



Narrative News - Complete Project Master Plan

AI-Powered News Analysis Platform: Automated system that scrapes articles from left and right-leaning sources, generates unbiased AI analysis comparing perspectives, and monetizes through ads and newsletters.



Project Overview

Core Concept

- **Automated Content Generation:** Scrapes news from multiple political perspectives
- **AI Analysis:** Uses OpenAI GPT-4 to compare how different outlets frame the same stories
- **Revenue Generation:** Monetized through Google AdSense and email newsletters
- **Hands-Off Operation:** Fully automated system requiring minimal maintenance
- **Scalable Architecture:** Built to handle growth from 0 to 100k+ users

Business Model

- **Primary Revenue:** Google AdSense (banner, sidebar, in-article, footer ads)
 - **Secondary Revenue:** Email newsletter sponsorships and premium subscriptions
 - **Content Strategy:** 5-10 new analyzed articles daily via automation
 - **Target Audience:** News consumers seeking unbiased perspectives
-



Technical Architecture

Tech Stack

- **Framework:** Next.js 14 (App Router) with TypeScript
- **Database:** PostgreSQL with Prisma ORM
- **AI Service:** OpenAI GPT-4 for content analysis
- **Email:** Nodemailer with SMTP (Gmail/SendGrid)
- **Styling:** Tailwind CSS + custom components
- **Hosting:** Vercel (recommended) or Railway
- **Caching:** Redis for performance optimization
- **Monitoring:** Built-in health checks + optional Sentry

Key Dependencies

```
json
{
  "critical": [
    "next@^14.0.0",
    "@prisma/client@^5.7.0",
    "openai@^4.20.0",
    "nodemailer@^6.9.0",
    "cheerio@^1.0.0", // Web scraping
    "rss-parser@^3.13.0", // RSS feed parsing
    "node-cron@^3.0.0", // Automation scheduling
    "zod@^3.22.0" // Environment validation
  ],
  "optional": [
    "bull@^4.12.0", // Queue management
    "ioredis@^5.3.0", // Redis caching
    "sharp@^0.33.0" // Image optimization
  ]
}
```

10-Module Development Plan

Module 1: Framework & Foundation



Objective: Set up bulletproof project structure and deployment configuration

Critical Tasks:

- ☐ Initialize Next.js 14 project with App Router
- ☐ Configure TypeScript with strict settings
- ☐ Set up Prisma with PostgreSQL
- ☐ Create environment variable validation (env.mjs)
- ☐ Configure next.config.js for production deployment
- ☐ Set up package.json with all required dependencies
- ☐ Create deployment-ready folder structure

Deliverables:

- ✓ Package.json with complete dependency list
- ✓ Next.config.js with security headers and optimization
- ✓ Environment variable templates (.env.example)

-  TypeScript configuration (tsconfig.json)
-  Project folder structure following Next.js 14 App Router conventions

Key Files:

```
narrative-news/  
├── src/app/           # Next.js App Router  
├── src/components/    # React components  
├── src/lib/           # Business logic  
├── prisma/            # Database schema  
├── env.mjs            # Environment validation  
├── next.config.js     # Build configuration  
└── package.json       # Dependencies
```

Integration Points: Provides foundation for all other modules







Module 2: Frontend Foundation

Objective: Create responsive, SEO-optimized frontend layouts

Critical Tasks:

- ☐ Design root layout with navigation and footer
- ☐ Create homepage with value proposition
- ☐ Implement responsive design with Tailwind CSS
- ☐ Add SEO meta tags and Open Graph
- ☐ Set up global styles and component patterns
- ☐ Create error boundaries and loading states
- ☐ Implement client/server component patterns correctly

Deliverables:

-  Root layout (src/app/layout.tsx)
-  Homepage (src/app/page.tsx)
-  Global styles (src/app/globals.css)
-  Navigation components
-  Responsive design system
-  SEO optimization

Key Components:

- Header with navigation
- Hero section explaining the concept
- Footer with social links
- Loading and error UI patterns

Integration Points: Used by Modules 3, 8, 9 for UI components

Module 3: Article Components

Objective: Build reusable components for displaying articles and AI analysis

Critical Tasks:

- ☐ Create ArticleCard component with left/right source display
- ☐ Build detailed article view with AI analysis
- ☐ Implement Newsletter signup component
- ☐ Create TypeScript interfaces for all data structures
- ☐ Add article interaction tracking
- ☐ Implement responsive article layouts
- ☐ Create article search and filtering UI

