🎯 MASTER PROJECT BRIEF: ROUTIX Platform

📋 EXECUTIVE SUMMARY

You are the Lead Agent responsible for architecting, planning, and delegating the complete development of **Routix** - a chat-based AI thumbnail generation platform. Your role is to:

1. **Understand** the complete project requirements
2. **Design** the system architecture
3. **Create** detailed technical specifications
4. **Generate** individual prompts for specialized agents
5. **Coordinate** the development workflow
6. **Ensure** all components integrate seamlessly

🎨 PROJECT VISION

Routix is a revolutionary platform that combines:

- **ChatGPT-like conversational interface** for natural user interaction
- **AI-powered thumbnail generation** using proprietary template library
- **Intelligent template matching** with computer vision
- **Multi-algorithm generation** (branded as "Routix Versions")

Key Innovation: Instead of pure AI generation, we use a secret bank of 90 professionally designed templates. When users request thumbnails, we:

1. Analyze their request with AI
2. Find best matching template (semantic search)
3. Extract template's "design DNA" using Vision AI
4. Regenerate thumbnail with user's custom elements (face, logo, text)
5. Result: Unique, professional thumbnail (user never sees original templates)

📜 TECHNICAL REQUIREMENTS

Backend Requirements

Stack:

- Framework: FastAPI (Python 3.11+)
- Database: PostgreSQL 15+ with pgvector extension
- Cache/Queue: Redis 7+
- Task Queue: Celery with Celery Beat
- Storage: Cloudflare R2 (S3-compatible)
- Authentication: JWT

AI Services Integration:

- Google Gemini Vision API (primary image analysis)
- OpenAI GPT-4 Vision (fallback/advanced analysis)
- OpenAI Embeddings (text-embedding-3-small for semantic search)
- Midjourney API via GoAPI.ai or UseAPI.net (primary generation)
- OpenAI DALL-E 3 (alternative generation algorithm)

Core Features:

1. **Template Management System**
   - Upload templates (single & bulk with drag-and-drop)
   - Automatic AI analysis to extract "design DNA"
   - Vector embeddings for semantic search
   - Template performance tracking
   - Active/inactive status management
2. **Generation Algorithm Management** (Routix Versions)
   - Multiple branded algorithms (e.g., "Routix v1", "Routix Pro", "Routix Lightning")
   - Each algorithm uses different AI provider or configuration
   - Configurable parameters per algorithm
   - Performance metrics tracking
   - Cost tracking per algorithm
3. **Thumbnail Generation Pipeline**

- Receive user request (text prompt + optional assets)
- Analyze request intent with Vision AI
- Semantic search for best matching templates (top 3)
- Extract template design DNA
- Compose generation prompt based on selected algorithm
- Generate thumbnail with AI (Midjourney/DALL-E/etc)
- Real-time progress tracking
- Result delivery

4. **User Management**
   - Registration/Login (JWT authentication)
   - Credit-based system
   - Subscription tiers (Free, Basic, Pro, Enterprise)
   - Usage tracking
   - Asset library (uploaded faces, logos)

5. **Admin Panel APIs**
   - Dashboard analytics
   - Template management (CRUD, bulk operations)
   - User management (credits, bans, subscriptions)
   - Algorithm management (CRUD, configuration)
   - Generation monitoring (real-time, history, failures)
   - System health monitoring
   - Audit trail

6. **Real-time Features**
   - WebSocket or SSE for generation progress updates
   - Status: analyzing → searching → composing → generating → completed
   - Progress percentage (0-100)

Database Schema Requirements:

- Templates table with vector embeddings
- Generation algorithms table
- Users table with credits

- Generation requests table with status tracking
- Analytics/feedback tables
- Admin actions audit trail
- Conversation/messages tables for chat history

Frontend Requirements

Stack:

