ChatGPT

Agency Social Media Automation Platform – Architecture & Plan

Project Overview

This project entails building an **AI-driven social media automation platform** for a marketing agency. The platform will manage clients' social media accounts on their behalf without giving clients direct access. It aims to streamline content creation, scheduling, posting, and engagement (likes, comments, direct messages) using advanced AI (LLM agents) and integrations. **Key objectives** include:

- Generating high-quality social media content (text captions, possibly images) with AI assistance.
- Scheduling and publishing posts across multiple platforms (e.g. Instagram, Facebook, LinkedIn, Twitter/X, TikTok, Google Business) via a unified system.
- Automating engagement tasks such as replying to comments, liking relevant posts, and handling direct message (DM) conversations through AI chat.
- Integrating with existing tools (like GoHighLevel CRM) where appropriate to leverage their capabilities (social scheduler, unified inbox, etc.), rather than reinventing the wheel for every feature.

The development will be **"vibe-coded"** – meaning an iterative, AI-assisted coding approach. We will employ CLI agents and an LLM co-pilot (such as Anthropic's Claude via the Claude Agent SDK) throughout the development process. This allows the team to rapidly prototype and build features with AI generating and refining code interactively [1]. The end result should be a robust internal platform that our agency team can use to manage clients' social presence efficiently, with minimal manual effort.

Team Composition and Roles

Given **unlimited human resources**, we will assemble a multidisciplinary team. Each member's role, duties, and skills are outlined below:

- **Project Manager / Product Owner** – Oversees the project roadmap and ensures the solution meets agency needs. Gathers requirements (e.g. what content and engagement actions clients need), prioritizes features, and coordinates between technical and non-technical teams. Skills: communication, agile methodology, domain knowledge in social media marketing.

- **Lead Solution Architect** – Designs the overall system architecture (frontend, backend, integrations, AI components). Decides how components interact (e.g. Next.js app with backend services, integration with GoHighLevel or direct APIs). Ensures the system is scalable, secure, and maintainable. Skills: software architecture, cloud infrastructure, API design, security best practices.

- **Full-Stack Developers** – Implement the web application (frontend in Next.js and backend services). One or more specialists here:

- *Frontend Developer*: Focused on Next.js UI, creating dashboards/calendars for scheduled posts, content editors with AI suggestions, etc. Skills: React/Next.js, TypeScript, UI/UX design basics.
- *Backend Developer*: Builds the serverless functions or API server that handle business logic – e.g., calling social media APIs, scheduling posts, storing data. Skills: Node.js/TypeScript (or Python), database design, integration of third-party APIs.

- **AI/ML Engineer** – Integrates and fine-tunes the **LLM agents**. This person sets up the Claude Agent SDK, designs the prompts and "tools" the agent will use, and ensures the AI outputs are relevant. They might train custom models or at least craft prompts so that content matches clients' brand voice. Skills: prompt engineering, familiarity with LLM APIs (Anthropic Claude, OpenAI, etc.), data analysis for fine-tuning.

- **DevOps & Cloud Engineer** – Manages deployment and infrastructure. Sets up Vercel for hosting the Next.js app and any additional servers or cloud functions needed for the agent or automations. Ensures continuous deployment, monitoring, and that development can use CLI agents safely (sandboxing as needed). Skills: Vercel platform, CI/CD, Docker, cloud services, security (especially for handling API keys/tokens securely).

- **Integration Specialist** – (Optional, given unlimited resources) Focuses on connecting with **social media APIs** and any external platforms. For example, this specialist ensures seamless OAuth connections to Facebook/Instagram Graph API, Twitter API, LinkedIn API, etc., and also integration with GoHighLevel's API. They keep up with platform policy changes and ensure our use of APIs (for auto-posting, commenting, messaging) stays compliant. Skills: REST APIs, OAuth flows, knowledge of social network developer policies.

- **QA/Test Engineer** – Responsible for testing the platform's functionality. This includes verifying that scheduled posts actually publish correctly on each platform, that the AI-generated content is accurate and appropriate, and that automation (likes/comments/DM responses) behaves as expected without errors. Skills: test automation, scenario testing, understanding of AI output evaluation.