Deliverables:

- ☒ ArticleCard component with perspective comparison
- ☒ Newsletter signup form with validation
- ☒ Complete TypeScript type definitions
- ☒ Article detail page layout
- ☒ Mobile-responsive article display

Key Features:

- Side-by-side source comparison
- AI analysis highlighting
- Source credibility indicators
- Reading time estimation
- Social sharing buttons

Integration Points: Consumes data from Module 7 APIs, uses Module 2 layouts

Module 4: Database & Models

Objective: Design and implement complete database schema with business logic

Critical Tasks:

- ☐ Design Prisma schema for articles, sources, users, analytics
- ☐ Create database service layer with CRUD operations
- ☐ Implement data relationships and constraints
- ☐ Add database indexes for performance
- ☐ Create migration and seeding scripts
- ☐ Add data validation and error handling
- ☐ Implement connection pooling and optimization

Deliverables:

- ☒ Complete Prisma schema (prisma/schema.prisma)
- ☒ Database service with all operations (src/lib/db.ts)
- ☒ Database seeding script (prisma/seed.ts)
- ☒ Migration files
- ☒ Connection optimization

Database Tables:

- `articles` - Analyzed news stories
- `news_sources` - Original articles from outlets
- `subscribers` - Newsletter email list
- `rss_feeds` - News source configurations
- `article_analytics` - View and click tracking
- `email_campaigns` - Newsletter management

Integration Points: Core dependency for Modules 5, 6, 7, 8, 9

Module 5: News Automation System

Objective: Build automated news scraping and content matching system

Critical Tasks:

- ☐ Create RSS feed parser for multiple news sources
- ☐ Implement web scraping for full article content
- ☐ Build story matching algorithm (same events, different outlets)
- ☐ Add content deduplication system
- ☐ Create automation scheduler with cron jobs
- ☐ Implement error handling and retry logic
- ☐ Add queue system for reliable processing

Deliverables:

- ☒ RSS feed scraping service (src/lib/scrapper.ts)
- ☒ Content matching algorithm
- ☒ Automation scheduler (src/lib/automation.ts)
- ☒ API endpoints for manual control
- ☒ Duplicate detection system

Key Features:

- Multi-source RSS monitoring
- Intelligent article matching
- Content extraction and cleaning
- Automated scheduling (every 6 hours)
- Manual trigger capabilities

Integration Points: Uses Module 4 database, triggers Module 6 AI analysis

Module 6: AI Analysis System

Objective: Integrate OpenAI for intelligent article analysis and comparison

Critical Tasks:

- ☐ Set up OpenAI API integration
- ☐ Create analysis prompts for unbiased comparison
- ☐ Implement structured response parsing
- ☐ Add fallback mechanisms for API failures
- ☐ Create batch processing for multiple articles
- ☐ Add confidence scoring and quality metrics
- ☐ Implement cost optimization strategies

Deliverables:

- ☒ OpenAI service integration (src/lib/ai.ts)
- ☒ Analysis generation system
- ☒ API endpoints for AI operations
- ☒ Fallback content system
- ☒ Cost monitoring and optimization

Key Features:

- GPT-4 powered analysis
- Structured comparison output
- Bias detection and explanation
- Quality confidence scoring
- Batch processing capabilities

Integration Points: Called by Module 5 automation, serves Module 7 APIs

Module 7: API Routes & Backend

Objective: Create comprehensive REST API for all frontend functionality

Critical Tasks:

- ☐ Build article CRUD endpoints
- ☐ Create search and filtering APIs
- ☐ Implement newsletter subscription endpoints
- ☐ Add analytics tracking APIs
- ☐ Create rate limiting middleware
- ☐ Add input validation and error handling
- ☐ Implement caching strategies

Deliverables:

- ☒ Complete REST API endpoints
- ☒ Rate limiting and security middleware
- ☒ Input validation with Zod
- ☒ Error handling patterns

-  API response standardization

API Endpoints:

- `GET /api/articles` - Article listings with filters
- `POST /api/newsletter` - Email subscriptions
- `GET /api/analytics` - Usage statistics
- `POST /api/ai-analysis` - Manual AI analysis
- `GET /api/search` - Article search

Integration Points: Serves all frontend modules, uses Modules 4, 5, 6






Module 8: Email & Monetization

Objective: Implement revenue generation through ads and email marketing

Critical Tasks:

