Project plan (high level overview)

Project: ACT 2.0

This document outlines the technical strategy, infrastructure, and development workflow for upgrading `onbrand.act.agency` to ACT 2.0. The goal is to evolve the platform into a more powerful, AI-native brand guardian that not only checks assets but actively creates and strategizes.

The target launch for this initiative is January 2026.

Proposed Development Infrastructure & Architecture

The architecture for ACT 2.0 will be a modern, scalable stack designed for AI-native applications. Based on the documentation and the GitHub repository, the core infrastructure will consist of the following components:

- **Frontend Framework:** The application will be built using **React** framework, as indicated in the repository. This provides a robust foundation for server-side rendering, static site generation, and a seamless developer experience.

- **Backend-as-a-Service (BaaS): Supabase** will serve as a primary backend service (MCP). Its responsibilities will include:
    - **Database:** PostgreSQL for storing user data, brand configurations, and generated content.
    - **Authentication:** Managing user logins and permissions through its built-in auth system.
    - **Storage:** Hosting brand assets, AI-generated images, and other files.
    - **Serverless Functions:** For custom backend logic and secure API integrations.

- **AI Integration Layer:** The **Vercel AI SDK** is the designated framework for connecting to and interacting with large language models. This is crucial for building streaming chat interfaces and ensuring a responsive user experience for features like text generation and the `CHAT module`.

- **UI/UX Component Library:** The team will research and select a reusable component library, with **`shadcn/ui`** being a primary candidate. This will help enforce design consistency and accelerate development by translating brand guidelines into actionable UX and UI principles.

- **Version Control & Deployment: GitHub** will be used for source code management. All development will follow established Git workflows, and CI/CD pipelines (likely via GitHub Actions or Vercel) will automate testing and deployment.

- **Email & Notifications: Resend** is the selected service for handling transactional emails. This will be used for user invitations, system alerts, and potentially the new **Push notifications** feature.

V2 Feature Implementation Strategy

V2 Feature Implementation Strategy

The new features for V2 represent a significant leap in AI capability. Here's a brief on the technical approach for the key features:

- **BRAND BRAIN + RAG (Retrieval-Augmented Generation)**: This is the core AI feature. The plan involves creating a vector database (likely using `pgvector` within Supabase) from each client's brand documentation. When a user prompts an AI agent, the RAG system will first retrieve the most relevant sections of the brand guidelines and feed them to the language model as context. This ensures all generated content (text, strategy ideas, etc.) is deeply aligned with the brand.

- **Agents Powered Workflows**: Building on the "BRAND BRAIN," these agents will be AI systems capable of performing multi-step tasks. For example, a "Campaign Agent" could take a brief, use the RAG system to access brand guidelines, generate social media copy , create brand-aligned images (potentially using custom-trained models), and schedule them in a content calendar. A `Node builder` is also being considered to potentially allow for custom workflow creation.

- **Own LORA Training**: This moves beyond generic AI image generation. The plan is to provide a service where we can fine-tune image models (like Stable Diffusion) using a brand's specific visual assets. This will allow the AI to generate `LORA Illustrations` and other visuals that perfectly match a unique brand aesthetic, not just brand colors or fonts.

- **Chatbot MCP Tooling**: This feature will focus on providing a simplified toolset for creating MVP-level chatbots. The initial integration target for this is **Simplicate**. This suggests the chatbot could pull data from or push data to the Simplicate CRM/project management system.

Design and Development Workflow

Design and Development Workflow

Design Process

The design phase follows a user-first approach to ensure V2 features solve real problems:

1. **Discovery** – Map current user journeys to identify pain points and opportunities
2. **Systemization** – Translate brand guidelines into actionable UX/UI principles
3. **Prototyping** – Validate new user flows with low-fidelity wireframes before development

Development Process

The team follows an Agile sprint-based methodology with two-week cycles:

- **Sprint Planning** – All work is tracked in Monday.com with clear priorities and dependencies
- **Estimation** – Tasks are estimated using Fibonacci sequence, targeting 30 story points per sprint to meet the January 2026 deadline

Core Technology Stack

Core Technology Stack

The foundation of ACT 2.0 is a modern, AI-native technology stack chosen for scalability and developer experience.

- **Framework**: The application will be built using **React**. The GitHub repository indicates this will be implemented within the **Next.js** framework, which is ideal for building high-performance, server-rendered applications that can seamlessly integrate AI features.
- **AI Integration**: The **Vercel AI SDK** is the designated tool for connecting the frontend to various language models. This is critical for building responsive, streaming interfaces for features like the `CHAT module` and `Text Generation`.
- **Backend and Database**: **Supabase** will serve as the primary backend platform. It will provide the PostgreSQL database, user authentication, file storage for assets, and serverless functions for custom logic, creating a unified backend environment.
- **UI Component Library**: The team will use a reusable component library to ensure design consistency and speed up development. The primary candidate for this is **`shadcn/ui`**, which allows for building a custom, accessible design system from scratch. This will be used in conjunction with **Tailwind CSS**.

Development Environment & Tooling

To ensure a consistent and efficient workflow, the team will standardize its development environment and tools.

- **Code Editor**: The team will use **Cursor/Windsurf**, an AI-first code editor. It's crucial that all developers **sync their settings** to ensure everyone is working from the same AI context and development rules.
- **Version Control**: **GitHub** will be used for all source code management and version control.
- **Project Management**: All sprints, tasks, and tickets will be created and tracked using **Monday.com**.