- **Social Media Strategist (Advisor)** – Although not a coder, having a strategist in the loop ensures the content and engagement tactics align with marketing best practices. They can provide guidelines for the AI (e.g. tone, hashtags, timing for posts) and review AI outputs initially. Skills: social media marketing, copywriting, brand strategy.

Each team member can collaborate closely, especially given the "vibe coding" approach where developers and the AI co-pilot work hand-in-hand. The **unlimited resource** assumption means we are not constrained by headcount – so we can have highly specialized roles, ensuring each aspect (AI, integrations, frontend, etc.) has expert attention.

Development Approach: *Vibe Coding* with AI Agents

We will leverage a **vibe-coded development workflow**, meaning the team will use AI co-pilots and CLI-based agent tools to speed up and enhance development. In practice, this involves using the **Claude Agent**

SDK and possibly tools like Anthropic's Claude Code CLI or Google's Gemini CLI to interactively generate and refine code.

Developers will "pair-program" with a highly capable LLM: writing prompts or high-level instructions and letting the AI agent propose code, run it, debug, and iterate. This approach has been shown to accelerate building apps. For example, using a conversational IDE assistant alongside an agentic CLI allows rapid prototyping – one engineer **"vibe coded a complete travel app"** by having the CLI generate a spec, then incrementally implementing it with the AI's help [2]. We will adopt a similar strategy:

- **Claude Agent SDK for Automation**: The Claude Agent SDK (formerly Claude Code SDK) provides an agent loop that can write and execute code, access files, and utilize tools [3]. By giving Claude "a computer" through this SDK, the AI can perform many development tasks autonomously [4] – from scaffolding Next.js pages to setting up API calls. The SDK's design lets Claude operate in a loop of *gather context -> take action -> verify work -> repeat* [5], much like a junior developer who continuously tests their code.

- **CLI-Based Workflow**: Team members will use CLI agents during development (e.g., running `claude-code` or similar in a terminal) to generate boilerplate and even execute it. For instance, we can prompt the agent to "create a Next.js project with a login page and dashboard", and it will create the files and run `npm` commands. The human developer then reviews and fine-tunes the output. This tight feedback loop continues for each feature. It not only speeds up coding but also helps catch errors early – the AI can run unit tests or point out bugs as it writes code.

- **Agent Tools and MCP**: Using the Agent SDK, we can give the AI **tools** for specific actions – for example, a tool to call the Facebook Graph API, or a tool to post via GoHighLevel's API. Tools are prominent in Claude's context, so it will intelligently choose them for tasks when coding [6]. We may also implement **MCP (Model Context Protocol)** servers for structured external actions. *MCP servers act as API proxies for the LLM* [7] – meaning instead of letting the AI craft arbitrary HTTP calls (which could be error-prone), we provide predefined endpoints. For instance, an "Instagram MCP server" could expose a simple API like `postToInstagram(content)` for the agent to call internally, and that server handles the actual Graph API call safely. This gives us fine control and security (the AI doesn't directly handle secrets), while still allowing flexibility in automation. Anthropic's SDK supports extending with custom tools and even sub-agents, which we will leverage for complex tasks (e.g., a sub-agent focused only on monitoring incoming DMs while another focuses on content scheduling) [8].

Overall, vibe coding with an AI co-pilot will infuse the development process with speed and creativity. The human developers will guide high-level architecture and verify outputs (ensuring quality and security), while the AI can handle a lot of the grunt work. This approach aligns with the idea that **"writing the code is the easy part now"**, and the real focus is on higher-order decisions [9] – which our team will concentrate on, using AI to handle repetitive coding tasks.

Technical Architecture and Stack

Platform & Hosting

We will build the web application using **Next.js** (React framework) and host it on **Vercel**, as initially planned. Next.js is a solid choice for this project because it provides a great developer experience, supports serverless API routes (for our backend logic), and can easily integrate with third-party APIs. Vercel hosting offers instantaneous deployments, automatic scaling, and edge network for fast global access. Many developers in the vibe-coding community deploy their projects to Vercel for the web frontend [10], which affirms this choice for a modern web app.

