 AM: Daily standup (15 min)

- 10:00 AM: Infrastructure review (Platform Ops)
- 2:00 PM: AI pipeline review
- 4:00 PM: Mid-week progress assessment

### Day 4 (Thursday)

- 9:00 AM: Daily standup (15 min)
- 10:00 AM: API integration sync
- 2:00 PM: Security and QA review
- 4:00 PM: Dependency resolution meeting

### Day 5 (Friday)

- 9:00 AM: Daily standup (15 min)
- 10:00 AM: Final integration testing
- 2:00 PM: Week 0 retrospective
- 3:00 PM: Week 1 planning session
- 4:00 PM: Executive summary and demo

## Critical Path Items

### Must Complete by End of Day 2

- Development environment working for all engineers
- API scaffolding and database schema defined
- Docker environment configured
- AI service credentials obtained
- Infrastructure provisioned

### Must Complete by End of Day 4

- Authentication system functional
- Basic CI/CD pipeline running
- n8n workflow engine deployed
- ML pipeline architecture finalized
- Frontend application scaffolding complete

### Must Complete by End of Week 0

- End-to-end hello world video generation
- All team members productive in development environment

- Cost tracking framework operational
- Security baseline implemented
- Week 1 sprint planned with all dependencies mapped

## Success Criteria

### Technical Success Metrics

- [ ] All 17 team members have working development environment
- [ ] Database schema supports 5+ channels per user
- [ ] API can handle 100 requests/second
- [ ] Docker compose brings up entire stack in <2 minutes
- [ ] Cost tracking accurate to $0.01 per video

### Process Success Metrics

- [ ] All P0 tasks completed by Day 2
- [ ] All P1 tasks completed by Day 4
- [ ] Zero blocking dependencies by end of week
- [ ] All team members have committed code
- [ ] Documentation wiki has 50+ pages

### Integration Success Metrics

- [ ] Frontend can authenticate with backend
- [ ] Backend can call AI services successfully
- [ ] n8n can trigger video generation pipeline
- [ ] Monitoring shows all services healthy
- [ ] One test video generated end-to-end

## Risk Mitigation

### Technical Risks

- **YouTube API Quotas**: Implement caching and quota monitoring Day 1
- **AI Service Costs**: Set up cost alerts and fallback chains Day 2
- **Local Hardware Delays**: Have cloud backup plan ready
- **Integration Complexity**: Daily integration testing from Day 2

### Team Risks

- **Knowledge Gaps**: Pair programming and knowledge sharing sessions
- **Communication Silos**: Daily standups and Slack channels
- **Dependency Blocks**: Twice-daily dependency check-ins

- **Scope Creep**: Strict P0/P1/P2 prioritization enforcement

## Week 0 Deliverables Checklist

### Documentation

- [ ] Technical architecture document (20 pages)
- [ ] API specification (OpenAPI format)
- [ ] Database schema (ERD + migrations)
- [ ] ML pipeline design document
- [ ] Security baseline document
- [ ] Cost model and projections

### Code Repositories

- [ ] Backend API (FastAPI + PostgreSQL)
- [ ] Frontend App (React + TypeScript)
- [ ] ML Pipeline (Python + PyTorch)
- [ ] Infrastructure as Code (Terraform/Docker)
- [ ] n8n Workflows (JSON exports)

### Infrastructure

- [ ] Development environment (Docker Compose)
- [ ] CI/CD pipeline (GitHub Actions)
- [ ] Monitoring stack (Prometheus + Grafana)
- [ ] Staging environment (Cloud)
- [ ] Backup and recovery system

### Team Enablement

- [ ] All team members onboarded
- [ ] Development environments verified
- [ ] Access credentials distributed
- [ ] Communication channels established
- [ ] Week 1 sprint planned

## End of Week 0 Demo

### Demo Agenda (Friday 4:00 PM)

1. **Infrastructure Tour** (10 min)
   - Show Docker environment running
   - Demonstrate monitoring dashboards
   - Display CI/CD pipeline execution

2. **API Walkthrough** (10 min)
   - Show API documentation
   - Demonstrate authentication flow
   - Execute sample API calls
3. **Frontend Preview** (10 min)
   - Show login/registration UI
   - Demonstrate dashboard layout
   - Display component library
4. **AI Pipeline Demo** (10 min)
   - Show trend detection model output
   - Demonstrate content generation
   - Display cost tracking
5. **End-to-End Test** (15 min)
   - Generate one test video from UI
   - Show video in YouTube (private)
   - Display cost breakdown
   - Show metrics dashboard
6. **Week 1 Preview** (5 min)
   - Review sprint backlog
   - Highlight dependencies
   - Set expectations

---

*This document represents the complete Week 0 execution plan for YTEMpire MVP. Each task has been carefully designed to enable rapid development in subsequent weeks while establishing a solid foundation for scaling to 10 beta users and beyond.*