Coding Standards & Workflow

A structured workflow and clear coding standards are essential for managing the project's complexity and meeting the January 2026 deadline.

- **Development Rules**: All code, whether written by a developer or generated by AI, must adhere to predefined standards. The team will follow the rules outlined by **Cursor for React** and the prompts from the **Supabase UI AI editor rules**.
- **Sprint Methodology**: The project will follow a two-week sprint cycle.
  - **Estimation**: Tasks will be estimated using the **Fibonacci number sequence** to assign story points.
  - **Sprint Capacity**: The team aims for a velocity of approximately **30 story points per sprint**.
- **CI/CD & Production Safety**: A continuous integration and deployment pipeline will be established, likely using GitHub Actions or Vercel. To safeguard the live application, the project will incorporate a tool like `bugbot` for production protection.

Sprints

Phase 0: Project Initialization & Team Alignment

This phase focuses on setting up the tools, processes, and shared understanding required for the project.

- **1. Create the Project Plan Markdown File:**
  - The first action item is to create a markdown (`.md`) file detailing the project plan. This file should be committed to the GitHub repository to give the AI an understanding of the project's scope.

- **2. Set Up Project Management Board:**
  - Set up the project in **Monday.com** to be used for tracking all tickets and tasks.
  - Invite Daniel and Zara for accounts
  - Monday.com walkthrough

- **3. Align Development Environment:**
  - Schedule a meeting for the development team to **sync their Cursor settings**. This is essential for ensuring everyone is developing from the same context.

- **4. Train the Team on Estimation:**
  - Ensure everyone on the team understands how to estimate tasks using the **Fibonacci number sequence**. This is required for your sprint planning process.

- **5. Set up Project access to Figma**
  - Ensure everyone on the team has access to figma.
  - Ensure figma MCP is loaded
  - Guard rails around the type of prompts to add so the Agent understands the context.

- 6. **Set up project guardrains**
  - How do we commit - message with commit
  - How do we test - Do we have a “BUG BOT” agent to do a final check?
  - Defining Figma rules -- https://developers.figma.com/docs/figma-mcp-server/add-custom-rules/

Phase 1: "Sprint 0" - Scoping & Design Foundation

This phase is dedicated to the detailed planning and design groundwork before the first development sprint begins.

- **1. Breakdown Technical Elements into Tickets:**
  - As a team, conduct a planning session to **break down all technical elements into sprint-based tickets**. Populate the Monday.com backlog with these tickets, keeping the **January 2026 timeline** in mind.

- **2. Map Current User Journeys:**
  - The design/UX lead should begin **mapping out the current user journeys** of the existing platform. This will inform the new design and identify areas for improvement.
  - **Deliverable** - A detailed report on the current use journey mapping with information of potential pain points identified.
  - **Deliverable** - A user flow diagram - E.g an example
  - **Deliverable -** discover a Figma / shadcn design in which we can translate all the current user journeys into low **-** fi wireframes.
  - **Deliverable** - define an ideal workflow from - UX designs into - Production react components - Figma MCP
  - **Deliverable** - use annotations to mock up a design

- **3. Define UX/UI Principles:**
  - Start the process of **translating the existing brand guidelines into actionable UX and UI principles**. This will be the foundation for the new component library and overall design.
  - Examples - https://www.figma.com/resource-library/ui-design-principles/
  - **Deliverable** Component Library Mapping Document
    - shadcn/ui component selection and customization guide
    - Brand-specific component variants
    - Component composition patterns
    - Do's and Don'ts for each component type
    - Usage examples with code snippets

Brand-to-Code Translation Guide

- Step-by-step process for converting brand elements to code
- Figma variable naming conventions that map to CSS/Tailwind
- Color palette with hex codes, Tailwind classes, and CSS variables
- Typography scale with font sizes, weights, and line heights

Phase 2: First Official Sprint

With the groundwork laid, we can now begin our first two-week development sprint.

- **1. Hold First Sprint Planning Meeting:**
  - Review the ticket backlog in Monday.com together as a team.
  - Estimate the tickets for the upcoming sprint and decide what work will be done over the next two weeks.
- **2. Begin Low-Fidelity Design:**
  - A key goal for the design team in the first sprint should be to **create low-fidelity UX flow diagrams** for the highest-priority V2 features.
- **3. Technical R&D:**
  - Assign a development ticket to formally evaluate the reusable UI libraries (like `shadcn/ui` and `kibo-ui`) to decide which is best for your specific needs.

Estimation sessions (Larger tickets)

Chatbot MCP Tooling

1. Chatbot MCP Tooling - Simplicate Integration

Vision: Create an intelligent chatbot that connects to Simplicate CRM, allowing users to query project data, track hours, view clients, and manage workflows without leaving the ACT platform.