Next.js Frontend: The frontend will serve as an internal dashboard for the agency team. It will likely include: - A **Client Management** section – list of client accounts, with their connected social profiles and settings. - **Content Calendar/Planner** – a calendar view showing scheduled posts for each client (similar to HighLevel's Social Planner calendar). - **Post Creation interface** – a form or editor to draft content. Here we will integrate AI assistance: e.g., a "Generate Caption" button that calls our LLM to produce suggested text, or an image generation tool for creatives if needed. - **Engagement Console** – a view to monitor recent activity on clients' social accounts (recent comments, DMs, etc.), possibly with an AI summary or suggestions on replies. Staff can supervise the AI's engagement here. - **Settings/Integrations** – for connecting or configuring social accounts, API keys (if needed), and toggling what automated actions are enabled per client (some clients might opt out of auto-DMs, for example).

The Next.js app will be primarily **client-side for UI** and use SSR/ISR as needed for performance. Sensitive operations (like actually posting content or responding to a comment via API) will be handled in **Next.js API routes (serverless functions)** or delegated to backend services, to keep secrets out of the client side.

Backend & Services: While Next.js can handle backend logic through its API routes (running on Vercel's serverless infrastructure), we may introduce additional services for long-running or specialized tasks: - An **Automation Orchestrator Service** (could be a Node.js or Python service running on a server or container) to manage continuous background tasks. For instance, listening to webhooks from social platforms for incoming DMs or comments is a long-running process – this doesn't fit well in a short-lived serverless function. We might deploy a small service (perhaps on a VM or Heroku/Fly.io, or even use something like a Cloud Run container) that stays online to handle real-time events and coordinate the AI responses. This service would use the Claude Agent SDK to spin up the AI agent when needed for an event (or keep an agent running with streaming responses). - A **Database** to store application data – likely a cloud database (PostgreSQL or MongoDB, etc., depending on preference). We'll need to persist client info, social account tokens, scheduled posts, content drafts, logs of AI conversations (for accountability), etc. Vercel can integrate with external DBs easily (we might use a hosted DB like Supabase or Planetscale). This ensures that even if we leverage GoHighLevel for some storage (like contacts or conversations), we have our own record of actions and content.

Claude Agent Integration (Automation Brain)

At the core of our system's automation capabilities will be an **LLM agent powered by Claude**. Using the Claude Agent SDK, we effectively "embed an AI co-worker" into our application. This agent can handle a variety of tasks autonomously or semi-autonomously: - **Content Generation**: When creating social posts, the agent can suggest captions or even complete posts given some context (e.g. client industry, campaign, recent news). We can prompt Claude with a structured instruction like "Write an engaging Facebook post for [Client], who is a [business type], promoting [offer] with a friendly tone." Claude's strength in understanding instructions should yield quality copy. - **Automated Engagement**: The agent can be set up to monitor inputs (new comments, messages) and formulate responses. For example, if someone comments "This looks great!" on our client's post, the agent could draft a polite reply thanking them. More powerfully, if a user sends a DM asking about pricing, the agent (with access to an FAQ or context file) could respond with relevant info or even book an appointment by interacting with calendars. These are dynamic tasks well-suited to an LLM that can understand nuance.

Using Claude in an **agentic loop** means it can plan and execute multi-step workflows on its own. We will configure a **feedback loop** where the agent: **gathers context, takes action, verifies the result, and repeats** as needed [5].