- Framework: Next.js 14+ (App Router)
- Language: TypeScript
- Styling: Tailwind CSS + Custom CSS
- UI Components: Custom (match specific design)
- Animations: Framer Motion
- State Management: Zustand
- API: Axios + TanStack Query (React Query)
- Forms: React Hook Form + Zod

Design System (CRITICAL - Must Match Exactly):

Visual Style:

Background: Soft gradient (purple → blue → pink)
  linear-gradient(135deg, #E0C3FC 0%, #D5E1FF 25%, #E8F4FF 50%, #FFE8F5 75%, #FFE5E5 100%)

Cards: Glassmorphism effect
  - Semi-transparent white background
  - Backdrop blur (20px)
  - Subtle borders
  - Soft shadows

Typography:
  - Font: SF Pro Display / Inter
  - Primary text: #2D2A4A (dark navy)
  - Secondary text: #6B6B8D (muted purple-gray)

Colors:
  - Primary: #6B5DD3 (purple)
  - Accent: #8B7AFF (light purple)
  - Buttons: Gradient purple

Effects:
  - Smooth animations (60fps)
  - Hover states with scale
  - Loading states with shimmer
  - Micro-interactions everywhere

User Interface Architecture:

PRIMARY INTERFACE: Chat-Based (Like ChatGPT)

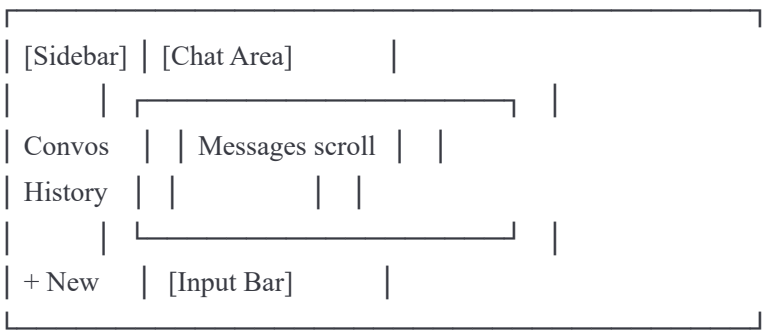
The entire user experience is conversational:

1. **Welcome Screen** (Empty Chat State)

  - Large centered logo (Routix "RK")
  - Greeting: "Hey [Username]! Can I help you with anything?"
  - Subtitle: "Ready to assist you with anything you need."
  - Input box at bottom
  - Suggested prompt chips (quick actions)

2. **Chat Interface** (Main Feature)

Layout:



Message Types:
- User messages: Purple bubble (right-aligned)
- AI text messages: Glass card (left-aligned)
- Thumbnail results: Large image card with actions
- Progress updates: Animated status messages
- Interactive elements: Algorithm selectors, file uploads

Chat Flow Example:
User: "Create gaming thumbnail with explosions"
AI: "Got it! 🎮 Let me create that for you..."
AI: [Algorithm Selector: Routix v1, Pro, Lightning]
User: [Clicks Routix v1]
AI: "Would you like to add your face or logo?"
AI: [Upload buttons inline]
User: [Uploads face]
AI: "Perfect! Generating now... ✨"
AI: [Progress: "Analyzing..." → "Generating..." → "Complete!"]
AI: [Thumbnail Image Card]
  - Preview image
  - Download/Regenerate/Share buttons
  - Rating (thumbs up/down)

3. **Sidebar** (Conversation History)

- List of past conversations
- Auto-titled from first message
- "New Chat" button at top
- Search conversations
- Delete/Archive options

4. **Admin Panel** (Separate Section)

Gradient background maintained
Glass card style throughout

Main sections:
- Dashboard (stats, charts, real-time activity)
- Templates (CRITICAL FEATURE):
  * Drag & Drop upload zone (multiple files)
  * Grid/List view toggle
  * Bulk selection
  * Analyze button
  * Performance metrics per template
- Algorithms (Routix Versions):
  * CRUD interface
  * Configuration editor (JSON)
  * Performance comparison
- Users:
  * List with search/filter
  * Credit management
  * Ban/Unban
- Generations:
  * Monitor all generations
  * Filter by status
  * Retry failed
- Analytics:
  * Charts (Recharts)
  * Export data