Why This Matters:

- Reduces context switching between platforms
- Provides conversational interface for business operations
- Enables natural language queries for complex data
- First step toward agent-based automation

Technical Considerations:

- Simplicate MCP needs to expose project management, time tracking, and client data
- Should support both query (read) and command (write) operations
- Needs to handle authentication securely (API keys, OAuth)
- Consider rate limiting and caching strategies
- Real-time updates vs polling trade-offs

User Stories:

- As a project manager, I want to ask "How many hours did we spend on Project X this week?" and get an instant answer
- As an account manager, I want to quickly pull up client information while in a meeting
- As a team member, I want to log hours conversationally: "Log 3 hours to ACT website under development"
- As a finance person, I want to check invoice status without switching to Simplicate

Tickets to Create:

Research & Planning:

- Audit Simplicate API documentation and identify all available endpoints
- Map current Simplicate workflows that should be accessible via chat
- Define conversation flows for common queries
- Document authentication and security requirements
- Create data privacy and access control matrix
- Prototype basic query/response patterns

Infrastructure:

- Set up Simplicate MCP server configuration
- Implement secure credential storage (environment variables, vault)
- Create connection health monitoring and error handling

- Build rate limiting and request throttling
- Set up logging for all Simplicate interactions
- Create fallback mechanisms for API failures

Core Chatbot Engine:

- Design chat interface component (message list, input, typing indicators)
- Implement message state management (sending, delivered, error states)
- Build intent recognition system (what is the user asking for?)
- Create response formatting for different data types (tables, cards, lists)
- Add conversation context and memory
- Implement multi-turn conversation handling

Simplicate Query Features:

- Project listing and search functionality
- Hour entry retrieval and filtering
- Client information lookup
- Invoice and quote viewing
- Team member availability checking
- Project budget and progress tracking

Simplicate Command Features:

- Hour entry creation via chat
- Project status updates
- Task creation and assignment
- Client note additions
- Quick reminders and follow-ups

UX Enhancements:

- Quick action buttons (common queries)
- Suggested questions based on context
- Export conversations or data
- Search within chat history
- Keyboard shortcuts for power users

Testing & Quality:

- Unit tests for all Simplicate integrations
- Integration tests for chat flows
- Load testing for concurrent users
- Security audit of API access
- User acceptance testing with internal team

LoRA Training

3) Own LoRA Training

Vision: Empower brands to train custom LoRA models using their own visual assets, creating truly brand-specific image generation that goes beyond generic AI outputs.

Why This Matters:

- Generic AI models can't capture unique brand aesthetics
- Brands have specific illustration styles, photo treatments, design patterns
- Training your own LoRA = brand consistency at scale
- Competitive advantage: your brand's AI can't be replicated

Technical Considerations:

- Requires significant GPU resources (A100/H100 ideally)
- Training time varies: 30 mins to 2+ hours depending on dataset size
- Storage for training data, checkpoints, and final models
- Model versioning and A/B testing infrastructure
- Integration with existing image generation pipelines (Nanobanana MCP)
- Consider cloud vs on-premise GPU infrastructure

What is LoRA?: LoRA (Low-Rank Adaptation) is a technique for fine-tuning large AI models efficiently. Instead of retraining the entire model, LoRA trains small "adapter" layers that modify the model's behavior. Think of it as teaching the AI your brand's specific style without starting from scratch.

User Journey:

1. Brand uploads 20-50 reference images (logos, product photos, illustration examples)
2. System validates images (resolution, format, quality)
3. User configures training parameters (trigger words, style focus)
4. Training job is queued and processed
5. User receives notification when complete
6. User tests model with sample generations
7. User approves and deploys to production OR iterates with more training

Tickets to Create:

Research & Planning:

- Research LoRA training best practices and parameter recommendations
- Evaluate GPU infrastructure options (RunPod, Lambda Labs, AWS, Azure)
- Calculate cost per training job and monthly estimates
- Define minimum/maximum dataset sizes
- Research automatic captioning and tagging tools
- Benchmark training times across different hardware

Dataset Management System:

- Design image upload interface with drag-and-drop
- Implement image validation (format, size, quality checks)
- Build image preview gallery
- Create tagging and categorization system
- Implement automatic image preprocessing (resize, crop, format conversion)
- Build dataset organization (group by style, type, usage)
- Add duplicate detection
- Create dataset versioning for experiments

Training Configuration UI:

- Design training parameter interface (learning rate, steps, batch size)
- Create trigger word input and suggestions
- Add style focus selectors (portraits, landscapes, products, illustrations)
- Build advanced settings panel for power users
- Implement training presets (quick, balanced, detailed)
- Add estimated time and cost calculator

Training Infrastructure:

- Set up GPU instance provisioning (auto-scale based on queue)
- Implement training queue management system
- Build job scheduling and priority handling
- Create checkpoint saving during training
- Implement training progress monitoring
- Add automatic recovery from failures
- Build cost tracking per training job

Model Testing & Validation:

- Create test generation interface (generate sample images)
- Build side-by-side comparison with base model
- Implement quality scoring and metrics
- Add prompt testing suite (run same prompts before/after)
- Create feedback collection system
- Build iteration workflow (refine and retrain)

Model Deployment:

- Build model versioning system (keep historical models)
- Implement A/B testing infrastructure
- Create production deployment pipeline
- Add rollback capabilities
- Build model performance monitoring
- Implement automatic model updates to generation endpoints

Integration with Existing Systems:

- Connect trained LoRAs to Nanobanana MCP
- Update image generation UI to show LoRA selector
- Add LoRA to social media generation workflow
- Integrate with brand asset library
- Update brand guidelines to reference available LoRAs

Admin & Monitoring:

- Build admin dashboard for all training jobs
- Create cost analytics and reporting
- Add GPU utilization monitoring
- Implement alert system for failed jobs
- Build user quota management
- Create training history and audit logs

Brand Strategy Brainstorm

3. Brand Strategy Brainstorm

Vision: An AI-powered collaborative space where brand teams can explore strategic directions, generate positioning ideas, develop audience personas, and craft brand narratives with AI as a creative partner.

