**AI-Powered Social Media Manager on ICP**

**Overview**

This project aims to build an **AI-based social media manager and content generator** on the **Internet Computer (ICP) blockchain** using OpenAI's API. The system will generate **tweets, captions, and blog posts**, store them on ICP, and manage automated posting to various social media platforms.

**Project Scope**

**Features**

* **Content Generation**: Uses OpenAI's GPT model to create social media posts.
* **Blockchain Integration**: Deploys on the ICP blockchain for transparency and security.
* **API-Based Execution**: Calls OpenAI's API to generate content dynamically.
* **Social Media Automation**: Connects to clients' social media accounts for automated posting.
* **Scalability**: Can later expand to analytics, engagement tracking, and multi-platform integration.

**Technology Stack**

* **Blockchain**: Internet Computer (ICP)
* **Backend**: Motoko (smart contracts)
* **AI Model**: OpenAI's GPT (via API calls)
* **Frontend**: React, Motoko frontend canisters
* **Storage**: ICP Canisters (persistent storage for AI-generated content)
* **Social Media APIs**: Twitter, Instagram, LinkedIn API for posting content

**Implementation Steps**

**1. Setting Up the Environment**

**Install the Internet Computer SDK**

* sh -ci "$(curl -fsSL https://smartcontracts.org/install.sh)"

**Create a New ICP Project**

* dfx new ai-content-generator
* cd ai-content-generator

**2. Writing the Motoko Smart Contract**

**File: src/ai-content-generator/main.mo**

This Motoko actor:

* Accepts a **prompt** and content type (tweet, caption, blog).
* Calls **OpenAI’s API**.
* Stores generated content and schedules posting to linked social media accounts.
* import Debug "mo:base/Debug";
* import Http "mo:base/Http";
* import Text "mo:base/Text";
* import Blob "mo:base/Blob";
* actor AIContentGenerator {
* let openAIUrl = "https://api.openai.com/v1/chat/completions";
* let apiKey = "your-openai-api-key";
* let storedPosts = Buffer.Buffer<Text>(0);
* public func generateContent(prompt: Text, contentType: Text) : async Text {
* let formatInstructions = switch contentType {
* case ("tweet") { "Generate a tweet under 280 characters with hashtags." };
* case ("caption") { "Generate an engaging Instagram caption with emojis and hashtags." };
* case ("blog") { "Generate a short blog post (around 300 words) with a title and conclusion." };
* case \_ { "Generate general content." };
* };
* let fullPrompt = formatInstructions # "\nPrompt: " # prompt # "\nAI Response:";
* let requestBody = Text.concat("{ \"model\": \"gpt-4\", \"messages\": [{\"role\": \"user\", \"content\": \"", fullPrompt # "\"}]}" );
* let headers = [("Authorization", "Bearer " # apiKey), ("Content-Type", "application/json")];
* let response = await Http.post(openAIUrl, headers, Blob.fromArray(Text.toArray(requestBody)));
* switch (response.status) {
* case (200) {
* let generatedText = Text.fromBlob(response.body);
* storedPosts.add(generatedText);
* return generatedText;
* };
* case \_ { Debug.print("Error: " # Int.toText(response.status)); return "Failed to generate content." };
* };
* };
* public func getGeneratedPosts() : async [Text] {
* return Buffer.toArray(storedPosts);
* };
* };

**3. Connecting to Social Media Accounts**

To automate posting, we will use **OAuth authentication** to link users’ social media accounts:

* **Twitter API** (for tweets)
* **Instagram Graph API** (for captions & stories)
* **LinkedIn API** (for blog posts)

Each user will authenticate and authorize our system to post content on their behalf.

**OAuth Flow**

1. User clicks "Connect Account."
2. Redirects to the social media platform for authentication.
3. User grants permission to our AI system.
4. We store an **access token** securely.
5. AI system uses this token to schedule and post content.

**4. Deploying and Running the Smart Contract**

**Start the local ICP network**

* dfx start --background

**Deploy the canister**

* dfx deploy

**Test the function**

* dfx canister call ai-content-generator generateContent '("How AI is changing social media", "tweet")'

This will return an AI-generated tweet.

**Next Steps**

* **Build a Frontend**: Use React or Motoko frontend canisters.
* **Store AI-Generated Content on ICP**: Use persistent storage.
* **Automate Posting**: Implement OAuth authentication and schedule AI-generated posts.
* **Expand to Multi-Modal AI**: Include image and video generation.
* **Analytics & Engagement Tracking**: Monitor likes, shares, and comments.