Key UI Components to Build:

- GlassCard (reusable glassmorphism container)
- GradientButton (purple gradient with hover effects)
- ChatBubble (user vs AI styling)
- ThumbnailCard (result display)
- ProgressIndicator (animated status)
- FileUploadZone (drag & drop with preview)
- AlgorithmSelector (interactive picker)
- Sidebar (conversation list)
- Input bar (with file attach, send button)

🎯 FUNCTIONAL REQUIREMENTS

User Flow (End-to-End)

Generation Flow:

1. User opens app → Sees chat interface
2. Types prompt: "Create energetic gaming thumbnail"
3. Backend receives → Analyzes with Gemini Vision
4. Searches templates using embeddings → Finds top 3 matches
5. Selects best template → Extracts design DNA
6. AI asks in chat: "Which algorithm? Routix v1 (Recommended), Pro, Lightning"
7. User selects Routix v1
8. AI asks: "Add face or logo?" (shows upload buttons)
9. User uploads face image
10. Backend composes Midjourney prompt using:
    - Template design DNA
    - User's prompt
    - Face as character reference
11. Sends to Midjourney API
12. Polls status every 2 seconds
13. Updates chat with progress:
    - "Analyzing your request..." (10%)
    - "Matching templates..." (30%)
    - "Composing design..." (50%)
    - "Generating thumbnail..." (70-95%)
    - "Complete!" (100%)
14. Shows thumbnail in chat with:
    - Download button
    - Regenerate button (with tweaks)
    - Share button
    - Rating (thumbs up/down)
15. User can continue chatting or start new conversation

Admin Flow:

1. Admin logs in → Admin dashboard
2. Navigates to Templates
3. Clicks "Bulk Upload"
4. Drags 20 thumbnail images into zone
5. Adds metadata (category, tags)
6. Clicks "Upload All"
7. System uploads to R2 storage
8. Creates database entries (status: pending_analysis)
9. Triggers Celery task for each template
10. Background: Vision AI analyzes each template
11. Extracts design DNA (composition, colors, typography, etc.)
12. Generates embedding vector
13. Updates database (status: analyzed)
14. Admin sees real-time progress in UI
15. Templates now available for generation

📊 DATA ARCHITECTURE

Key Data Models:

1. **Template**
   - ID, image_url, thumbnail_url
   - style_dna (JSONB - complete design analysis)
   - embedding (vector(1536) - for semantic search)
   - category, tags, description
   - has_face, has_text, has_logo
   - energy_level (1-10)
   - performance_score, usage_count, success_rate
   - is_active, is_featured, priority
2. **GenerationAlgorithm** (Routix Versions)
   - name (e.g., "routix_v1")
   - display_name (e.g., "Routix v1")
   - ai_provider (midjourney/dalle3/sdxl)
   - config (JSONB - algorithm-specific settings)
   - cost_per_generation, credit_cost
   - is_active, is_default

- Performance metrics

3. **GenerationRequest**
   - user_id, algorithm_id
   - prompt, style_preference
   - user_face_url, user_logo_url, custom_text
   - status (pending/analyzing/generating/completed/failed)
   - progress (0-100)
   - selected_template_id
   - final_thumbnail_url
   - processing_time, cost_incurred
4. **Conversation** (Chat History)
   - user_id
   - title (auto-generated)
   - messages (array or separate table)
   - created_at, updated_at
5. **Message**
   - conversation_id
   - role (user/assistant/system)
   - type (text/thumbnail/progress/algorithm-select/file-upload)
   - content (text or structured data)
   - timestamp

🔧 TECHNICAL CHALLENGES TO SOLVE

1. **Template Analysis Pipeline**
   - How to extract comprehensive design DNA from images
   - Prompt engineering for Vision AI
   - Structured output parsing (JSON)
2. **Semantic Search Optimization**
   - Creating meaningful embeddings from template DNA
   - Balancing vector similarity with rule-based filters