Why This Matters:

- Brand strategy often happens in scattered documents and meetings
- AI can accelerate ideation and provide fresh perspectives
- Centralizing strategy work creates a single source of truth
- Integrates strategy directly with execution (guidelines, assets)

Use Cases:

- Launching a new sub-brand or product line
- Rebranding or refreshing brand positioning
- Developing new audience segments
- Competitive positioning exercises
- Brand architecture mapping
- Messaging hierarchy development

Key Capabilities:

- AI-assisted brainstorming sessions
- Strategic framework templates (Golden Circle, Brand Key, Perceptual Maps)
- Competitive analysis and differentiation
- Audience persona development
- Positioning statement generation
- Brand voice and tone definition
- Visual identity exploration

Tickets to Create:

Research & Discovery:

- Interview brand strategists about their process and pain points
- Research existing brand strategy frameworks and methodologies
- Identify which strategic exercises are most valuable
- Map out typical brand strategy project phases
- Document how strategy connects to brand guidelines
- Research AI-assisted ideation techniques

Strategic Framework Library:

- Build library of strategy templates (Simon Sinek Golden Circle, Brand Pyramid, etc.)
- Create guided workshops for each framework
- Design interactive templates with fillable fields
- Add explanations and examples for each framework
- Build template preview and selection interface

AI Brainstorm Engine:

- Design prompting strategy for brand-aware suggestions
- Integrate with Brand Brain (RAG) for context-aware ideas
- Build suggestion generation for positioning statements
- Create competitive differentiation analyzer
- Implement tone/voice variation generator
- Add "what if" scenario exploration

Positioning Workshop:

- Design step-by-step positioning wizard
- Create audience targeting questionnaire
- Build benefit vs. feature extraction
- Implement reason-to-believe generator
- Add positioning statement templates
- Create multiple positioning variations for testing

Persona Generator:

- Design persona creation workflow
- Build demographic input interface
- Create psychographic questionnaire
- Implement AI-generated persona narratives
- Add persona image generation (using LoRA)
- Build persona card templates
- Create persona gallery and comparison

Competitive Analysis:

- Design competitor input interface
- Build perceptual mapping tool
- Create SWOT analysis templates
- Implement gap analysis visualizations
- Add competitive positioning suggestions
- Create differentiation opportunity highlights

Brand Voice Development:

- Design tone of voice spectrum sliders

- Create example phrase generator for each tone
- Build do's and don'ts generator
- Implement voice comparison with competitors
- Add sample copy in different tones
- Create voice and tone guide export

Collaborative Features:

- Build real-time collaboration (multiple users in same session)
- Add commenting and feedback on ideas
- Implement voting and ranking mechanisms
- Create version history for strategy iterations
- Add export to presentation format
- Build sharing and permission controls

Export & Documentation:

- Create branded PDF export templates
- Build presentation slide generator
- Implement brand strategy brief template
- Add executive summary generation
- Create integration with brand guidelines
- Build strategy archive and searchability

Integration Points:

- Connect strategy insights to Brand Brain
- Use positioning to inform AI tone of voice
- Link personas to content calendar targeting
- Feed visual direction to LoRA training
- Connect to brand guidelines documentation

Push Notifications

4. Push Notifications

Vision: Keep users informed and engaged with timely, relevant notifications about brand updates, workflow completions, approvals needed, and team collaboration.

Why This Matters:

- Reduces time checking for updates
- Accelerates approval workflows
- Improves team coordination
- Keeps brands on track with deadlines
- Provides visibility into AI generation progress

Notification Types:

Brand Updates:

- Brand guidelines updated
- New brand assets uploaded
- LoRA training completed
- Brand compliance issues detected

Workflow & Approvals:

- Content pending your review
- Asset approved/rejected
- Your approval requested
- Deadline approaching
- Task assigned to you

AI Generations:

- Image generation complete
- Video processing finished
- LoRA training done
- Batch content generation ready

Collaboration:

- Mentioned in comment
- Reply to your comment
- Shared item with you
- Team member joined brand
- Strategy session scheduled

System:

- System maintenance scheduled
- Feature updates available
- Usage limits approaching
- Subscription changes

Delivery Channels:

- In-app notifications (toast, banner, notification center)
- Email (via Resend.com)
- Browser push notifications
- Optional: SMS for critical items
- Optional: Slack/Teams webhooks

Tickets to Create:

Architecture & Infrastructure:

- Design notification event architecture
- Choose notification service/queue system
- Plan database schema for notifications
- Design retry and failure handling
- Create notification routing logic
- Plan real-time delivery mechanism (WebSocket, SSE, polling)

Notification Service Core:

- Build notification creation API
- Implement notification queue system
- Create delivery worker processes
- Build retry logic with exponential backoff
- Implement dead letter queue for failures
- Add notification deduplication
- Create notification batching (multiple events → one notification)

In-App Notification Center:

- Design notification center UI (sidebar, modal, or popover)
- Build notification list component
- Implement unread badge counter
- Create notification item components (different types have different layouts)
- Add mark as read/unread functionality
- Implement bulk actions (mark all read, delete)
- Build notification filtering (by type, date, brand)
- Add notification search

Toast & Banner Notifications:

- Build toast notification component
- Create banner notification for important items
- Implement auto-dismiss timing
- Add pause-on-hover functionality
- Create toast queue management (don't show 20 toasts at once)
- Build action buttons in toasts (Quick view, Dismiss, Undo)