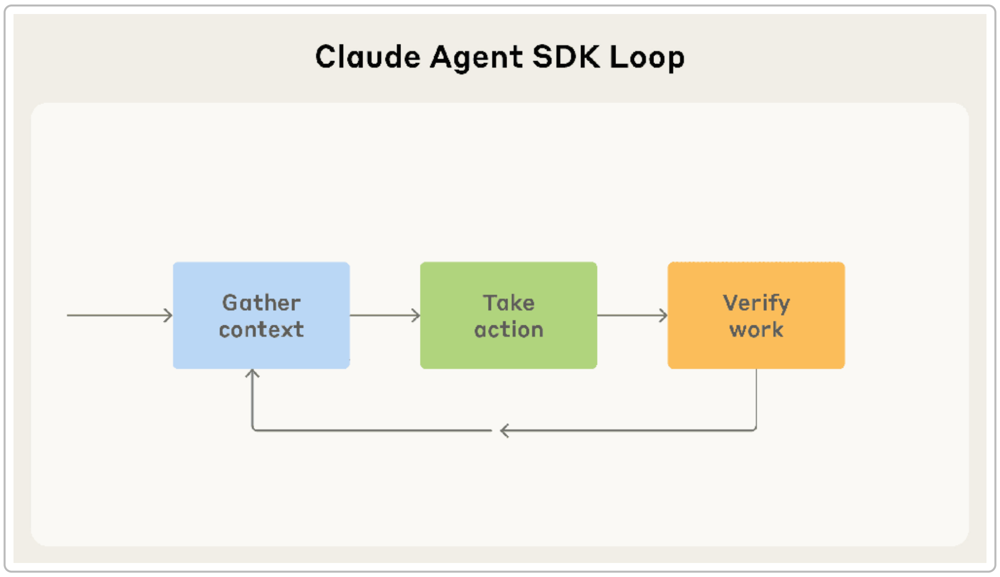

Illustration of an AI agent's iterative loop for problem-solving (Claude Agent SDK): the agent gathers context, takes an action (e.g., calls a tool or API), checks the outcome, and continues the cycle until the task is complete [5].

To enable this, we provide Claude with **tools and permissions**: - **File and Data Access**: The agent might need to read/write certain files or database records. For example, it could read a "content ideas" file or write to a log. The Claude Agent SDK allows defining such file access tools easily [4], essentially *"giving Claude a computer"* to use safely. - **Internet/API Access**: We will register custom tools for the agent to call external APIs. This is where the **MCP integration** is important. Instead of hard-coding API calls, we define tools (or use MCP servers) that the agent can invoke. For instance, a tool called `PostToSocial(network, content)` could be available. Internally that tool could call either the GoHighLevel API (if we use HL for posting) or directly the platform's API. The agent doesn't need to know the low-level details; it just decides *when* to use the tool and with what parameters. This significantly reduces complexity in the agent's prompt. Anthropic notes that **"tools are the primary building blocks of execution for your agent"**, so designing clear, high-level tools will guide Claude effectively [6].

- **Limits & Verification**: We will impose safety checks. For example, Claude's outputs (captions or replies) can be sent in "draft" mode to a human for approval if needed, especially early on. The agent can verify work by checking API responses (did the post actually get scheduled? did the DM send successfully?), and if something fails or looks off, it can adjust (or alert a human). Because we have unlimited team members, we might dedicate some QA or even a moderation layer to review AI-generated content before it goes live, at least until trust in the system is established.

By using the Claude Agent SDK in this manner, we **avoid reliance on third-party automation services** like n8n for orchestration. We essentially build our own orchestration with AI-driven logic. The Claude Agent can manage the workflow of "when X event happens, do Y and Z" in a flexible way that a static n8n workflow might not handle. (For example, instead of a fixed flowchart, the AI can dynamically decide how to handle an unusual customer inquiry in DMs by consulting knowledge base files, composing a custom answer, and only then calling the "send DM" tool.) This approach is powerful, as it leverages Claude's reasoning to handle complex tasks that would otherwise require intricate programming.

Social Media & CRM Integrations

A major aspect is connecting to social media platforms and potentially the GoHighLevel (GHL) CRM platform. We have two strategic options for posting and messaging integration:

Option A: Integrate Directly with Social Platform APIs – We could connect individually to the Meta Graph API (for Facebook/Instagram), Twitter API, LinkedIn API, TikTok API, etc. This gives full control and avoids dependency on any intermediary. However, it requires handling separate OAuth flows for each client & platform, abiding by each platform's rate limits and quirks, and maintaining each integration as APIs change. For tasks like auto-commenting or liking, direct API calls may be the only way (since those are not always exposed in third-party tools). For example, Instagram's API allows replying to comments and sending DMs via a business account; we could subscribe to webhooks for new comments or DMs and have our system respond accordingly. ManyChat's platform demonstrates what's possible – **it can automate responses to comments on posts and send each user a personalized DM** in Instagram [11]. We aim to implement similar capabilities (comment triggers and auto-replies) but within our own system, using AI to personalize responses instead of pre-scripted flows.