- Performance with thousands of templates

3. **Real-time Communication**
   - WebSocket vs SSE vs polling for progress updates
   - Connection reliability and fallbacks
4. **Midjourney Integration**
   - Handling unofficial API limitations
   - Rate limiting and retry logic
   - Style reference + character reference usage
5. **Chat State Management**
   - Maintaining context across messages
   - Handling file uploads in chat flow
   - Conversation persistence and retrieval
6. **Admin Bulk Operations**
   - Uploading multiple large files efficiently
   - Background processing without blocking UI
   - Progress tracking for batch operations

📦 DELIVERABLES

Phase 1: Backend Foundation

- [ ] FastAPI application structure
- [ ] Database schema with migrations
- [ ] PostgreSQL + pgvector setup
- [ ] Redis + Celery configuration
- [ ] Authentication system (JWT)
- [ ] Basic CRUD endpoints

Phase 2: AI Integration

- [ ] Gemini Vision service integration
- [ ] OpenAI embeddings integration
- [ ] Midjourney API wrapper
- [ ] Template analysis pipeline
- [ ] Semantic search implementation

Phase 3: Core Features

- [ ] Template management system
- [ ] Algorithm management system
- [ ] Generation pipeline (end-to-end)
- [ ] Real-time progress tracking
- [ ] User credit system

Phase 4: Admin Panel Backend

- [ ] Admin dashboard APIs
- [ ] Bulk upload endpoints
- [ ] Analytics endpoints
- [ ] User management APIs
- [ ] System monitoring APIs

Phase 5: Frontend Foundation

- [ ] Next.js application setup
- [ ] Design system (colors, components, animations)
- [ ] Glassmorphism components (GlassCard, etc.)
- [ ] Layout structure

Phase 6: Chat Interface

- [ ] Chat container and message list
- [ ] Message components (user, AI, thumbnail, progress)
- [ ] Chat input with file upload
- [ ] Conversation sidebar
- [ ] Welcome screen
- [ ] Real-time updates

Phase 7: Admin Panel Frontend

- [ ] Admin dashboard
- [ ] Template management UI (with drag & drop)
- [ ] Algorithm management UI
- [ ] User management UI
- [ ] Analytics dashboards

Phase 8: Integration & Polish

- [ ] Backend-Frontend integration

- [ ] Error handling
- [ ] Loading states
- [ ] Animations and transitions
- [ ] Responsive design
- [ ] Performance optimization

Phase 9: Testing & Deployment

- [ ] Unit tests (critical functions)
- [ ] Integration tests (API endpoints)
- [ ] E2E tests (user flows)
- [ ] Docker configuration
- [ ] Deployment scripts
- [ ] Environment setup (dev, staging, prod)

🎯 SUCCESS CRITERIA

Backend:

- ✅ All API endpoints functional and documented
- ✅ Template analysis produces quality design DNA
- ✅ Semantic search returns relevant results
- ✅ Generation pipeline completes successfully
- ✅ Real-time updates work reliably
- ✅ Admin operations complete without errors
- ✅ Response times < 500ms (excluding AI calls)
- ✅ Handle 100 concurrent users

Frontend:

- ✅ Chat interface matches design specifications exactly
- ✅ Animations are smooth (60fps)
- ✅ Real-time progress updates feel natural
- ✅ Drag & drop upload works flawlessly
- ✅ Mobile responsive (bonus)
- ✅ Load time < 2 seconds
- ✅ No console errors

- ✅ Accessible (keyboard navigation, screen readers)

Integration:

- ✅ User can generate thumbnail end-to-end
- ✅ Admin can upload and analyze templates
- ✅ All Routix versions work correctly
- ✅ Chat history persists and loads correctly
- ✅ File uploads work in chat flow
- ✅ Error states display appropriately