Email Notifications:

- Set up Resend.com integration
- Design email notification templates (transactional style)
- Build email rendering system
- Create email preference center
- Implement unsubscribe handling
- Add email tracking (opens, clicks)
- Build digest emails (daily/weekly summary)

Browser Push Notifications:

- Implement Web Push API
- Create push notification service worker
- Design permission request flow
- Build push notification management
- Add push notification preferences
- Handle notification clicks and navigation

User Preferences:

- Design notification settings interface
- Build per-channel toggles (email, push, in-app)
- Create frequency controls (instant, digest, off)
- Implement per-notification-type preferences
- Add quiet hours configuration
- Build per-brand notification settings
- Create role-based default preferences

Notification Triggers:

- Implement brand guideline update triggers
- Add asset upload/update triggers
- Create workflow state change triggers
- Build AI generation completion triggers
- Add collaboration event triggers (mentions, comments)
- Create scheduled notification triggers

- Implement custom alert rules

Admin & Analytics:

- Build notification analytics dashboard
- Track delivery rates and failures
- Monitor user engagement with notifications
- Create notification debugging tools
- Add notification testing interface
- Build notification audit logs

Integration Points:

- Connect to all content generation workflows
- Integrate with approval systems
- Link to collaboration features
- Connect to LoRA training pipeline
- Integrate with Brand Brain updates

Brand brain + RAG

5. BRAND BRAIN + RAG

Vision: Create an intelligent "brain" for each brand that understands their guidelines, history, preferences, and context. Using Retrieval-Augmented Generation (RAG), the AI can provide accurate, brand-specific responses by retrieving relevant information before generating content.

Why This Matters:

- Generic AI doesn't know your brand's specific guidelines
- RAG grounds AI responses in actual brand documentation
- Reduces hallucinations and off-brand suggestions
- Creates institutional knowledge that persists
- Enables more sophisticated AI features

What is RAG?: Retrieval-Augmented Generation combines information retrieval with AI generation. Instead of relying solely on the AI's training:

1. User asks a question or requests content
2. System retrieves relevant brand documents/context
3. AI generates response informed by those specific documents
4. Result: brand-accurate, contextual responses

Brand Brain Contains:

- Brand guidelines (written documents)
- Visual identity standards (colors, fonts, logos)
- Tone of voice examples and rules
- Previous content and campaigns
- Product information and specifications
- Target audience descriptions
- Competitor information
- Brand history and evolution
- Do's and don'ts
- Success metrics and KPIs

Technical Architecture:

- Vector database for semantic search (Supabase Vector, Pinecone, or Weaviate)
- Embedding model for converting text to vectors
- Document processing pipeline
- Chunking strategy (how to split large documents)
- Metadata tagging and filtering
- Query processing and reranking
- Context assembly for AI prompts

Tickets to Create:

Research & Architecture:

- Research vector database options and capabilities
- Choose embedding model (OpenAI, Cohere, open-source)
- Design document chunking strategy
- Define metadata schema for brand documents
- Plan query processing pipeline
- Research reranking algorithms
- Benchmark search quality and performance
- Plan for multi-brand isolation and security

Vector Database Setup:

- Set up vector database instance
- Configure indexes and collections
- Implement connection pooling
- Build database backup strategy
- Create monitoring and alerting
- Plan scaling strategy
- Set up development and staging instances

Document Ingestion Pipeline:

- Build document upload interface
- Implement file parsing (PDF, DOCX, TXT, MD, HTML)
- Create text extraction from images (OCR)
- Build document chunking logic
- Implement embedding generation
- Create metadata extraction
- Add document deduplication
- Build batch processing for large uploads

Brand Content Processing:

- Parse brand guideline documents
- Extract structured data (colors, fonts, logos)
- Process tone of voice examples
- Import existing content library
- Extract visual assets metadata
- Process competitor information
- Import audience personas
- Handle brand history timelines

Semantic Search Engine:

- Build query processing and cleaning
- Implement query embedding generation
- Create vector similarity search
- Add metadata filtering
- Implement hybrid search (keyword + semantic)
- Build result reranking
- Add relevance scoring
- Create search result explanation

Context Assembly:

- Design prompt template system
- Build context retrieval for different query types
- Implement context window management (token limits)
- Create context summarization
- Add multi-document synthesis
- Build citation tracking (which docs were used)

Brand Brain API:

- Create query endpoint for Brand Brain
- Build content generation endpoints with RAG
- Implement brand question answering
- Add guideline checking with context
- Create brand-aware suggestion API
- Build confidence scoring

Admin Interface:

- Design Brand Brain management dashboard
- Build document management UI
- Create index monitoring and health checks
- Implement search quality testing interface
- Add manual document upload
- Build bulk update capabilities
- Create document version history

Integration with Existing Features:

- Enhance text generation with brand context
- Improve tone of voice checking with examples
- Enrich image generation prompts with brand info
- Power strategy brainstorm with brand history
- Inform chatbot responses with brand knowledge
- Guide agents with brand guidelines

Quality & Testing:

- Build search quality evaluation framework
- Create test query datasets
- Implement A/B testing for retrieval strategies
- Add relevance feedback loops
- Build continuous quality monitoring
- Create brand knowledge coverage reports

Auto-Update Mechanisms:

- Implement automatic guideline sync when updated
- Add incremental indexing for new content
- Create automatic re-embedding on model updates
- Build change detection and delta updates
- Add scheduled full re-indexing

Agents Powered Workflows

6. Agents Powered Workflows

Vision: Deploy autonomous AI agents that can execute complex, multi-step brand management tasks with minimal human intervention. These agents understand brand context, use tools, make decisions, and coordinate to achieve goals.