- ☐ Set up email service with beautiful HTML templates
- ☐ Integrate Google AdSense with strategic placement
- ☐ Create automated newsletter campaigns
- ☐ Build subscriber management system
- ☐ Implement email analytics and tracking
- ☐ Add revenue optimization features
- ☐ Create premium subscription infrastructure

Deliverables:

-  Email service with automation (`src/lib/email.ts`)
-  Google AdSense integration components
-  Newsletter HTML templates
-  Revenue tracking and analytics
-  Subscriber management tools

Revenue Features:

- Daily automated newsletters
- Strategic ad placement (banner, sidebar, in-article)
- Subscriber segmentation

- Email performance tracking
- Premium subscription ready

Integration Points: Uses Module 4 for subscribers, Module 7 for APIs

Module 9: Admin Dashboard

Objective: Build comprehensive control panel for system management

Critical Tasks:

- ☐ Create system health monitoring
- ☐ Build automation control interface
- ☐ Implement analytics dashboard with charts
- ☐ Add subscriber and content management
- ☐ Create RSS feed management tools
- ☐ Build admin authentication system
- ☐ Add system alerts and notifications

Deliverables:

- ☒ Complete admin dashboard (src/app/admin/)
- ☒ System health monitoring
- ☒ Automation control panel
- ☒ Analytics visualization
- ☒ Content management interface

Dashboard Features:

- Real-time system status
- Automation start/stop/schedule controls
- Traffic and revenue analytics
- Subscriber management
- Content moderation tools
- RSS feed configuration

Integration Points: Uses all previous modules, provides system oversight

Module 10: Deployment & Production

Objective: Configure bulletproof production deployment and monitoring

Critical Tasks:

- ☐ Create production environment configurations
- ☐ Set up database migrations and seeding
- ☐ Configure monitoring and error tracking
- ☐ Implement backup and recovery procedures
- ☐ Create Docker containerization
- ☐ Set up CI/CD pipelines
- ☐ Add performance optimization

Deliverables:

- ☒ Production deployment configurations
- ☒ Environment variable templates
- ☒ Docker setup and health checks
- ☒ Deployment guides for multiple platforms
- ☒ Monitoring and alerting system

Production Features:

- Zero-downtime deployments
- Automated backups
- Performance monitoring
- Error tracking and alerting
- Scalability configurations

Integration Points: Deploys and monitors entire system

Critical Integration Fixes Required

1. App Router Consistency

- Ensure all API routes use Next.js 14 App Router patterns
- Fix dynamic route parameter handling (await params)
- Correct server/client component declarations

2. Database Connection Optimization

- Implement Prisma singleton pattern to prevent connection exhaustion
- Add connection pooling configuration
- Create query optimization for N+1 problems

3. Environment Validation

- Create env.mjs with Zod validation for all environment variables
- Add build-time environment checks
- Ensure development/production parity

4. Error Handling

- Add error boundaries for all route segments
- Implement graceful fallbacks for external service failures
- Create comprehensive logging system

5. Caching Strategy

- Add Redis caching for API responses
- Implement CDN optimization
- Create cache invalidation strategies



Development Timeline

Phase 1: Foundation (Week 1-2)

- **Day 1-3:** Module 1 (Framework & Foundation)
- **Day 4-6:** Module 2 (Frontend Foundation)
- **Day 7-10:** Module 4 (Database & Models)
- **Day 11-14:** Integration testing and fixes

Phase 2: Core Features (Week 3-4)

- **Day 15-18:** Module 3 (Article Components)
- **Day 19-22:** Module 7 (API Routes & Backend)
- **Day 23-26:** Module 5 (News Automation)
- **Day 27-28:** Integration testing

Phase 3: Intelligence & Revenue (Week 5-6)

- **Day 29-32:** Module 6 (AI Analysis System)
- **Day 33-36:** Module 8 (Email & Monetization)
- **Day 37-40:** Revenue optimization testing
- **Day 41-42:** Performance optimization

Phase 4: Management & Deployment (Week 7-8)

- **Day 43-46:** Module 9 (Admin Dashboard)
 - **Day 47-50:** Module 10 (Deployment & Production)
 - **Day 51-54:** End-to-end testing
 - **Day 55-56:** Production deployment and monitoring
-

Success Metrics

Technical Metrics

- **Uptime:** >99.5% system availability
- **Performance:** <2s average page load time
- **Automation:** 95%+ successful article processing rate
- **AI Quality:** >8/10 analysis quality score