Option B: Use GoHighLevel's API for Social & Conversations – GoHighLevel already provides a unified interface to many social platforms. The **HighLevel Social Planner** supports scheduling posts to **major platforms (Instagram, Facebook, LinkedIn, Twitter/X, TikTok, Google Business)** from one place [12]. It also has a unified inbox for messages and comments across channels (in the HighLevel conversations module). HighLevel recently introduced **Content AI** for captions in its Social Planner [13], indicating it can generate copy suggestions too. If HighLevel offers **public APIs for these features** (and indeed, as of late 2025 it has begun exposing Social Planner endpoints [14]), we can offload a lot of heavy lifting to it: - *Scheduling/Posting*: Our app can call HighLevel's API to schedule a post on a certain date/time with given content (and image/video if needed). HighLevel will then handle posting it at the correct time to the respective network, and even track basic metrics. This covers the "content publishing" aspect reliably, including support for multiple networks without us writing separate integration for each. (HighLevel's Social Planner **"supports major platforms with API-based limits"** and can even handle recurring posts and bulk uploads [15] [12].) - *DMs and Comments*: HighLevel's conversation API could allow us to fetch incoming messages or comments and send out replies. For instance, when a new Instagram DM or comment comes in, HighLevel might fire a webhook (or we can poll their API) to get the content. Our AI agent can then generate a reply and we send it back via HighLevel's messaging API. This way, HighLevel deals with authentication and platform delivery, and we just handle the intelligence. If HighLevel supports comment automation (it might not natively auto-reply to comments with AI yet, but we can simulate it by listening for comment notifications and replying via the Graph API ourselves if needed). - *Contacts/CRM*: If needed, our system can push leads or conversations into HighLevel (since it's also our CRM). For example, if someone in a DM wants to book an appointment, our agent could create a contact in HighLevel and add an appointment or task. Using HighLevel's API for these CRM functions would save us from building our own CRM.

We will likely **combine both options** to get the best outcome: - **Use HighLevel where it excels**: For scheduling multi-platform posts and consolidating messages. HighLevel is *"a functional scheduling layer inside a broader platform"* [16] , which is great if we treat our social automation as part of that ecosystem. Since our agency already uses HighLevel, this integration means all social posts and communications could be logged in the same place as other client interactions (calls, emails, etc.), maintaining a unified record. We avoid reinventing features that HighLevel already provides (why rebuild a social calendar and analytics from scratch if HL gives basic versions?). For instance, HighLevel can provide basic engagement metrics for posts (reach, impressions, etc.) [17] which we can display in our app via API, rather than implementing our own analytics pipeline initially. - **Custom integration for unique automation**: For things outside HighLevel's scope or where we need finer control, we'll implement direct calls. An example is **auto-liking posts or following users** for engagement – this is not something HighLevel does, but an agent using the Instagram API could do it (within moderation to avoid spam flags). Another example is if we want to implement ManyChat-style growth hacks (e.g., "reply to this story with a keyword and the bot will send you a resource"), we might directly use the platform webhook and AI, since scripting that through HighLevel could be complex.

To wire this up, the **Integration Specialist** on the team will set up the necessary API connections: - **HighLevel API**: We'll use our agency's HighLevel API keys (and potentially one for each client sub-account) via a secure proxy or MCP tool. HighLevel's API would handle actions like creating a scheduled post entry, retrieving scheduled posts status, fetching incoming messages, and posting replies. We should be mindful of rate limits and ensure our use aligns with HighLevel's terms (particularly since we might be controlling HL features from an external app). - **Facebook/Instagram Graph API**: If needed for certain real-time features (Instagram comment triggers, for example), we'll configure a Facebook App with webhooks. Whenever a new comment or DM is received on a connected Instagram Business account, Facebook's servers will send an event to our webhook endpoint. Our orchestrator service receives it, verifies which client/page it's for, and then invokes Claude to generate an appropriate response. Once the AI composes a reply, we can either respond via the Graph API (for a comment reply or DM) or, if using HighLevel, possibly feed that reply into HighLevel's conversation (so it's logged and sent out through HL). A similar approach can work for Facebook comments/DMs. For LinkedIn or others, we might rely on HighLevel for inbox messages since LinkedIn's API for messaging is limited for third parties.