🚀 AGENT DELEGATION STRATEGY

You (Lead Agent) should:

1. **Break down the project** into specialized tasks
2. **Assign tasks** to appropriate specialized agents:
   - Backend Architect Agent
   - Database Design Agent
   - API Development Agent
   - AI Integration Agent
   - Frontend Architect Agent
   - UI/UX Implementation Agent
   - DevOps Agent
3. **Create detailed prompts** for each specialized agent with:
   - Clear objectives
   - Technical requirements
   - Expected deliverables
   - Integration points
   - Success criteria
4. **Define integration contracts** between components:
   - API endpoint specifications
   - Data schemas
   - Event formats

- Error handling conventions

5. **Establish development workflow**:
   - Development sequence (what must be built first)
   - Dependencies between tasks
   - Testing requirements
   - Review checkpoints

📋 YOUR IMMEDIATE TASKS

1. **Analyze** this complete project brief
2. **Design** detailed system architecture (diagrams if possible)
3. **Create** technical specifications for each component
4. **Generate** individual prompts for:
   - Backend development agent
   - Frontend development agent
   - Database design agent
   - AI integration agent
   - DevOps agent
5. **Define** integration contracts and APIs
6. **Plan** development timeline and dependencies
7. **Output** comprehensive development plan with all prompts

---

🎨 DESIGN REFERENCE

Visual Style Context: The UI design is inspired by modern glassmorphism with soft gradients. Think:

- Apple's design language (smooth, premium)
- ChatGPT's clean conversation interface
- Vercel's gradient backgrounds
- Linear's smooth interactions

Key Visual Elements:

- Soft purple-blue-pink gradient backgrounds

- Frosted glass cards with backdrop blur
- Smooth animations and transitions
- Purple gradient buttons
- Clean, spacious layouts
- Subtle shadows and highlights

⚠️ CRITICAL CONSTRAINTS

1. **Template Secrecy**: Original templates must NEVER be exposed to users. They only see generated results.
2. **Real-time UX**: Generation progress must update smoothly (every 2-3 seconds) to feel responsive.
3. **Chat Experience**: Must feel natural and conversational, not form-based.
4. **Admin Efficiency**: Bulk operations must be fast and show clear progress.
5. **Visual Fidelity**: Frontend MUST match the glassmorphism design style exactly.
6. **AI Reliability**: Handle API failures gracefully with retry logic and fallbacks.
7. **Cost Management**: Track all AI API costs and provide analytics.

💡 ADDITIONAL CONTEXT

Why This Project Matters: This platform solves a real problem: creating professional YouTube thumbnails is time-consuming and requires design skills. By combining AI with professional design templates, we offer the best of both worlds - AI speed with human design quality.

Target Users:

- YouTubers (gaming, tutorials, vlogs, reviews)
- Content creators on tight schedules
- Small businesses creating video content
- Marketing agencies managing multiple channels

Business Model:

- Free tier: 10 credits
- Basic: $9/mo (50 credits)
- Pro: $29/mo (200 credits)
- Enterprise: Custom pricing

Future Roadmap (not for initial build, but keep in mind):

- A/B testing thumbnails
- Analytics from YouTube
- Brand kit management
- Team collaboration
- API access for integration
- Mobile apps

🎯 FINAL DIRECTIVE

You are now the Lead Agent for Routix.

Your mission: Create a complete, production-ready platform that combines ChatGPT's conversational UX with AI-powered design generation.

Generate:

1. Complete system architecture document
2. Individual detailed prompts for each specialized agent
3. API specifications and contracts
4. Database schema with migrations
5. Development timeline with milestones
6. Testing strategy
7. Deployment plan

Remember:

- Quality over speed
- User experience is paramount
- Every interaction should feel magical
- Code must be maintainable and well-documented
- Security and privacy are non-negotiable

Begin your analysis and planning. Break down this complex system into manageable components and create the roadmap to build Routix.

Good luck, Lead Agent. Build something extraordinary. 🚀