Why This Matters:

- Automates repetitive brand management tasks
- Handles complex workflows that require multiple steps
- Reduces time from brief to finished content
- Scales brand operations without scaling headcount
- Provides 24/7 brand management capabilities

What are AI Agents?: Unlike simple AI tools that respond to single prompts, agents can:

- Plan multi-step approaches to complex goals
- Use tools and APIs to take actions
- Make decisions based on intermediate results
- Retry and adapt when things go wrong
- Coordinate with other agents
- Learn from outcomes

Example Agent Workflows:

Campaign Launch Agent:

1. User provides campaign brief
2. Agent generates content strategy
3. Creates multiple content variations
4. Checks all content against brand guidelines
5. Generates visual assets with LoRA
6. Compiles campaign package
7. Requests approval with explanations
8. Publishes approved content

Potential tickets to Tickets to Create:

Research & Architecture:

- Research agent frameworks (LangGraph, CrewAI, AutoGPT, custom)
- Design agent architecture and communication patterns
- Define agent capabilities and limitations

- Plan tool integration strategy
- Design agent state management
- Research agent coordination patterns
- Plan observability and debugging approach
- Define success metrics for agents

Agent Framework Implementation:

- Set up chosen agent framework
- Build agent execution engine
- Implement task planning and decomposition
- Create tool/function calling system
- Build agent state persistence
- Implement error handling and recovery
- Create agent coordination layer
- Add agent-to-agent communication

Tool Integration Layer:

- Create tool registry and discovery
- Build tool execution wrappers
- Implement tool authentication
- Add tool rate limiting
- Create tool result parsing
- Build tool error handling
- Add tool usage logging
- Create tool testing framework

Content Creation Agent:

- Design content creation workflow
- Implement brief analysis
- Build content structure generation
- Create multi-variation generation
- Add brand guideline checking
- Implement iterative refinement
- Build quality assessment
- Add human-in-the-loop approval points

Brand Compliance Agent:

- Design compliance checking workflow
- Build asset discovery and scanning
- Implement multi-criteria checking
- Create issue prioritization
- Generate remediation suggestions

- Build compliance reporting
- Add tracking over time
- Create compliance scoring

Campaign Orchestration Agent:

- Design campaign workflow
- Build brief parsing and understanding
- Implement strategy generation
- Create multi-asset coordination
- Add timeline management
- Build approval workflow integration
- Implement publishing coordination
- Add campaign performance setup

Asset Generation Agent:

- Design generation workflow
- Implement prompt optimization
- Build batch generation
- Add variant creation
- Create quality filtering
- Implement brand checking
- Add revision handling
- Build asset organization

Research & Analysis Agent:

- Design research workflow
- Build information gathering
- Implement source verification
- Create synthesis and summarization
- Add insight extraction
- Build report generation
- Implement citation tracking

Agent Monitoring Dashboard:

- Design agent status interface
- Build real-time activity monitoring
- Create task history and logs
- Implement performance metrics
- Add cost tracking per agent
- Build error and failure analysis
- Create agent health indicators
- Add agent comparison analytics

Workflow Templates:

- Create template library interface
- Build workflow definition format
- Implement template customization
- Add template sharing
- Create template versioning
- Build template marketplace (internal)
- Add usage analytics per template

Human-in-the-Loop Controls:

- Design approval checkpoint UI
- Build notification for agent requests
- Create decision interface
- Implement feedback collection
- Add override capabilities
- Build audit trail
- Create delegation settings

Safety & Guardrails:

- Implement action confirmation for critical operations
- Build spending limits per agent
- Create resource usage caps
- Add approval requirements for sensitive actions
- Implement rollback capabilities
- Build emergency stop functionality
- Create agent behavior monitoring

Testing & Quality:

- Build agent testing framework
- Create simulation environments
- Implement regression testing
- Add performance benchmarking
- Build quality scoring
- Create A/B testing for agent strategies
- Add continuous improvement feedback loops

ARCHIVE

Development

Objective: To develop the best approach for rebuilding modules in the next environment.

We can create a plan using Claude.MD with relevant tools required

- https://ai-sdk.dev/docs/introduction
- Rules
- Design elements

From this plan we can ask for the sprint plan which will divide up our tasks which, so we haver

MCP

Objective: To understand impact of having MCP powered agents in our systems

Server	**link**	**why**
Supbase MCP	https://supabase.com/docs/guides/getting-started/mcp	Can perform tasks like launching databases, managing tables, fetching config, and querying data on your behalf.
Github MCP		To aid the superbase MCP with debugging or carrying out code reviews
Simplicate - MCP	https://github.com/daanno/simplicate-mcp	To aid the account team
Veo3	https://github.com/dayongd1/mcp-veo3	We can use these services as and when needed to generate videos, images
Nanobanana		
Sora 2		
https://resend.com/	https://resend.com/docs/knowledge-base/mcp-server	Equipping the LLM agent with this could help with debugging a a lost password at any point or even having the assets email to a person for sign off.

IF we create our own MCP servers - just have to host them, and expose them so the server n8n can access them

Tasks:

Explore the cleanest way to host our own MCP servers

2.0

Back end

- AI SDK
- Database ??