Security & Permissions: It's crucial to manage tokens and permissions carefully. Our system will store OAuth tokens for each client's social accounts (or HighLevel tokens for each sub-account). We'll encrypt these in our database. The MCP approach helps here – the AI agent doesn't directly handle raw tokens; it simply calls a tool. The tool's backend knows the token and adds it to the API call. This means even if the AI is composing an HTTP request, it doesn't "see" the sensitive credential, as the MCP server injects it. This approach aligns with best practices in agent design where **MCP servers provide structure and ensure consistent implementation** of external calls [18] [19] .

AI-Driven Social Media Functions

Let's detail how the platform will fulfill its core functions with the above architecture:

- **Automated Content Creation**: The platform will assist human marketers in creating content, using AI. Inside the post creation UI, an agent (or just an API call to Claude) can be prompted with a series of questions (HighLevel's Content AI uses guided questions [13]; we can do similarly or use a one-shot prompt). The user might enter a few keywords or a brief (e.g. "Promote the new holiday discount for Client X's product line"), and the AI will generate a few caption options, possibly with relevant emojis and hashtags. The marketer can edit or approve one. We could also integrate an image generation step (perhaps DALL-E or Stability if needed for creating backgrounds or simple graphics), but given the complexity, initial focus may be text and using the client's existing image assets. We ensure each AI-generated caption is reviewed – maybe requiring human approval toggle unless the user explicitly marks the AI content as ready. Over time, with trust, more could be automated.

- **Scheduling and Publishing**: Once content is ready, scheduling can be done via HighLevel's Social Planner API (if available). We'll programmatically create a schedule entry for the post to publish on the desired date/time on chosen platforms [20]. The platform UI will show scheduled posts on a calendar (we can mirror the data we send to HL). If HighLevel API is not fully open for scheduling, as a fallback we might build a scheduling mechanism in our app (storing posts in our DB with a timestamp and having a CRON or scheduled Lambda/Job to push them via native APIs at the right time). However, since **HighLevel supports bulk scheduling and even recurring posts** [15], using it as backend for scheduling is preferable and likely more reliable (their system will take care of hitting the exact post time, accounting for platform restrictions).

- **Monitoring & Analytics**: After posts go out, the system can fetch performance data. HighLevel's API (or our own via each platform) can retrieve basic metrics like likes, comments, reach, etc. As noted, HL's built-in analytics are basic [17], but sufficient for an overview. We could display these on the client dashboard for the agency team to report to clients. For deeper analytics (sentiment of comments, follower growth over time), we might export data to an analytics tool later, but that's beyond MVP scope.

- **Automated Engagement (Likes & Comments)**: We will implement a rule-based trigger system augmented by AI:

- *Auto-Liking*: For example, when someone comments on a client's post, we can automatically "like" that comment via API to show responsiveness. This can be immediate and doesn't require AI (just a small action via Graph API endpoint for likes). It's a simple gesture but we mention it because it's part of "engagement automation." We will ensure this is done within reasonable limits (liking every comment is fine; but if we talk about liking others' posts to engage, we'd need criteria to find target posts — that might be phase 2 feature using hashtags or geolocation search with moderation).
- *AI-Powered Comment Replies*: If a comment is straightforward ("Great post!"), the agent can craft a polite reply ("Thanks so much, we appreciate your support! "). If the comment is a question ("How can I buy this?"), the agent can respond with a helpful answer or link. Initially, we might limit auto-replies to common positive comments or FAQs, and alert a human for anything complex or sensitive. This can be configured per client (some may want fully auto comment replies, some may not).

- *Direct Messages (AI Chat in DMs)*: This is a big feature. When a user sends a DM (on Instagram, Facebook Messenger, etc.), our platform should answer as if a human from the client's team is responding promptly. HighLevel already aggregates DMs from various sources into one inbox; they even introduced an AI bot for lead qualification in chats (Lead Connector AI). However, we want a custom, more flexible AI. So when a new message arrives:
    - HighLevel could send a webhook to us (or we poll their conversation API).
    - We input the message text (and possibly conversation history context) to Claude with a prompt tailored to the client's context ("You are an assistant for [Client], here's what you know about their products/services: ... The user asked: 'Do you have this in stock?'..." etc.).
    - Claude generates a reply. We might integrate some *guardrails* – e.g., if it's unsure or the question is very business-specific and not in its context, it might either say "I'll get back to you" or flag human intervention. There could also be a fallback where outside of business hours the AI answers frequently asked questions, but during work hours maybe a human takes over; this can be configured.
    - Finally, our system sends the AI's response via the appropriate channel. If we use HighLevel, we'd call its API to reply to that conversation thread (so it appears in HL's interface too). If direct, we'd call the platform's messaging API.