Business Metrics

- **Content:** 5-10 new analyzed articles daily
 - **Audience:** 1,000+ newsletter subscribers by month 3
 - **Revenue:** \$500+ monthly ad revenue by month 6
 - **Engagement:** 3+ minute average session time
-

Security & Compliance

Data Protection

- **GDPR Compliance:** Cookie consent, data export, right to deletion
- **Email Compliance:** CAN-SPAM compliance, one-click unsubscribe
- **Content Security:** XSS protection, SQL injection prevention

- **API Security:** Rate limiting, input validation, authentication

Monitoring & Alerting

- **System Health:** Automated health checks every 5 minutes
 - **Error Tracking:** Real-time error alerts via Sentry
 - **Performance:** Core Web Vitals monitoring
 - **Revenue:** Daily revenue tracking and alerts
-

Revenue Projections

Conservative Estimates

- **Month 1-2:** \$0-50 (building audience)
- **Month 3-4:** \$100-300 (initial monetization)
- **Month 5-6:** \$300-800 (optimization phase)
- **Month 7-12:** \$800-2,500 (scale and growth)

Growth Multipliers

- **Premium Features:** +30-50% revenue potential
 - **Sponsored Content:** +25-40% revenue potential
 - **API Access:** +15-25% revenue potential
 - **White Label:** +100-200% revenue potential
-

Risk Mitigation

Technical Risks

- **API Limits:** OpenAI rate limiting → Implement queueing and fallbacks
- **Database Load:** Connection exhaustion → Connection pooling and caching
- **Scraping Blocks:** Sites blocking scrapers → Rotate user agents and proxies
- **Deployment Issues:** Production failures → Staging environment and rollback procedures

Business Risks

- **AdSense Approval:** Rejection risk → Build quality content first, apply after 3 months
- **Content Quality:** Poor AI analysis → Human review process and quality metrics

- **Legal Issues:** Copyright concerns → Fair use compliance and source attribution
 - **Competition:** Market saturation → Unique positioning and superior automation
-

Support & Resources

Development Resources

- **Next.js Documentation:** <https://nextjs.org/docs>
- **Prisma Documentation:** <https://www.prisma.io/docs>
- **OpenAI API Docs:** <https://platform.openai.com/docs>
- **Vercel Deployment:** <https://vercel.com/docs>

Business Resources

- **Google AdSense:** <https://adsense.google.com/start/>
 - **Email Services:** SendGrid, Mailgun, or Gmail SMTP
 - **Analytics:** Google Analytics 4, Mixpanel
 - **Domain & Hosting:** Namecheap, Vercel, Railway
-

Pre-Launch Checklist

Technical Readiness

- ☐ All 10 modules implemented and tested
- ☐ Database migrations run successfully
- ☐ Environment variables configured
- ☐ SSL certificates active
- ☐ DNS configuration complete
- ☐ Monitoring and alerting active

Content Readiness

- ☐ RSS feeds configured and tested
- ☐ AI analysis quality validated
- ☐ Email templates designed and tested
- ☐ Legal pages created (Privacy, Terms)
- ☐ SEO optimization complete

Business Readiness

- ☐ Google AdSense approved and ads showing
 - ☐ Email service configured and tested
 - ☐ Analytics tracking implemented
 - ☐ Social media accounts created
 - ☐ Launch marketing plan prepared
-

Post-Launch Roadmap

Month 1-3: Optimization

- Fine-tune AI analysis quality
- Optimize ad placement and revenue
- Build initial subscriber base
- Monitor and fix any technical issues

Month 4-6: Growth Features

- Add more news sources and categories
- Implement user accounts and personalization
- Create mobile app or PWA
- Add social sharing and viral features

Month 7-12: Scale & Monetize

- Launch premium subscription tiers
 - Add sponsored content opportunities
 - Explore API licensing
 - Consider white-label opportunities
-

Final Notes

This master plan provides a complete roadmap for building Narrative News from initial development through successful deployment and monetization. The modular approach allows for iterative development and testing, while the comprehensive integration plan ensures all components work together seamlessly.

Key Success Factor: Following the deployment-first approach throughout development to avoid the integration issues that plagued your previous project.

Total Estimated Development Time: 8 weeks for full implementation **Total Estimated Cost:** \$200-500 for services during development **Revenue Potential:** \$500-2,500/month within first year

The system is designed to be largely hands-off once deployed, with the admin dashboard providing full control over all aspects without requiring code changes.