Front end

- **Figma MCP**
- Dwayne Paisley-Marshall: This is https://www.youtube.com/watch?v=6G9yb-LrEqg what ... | Technology Team 🦾 | Microsoft Teams
-

Features

- Train a lora
- AI agents - generate image - video - voice?

Brand checker elements

- Video
- Copy
- Tone of voice

Technical element
> RAG?

AI agents integration

- QA agent
- Code base agent
- Chat bot agent

Proposed features

New features	Technical set up
2) **Own lora training**	https://www.baseten.co/ (current in beta) https://docs.langflow.org/
5) BRAND BRAIN + RAG	https://docs.langflow.org/mcp-server
6) Agents powered workflows	N8n or langflow
7 Node builder?	https://reactflow.dev/
1) Chatbot MCP tooling	https://mcp.so/
4) Push notifications	https://pushalert.co/
3) brand strategy brainstorm -	

1. Chatbot MCP tooling - Simplicate MCP for now
2. Own lora training
3. Brand strategy brainstorm
4. Push notifications
5. BRAND BRAIN + RAG
6. Agents powered workflows
7. Node builder?

Design & UX

Objective: Transform brand guidelines into a dynamic, user-friendly digital experience across all touchpoints.

First Steps:

- Translate existing brand guidelines into actionable UX and UI principles.

- Use Lovari (https://www.lovari.ai/) to visualize user journeys and identify friction points.

Next Steps:

1. Map the complete user journey across all major touchpoints.

2. Create UX flow diagrams for key interactions and success paths.

3. Develop reusable UI components using shadcn/ui MCP components (https://ui.shadcn.com/docs/mcp).

AI agents -

Objective: Define and implement specialized AI agents to handle targeted workflows.

Agent Type	Primary Function
Social Media Agent	Generates and schedules brand-aligned social content.
Video Agent	Creates and reviews brand-compliant video assets.
General Agent	Responds to client and internal queries using contextual data.
Account Agent	Provides structured insights and reporting through MCP.
Brand Checker Agent (RAG)	Ensures compliance with client brand guidelines across all assets.

Database

Objective: To understand the current database backend to how functions and tools are being handled.

New features:

- To implement a production and staging testing env, which can be done using a .env file.

Open questions:

How do we allow for different brands to have their own integration of ACT.onbrand

Do we want to allow our clients to host and upload all of their videos on our superbase?

How do we allow for videos and images to be compressed when uploaded to save on storage space.

Do we want to consolidate all our API calls so they're handled by agents' workflows and end points.

Next Steps:
- Review and remove redundant database tables.
- Confirm Supabase as the preferred backend for ACT 2.0.
- Maintain existing email resend service integration.

R&D

- Own Supabase

1.0

Current features

The architecture for ACT 2.0 will be a modern, scalable stack designed for AI-native applications. Based on the documentation and the GitHub repository, the core infrastructure will consist of the following components:

- **Frontend Framework:** The application will be built using **React**, likely within the **Next.js** framework, as indicated in the repository. This provides a robust foundation for server-side rendering, static site generation, and a seamless developer experience.

- **Backend-as-a-Service (BaaS): Supabase** will serve as a primary backend service (MCP). Its responsibilities will include:
    - **Database:** PostgreSQL for storing user data, brand configurations, and generated content.
    - **Authentication:** Managing user logins and permissions through its built-in auth system.
    - **Storage:** Hosting brand assets, AI-generated images, and other files.
    - **Serverless Functions:** For custom backend logic and secure API integrations.

- **AI Integration Layer:** The **Vercel AI SDK** is the designated framework for connecting to and interacting with large language models. This is crucial for building streaming chat interfaces and ensuring a responsive user experience for features like text generation and the `CHAT module`.

- **UI/UX Component Library:** The team will research and select a reusable component library, with **`shadcn/ui`** being a primary candidate. This will help enforce design consistency and accelerate development by translating brand guidelines into actionable UX and UI principles.

- **Version Control & Deployment: GitHub** will be used for source code management. All development will follow established Git workflows, and CI/CD pipelines (likely via GitHub Actions or Vercel) will automate testing and deployment.

- **Email & Notifications: Resend** is the selected service for handling transactional emails. This will be used for user invitations, system alerts, and potentially the new **Push notifications** feature.

V2 Feature Implementation Strategy

The new features for V2 represent a significant leap in AI capability. Here’s a brief on the technical approach for the key features:

- **BRAND BRAIN + RAG (Retrieval-Augmented Generation)**: This is the core AI feature. The plan involves creating a vector database (likely using `pgvector` within Supabase) from each client's brand documentation. When a user prompts an AI agent, the RAG system will first retrieve the most relevant sections of the brand guidelines and feed them to the language model as context. This ensures all generated content (text, strategy ideas, etc.) is deeply aligned with the brand.

- **Agents Powered Workflows**: Building on the "BRAND BRAIN," these agents will be AI systems capable of performing multi-step tasks. For example, a "Campaign Agent" could take a brief, use the RAG system to access brand guidelines, generate social media copy , create brand-aligned images (potentially using custom-trained models), and schedule them in a content calendar. A `Node builder` is also being considered to potentially allow for custom workflow creation.

- **Own LORA Training**: This moves beyond generic AI image generation. The plan is to provide a service where we can fine-tune image models (like Stable Diffusion) using a brand's specific visual assets. This will allow the AI to generate `LORA Illustrations` and other visuals that perfectly match a unique brand aesthetic, not just brand colors or fonts.