One advantage of building this ourselves with Claude vs using ManyChat or HighLevel's built-in flows is flexibility. **ManyChat** is indeed a "gold standard" for IG automation [21] , offering robust templates and quick setup for FAQs, but it primarily uses predefined flows or basic AI. Our solution can use a *more advanced AI* (Claude) that can handle arbitrary questions, maintain context over long conversations, and even perform actions (e.g., book an appointment using a calendar API if the user says they want to schedule a call). This level of integration between chat and actions is something we can achieve by combining the Claude agent with our backend tools. (For instance, Claude could have a tool like `BookAppointment(date, name)` that it might invoke during a chat if the user asks to book something – the tool could tie into HighLevel's calendar or our scheduling system. The user would get a DM confirming the appointment in real time.)

Summary of why not simply use third-party bots: We acknowledge that certain features (Instagram DM automation, Facebook Messenger bots, etc.) can be done with third-party platforms (ManyChat, MobileMonkey, HighLevel's own workflows). However, by building it in-house with our AI agent: - We maintain **full control** over the logic and data (important for customizing to each client's needs, and ensuring the AI aligns with the brand's voice). - We can **iterate faster**. If we need a new automation, we can "vibe code" it quickly with our AI dev pipeline, rather than waiting or contorting a third-party platform to fit our use case. - We avoid additional **costs and dependencies** on multiple SaaS tools. Given the state of technology (powerful open AI models, open APIs), there's less reason to rely on no-code automation services if we can program our own with reasonable effort. As noted, the Claude Agent SDK makes it feasible to create powerful agents with minimal boilerplate [3] . Our only third-party heavy-lift service in this plan is HighLevel, which we already use and which provides a strong backbone (CRM + scheduler); everything else, we either build or integrate via API.

Deployment and Workflow

The application will be deployed on **Vercel** for the frontend and lightweight API routes. This covers the UI and any short, stateless API calls (like "generate caption" which calls Claude API and returns a result – these can run on Vercel serverless, with perhaps a 10s timeout in mind). For **background tasks and webhooks**, we'll deploy a separate service: - Possibly a **Node.js Express server** or a **Python FastAPI** that stays live (hosted on AWS, DigitalOcean, etc., or even on our own always-on VM as some vibe coders do by running

Claude on a Linux VPS [22]). This service will handle incoming webhooks (so social platforms can reach it any time), and it can also run scheduled jobs (like checking for posts to publish if not using HL scheduling, or summarizing daily reports). - We will containerize this service using Docker for portability, managed via our DevOps engineer. In production, we'll have it scale or at least be redundant for reliability (can use a serverless function with queue if that fits, or a small Kubernetes deployment for the agent service).

The **development workflow** will be agile and iterative. We'll use GitHub for version control, and possibly integrate the vibe-coding CLI into our IDEs (there are ways to have an AI CLI assist inside VSCode or others [1]). Each feature will be developed with heavy involvement of the AI co-pilot, but human developers will review and test before merging. Given the fast pace AI coding can provide, we must enforce **code reviews and tests** to ensure quality (the QA engineer will create test scenarios including how the AI agent responds in various social interactions).

Testing the AI agent is a bit different from traditional software testing. We will likely simulate a lot of conversations and posts in a staging environment. For example, test with a dummy Facebook Page and Instagram account: post some content, have team members (or scripted accounts) comment or DM various things, and observe how the agent responds. This helps tune the prompts and tools (maybe we'll discover the AI needs additional knowledge or should avoid certain phrasings). We'll refine this before rolling out to real client accounts.

Finally, **monitoring and logging** will be important in production. We will implement logging for the agent's actions (all tool uses, all messages sent, etc.) so that if anything goes wrong or a client questions a response, we can trace what happened. Alerts can be set if, say, the agent uses an uncommon tool or if an error occurs calling an API, so the dev team can intervene quickly. With great power (an autonomous agent interacting with public platforms) comes the need for oversight!

Considerations and Next Steps

In this plan, we have outlined a comprehensive system leveraging advanced AI and existing platforms. Before implementation, here are a few considerations and possible questions:

- **Platform API Limits & Policies**: We must ensure compliance with each social platform's terms. Automating likes and comments, in particular, can border on spam if overdone. We'll implement rate limiting and perhaps randomness to mimic human behavior, and focus on *quality engagement* (the AI should post meaningful comments, not generic ones, to avoid being flagged as bot). Each platform has specific rules (e.g. Instagram API won't let a non-business account auto-reply, and DMs can only be sent within 24h of last user message for Messenger, etc.). The integration specialist will need to navigate these details.

- **GoHighLevel Usage**: HighLevel is a critical piece if we use it for scheduling and messaging. We should verify what portions of their functionality have open APIs. The presence of new **Social Planner APIs** and mention of "Edge Rules for custom content" [14] is promising. If something we need isn't available via API (for example, approval workflows or story posting might be limited), we might either avoid that feature or use the platform APIs directly for that part. The plan allows flexibility to do a hybrid approach as noted.

- **Claude vs Other Models**: We're planning on Claude as the AI brain (which has the advantage of the Agent SDK and a large context window). It's worth noting we could also consider OpenAI GPT-4 or Google's Gemini if needed, but since the question specifically mentions Claude, we stick to that. Claude is known for its friendly tone and large context, which is great for support-like interactions and handling lots of instructions. We should watch the cost – running many AI calls (for every DM or comment reply) could add up. We might need to implement some logic like: trivial replies use a cheaper model or a cached response, whereas complex queries go to Claude. But since we want best quality and have resources, we lean on Claude for now and perhaps fine-tune or optimize later.

- **Timeline & Milestones**: With unlimited team members, we can parallelize much work:

- *Milestone 1*: Setup the project scaffold (Next.js app on Vercel, database, basic Claude SDK integration). Demo generating a simple post via AI and storing it.
- *Milestone 2*: Implement social posting integration (either via HighLevel or direct API for one platform as POC). Demonstrate scheduling a post from our app.
- *Milestone 3*: Implement DM automation for one platform (e.g. Instagram via direct Graph API webhook to agent response). Test AI conversation quality.
- *Milestone 4*: Expand to other platforms and refine AI prompts (cover more use cases, add sub-agents if needed for multitasking).
- *Milestone 5*: QA and security hardening, then internal launch for the agency team to use on a few real clients as beta.

Throughout, we will use vibe coding to accelerate development, but also ensure the architecture is robust. The unique combination of **LLM-driven development and LLM-driven runtime** (the agent that does automation) is cutting-edge, but our plan accounts for human oversight and using proven tools where sensible (like HighLevel for distribution). By following this plan, we should achieve a state-of-the-art social media automation platform that boosts our agency's efficiency and showcases the power of AI in workflow automation.

---

1 2 7 9 18 19 AI/ML – Richard Seroter's Architecture Musings
https://seroter.com/category/ai-ml/

3 Agent SDK overview - Claude Docs
https://platform.claude.com/docs/en/agent-sdk/overview

4 5 6 8 Building agents with the Claude Agent SDK \ Anthropic
https://www.anthropic.com/engineering/building-agents-with-the-claude-agent-sdk

10 22 I am curious what people use to host their projects that they vibe code. : r/vibecoding
https://www.reddit.com/r/vibecoding/comments/1pkehbf/i_am_curious_what_people_use_to_host_their/

11 How to Get Started with Instagram DM Automation - Manychat Blog
https://manychat.com/blog/how-to-get-started-with-instagram-dm-automation/

12 15 16 17 20 HighLevel Social Media Planner (2026) — Honest Review
https://ghl-services-playbooks-automation-crm-marketing.ghost.io/is-the-go-high-level-social-media-planner-worth-it/

13 Content AI with Social Planner - HighLevel Support Portal
https://help.gohighlevel.com/support/solutions/articles/48001234788-content-ai-with-social-planner

14 New API Endpoints for Social Media Management in Go High Level ...

https://www.facebook.com/groups/highlevelmake/posts/2467953053593148/

21 Unlock the Power of Instagram Automation: Transform Your DMs ...

https://manychat.com/blog/unlock-the-power-of-instagram-automation-transform-your-dms-into-a-growth-engine/