- **Chatbot MCP Tooling**: This feature will focus on providing a simplified toolset for creating MVP-level chatbots. The initial integration target for this is **Simplicate**. This suggests the chatbot could pull data from or push data to the Simplicate CRM/project management system.

Design and Development Workflow

Design Process:

1. **Discovery:** The first step is to map out all current user journeys to identify pain points and opportunities for improvement.
2. **Systemization:** Translate existing brand guidelines into a concrete set of UX and UI principles.
3. **Prototyping:** Create low-fidelity UX flow diagrams to validate the new user flows before any code is written.

Development Process (Agile Sprint-Based):

- **Sprint Planning:** The team will use **Monday.com** for creating and tracking all tickets. Sprints will be two weeks long.
- **Estimation:** Tickets will be estimated using the Fibonacci sequence, with a target velocity of **30 story points per sprint**. This will help forecast progress and communicate any potential delays against the January 2026 deadline.
- **Development Environment:** To ensure consistency, all team members must sync their **Cursor settings**. This will align the development context and the rules provided by `cursor.directory` and Supabase.
- **Production Safety:** The project will utilize tools like **`bugbot`** to protect the production environment from regressions and bugs.

Core Technology Stack

The foundation of ACT 2.0 is a modern, AI-native technology stack chosen for scalability and developer experience.

- **Framework**: The application will be built using **React**. The GitHub repository indicates this will be implemented within the **Next.js** framework, which is ideal for building high-performance, server-rendered applications that can seamlessly integrate AI features.
- **AI Integration**: The **Vercel AI SDK** is the designated tool for connecting the frontend to various language models. This is critical for building responsive, streaming interfaces for features like the `CHAT module` and `Text Generation`.

- **Backend and Database**: **Supabase** will serve as the primary backend platform. It will provide the PostgreSQL database, user authentication, file storage for assets, and serverless functions for custom logic, creating a unified backend environment.
- **UI Component Library**: The team will use a reusable component library to ensure design consistency and speed up development. The primary candidate for this is `shadcn/ui`, which allows for building a custom, accessible design system from scratch. This will be used in conjunction with **Tailwind CSS**.

To ensure a consistent and efficient workflow, the team will standardize its development environment and tools.

- **Code Editor**: The team will use **Cursor**, an AI-first code editor. It's crucial that all developers **sync their Cursor settings** to ensure everyone is working from the same AI context and development rules.
- **Version Control**: **GitHub** will be used for all source code management and version control.
- **Project Management**: All sprints, tasks, and tickets will be created and tracked using **Monday.com**.

A structured workflow and clear coding standards are essential for managing the project's complexity and meeting the January 2026 deadline.

- **Development Rules**: All code, whether written by a developer or generated by AI, must adhere to predefined standards. The team will follow the rules outlined by **Cursor for React** and the prompts from the **Supabase UI AI editor rules**.
- **Sprint Methodology**: The project will follow a two-week sprint cycle.
  - **Estimation**: Tasks will be estimated using the **Fibonacci number sequence** to assign story points.
  - **Sprint Capacity**: The team aims for a velocity of approximately **30 story points per sprint**.
- **CI/CD & Production Safety**: A continuous integration and deployment pipeline will be established, likely using GitHub Actions or Vercel. To safeguard the live application, the project will incorporate a tool like `bugbot` for production protection.

This phase focuses on setting up the tools, processes, and shared understanding required for the project.

- **1. Create the Project Plan Markdown File:**
  - The first action item is to create a markdown (`.md`) file detailing the project plan. This file should be committed to the GitHub repository to give the AI an understanding of the project's scope.
- **2. Set Up Project Management Board:**
  - Set up the project in **Monday.com** to be used for tracking all tickets and tasks.
- **3. Align Development Environment:**
  - Schedule a mandatory meeting for the development team to **sync their Cursor settings**. This is essential for ensuring everyone is developing from the same context.
- **4. Train the Team on Estimation:**
  - Ensure everyone on the team understands how to estimate tasks using the **Fibonacci number sequence**. This is required for your sprint planning process.

This phase is dedicated to the detailed planning and design groundwork before the first development sprint begins.

- **1. Breakdown Technical Elements into Tickets:**
  - As a team, conduct a planning session to **break down all technical elements into sprint-based tickets**. Populate the Monday.com backlog with these tickets, keeping the **January 2026 timeline** in mind.
- **2. Map Current User Journeys:**
  - The design/UX lead should begin **mapping out the current user journeys** of the existing platform. This will inform the new design and identify areas for improvement.
- **3. Define UX/UI Principles:**
  - Start the process of **translating the existing brand guidelines into actionable UX and UI principles**. This will be the foundation for the new component library and overall design.

With the groundwork laid, we can now begin our first two-week development sprint.

- **1. Hold First Sprint Planning Meeting:**
  - Review the ticket backlog in Monday.com together as a team.

    - Estimate the tickets for the upcoming sprint and decide what work will be done over the next two weeks.
- **2. Begin Low-Fidelity Design:**
    - A key goal for the design team in the first sprint should be to **create low-fidelity UX flow diagrams** for the highest-priority V2 features.
- **3. Technical R&D:**
    - Assign a development ticket to formally evaluate the reusable UI libraries (like `shadcn/ui` and `kibo-ui`) to decide which is best for your specific needs.