1. Project Overview

The Web Platform is a comprehensive, multi-functional application built on the CodeIgniter 4 framework. It serves as a portal for registered users to access a suite of powerful digital services. The platform is designed with a modular architecture, featuring a robust user authentication system, an account management dashboard with an integrated balance and payment system, and an administrative panel for user oversight.

The core services offered include:

- **Gemini AI Studio:** An advanced interface allowing users to interact with Google's Gemini AI, featuring text and multimedia prompts, conversational memory, and context-aware responses.
- **Cryptocurrency Data Service:** A tool for querying real-time balance and transaction data for Bitcoin (BTC) and Litecoin (LTC) addresses.

The application integrates with several third-party APIs, including Paystack for secure payments, Google reCAPTCHA for spam prevention, and the respective blockchain and AI service endpoints.

2. Core Technologies

- **Backend:** PHP 8.1+, CodeIgniter 4
- **Frontend:** Bootstrap 5, JavaScript, HTML5, CSS3
- **Database:** MySQL (via MySQLi driver)
- **Key Libraries:** Parsedown (for Markdown rendering), TinyMCE (for rich text editing), Kint (for debugging), **NlpTools (for Natural Language Processing)**
- **Development Tooling:** Composer, PHPUnit

3. Installation and Setup

1. **Prerequisites:** Ensure you have PHP 8.1+, Composer, and a MySQL database server installed.
2. **Clone Repository:** Clone the project to your local machine.
3. **Install Dependencies:** Run `composer install` to download the required PHP packages, including the `nlp-tools` library.
4. **Configure Environment:**
    - Copy the `env` file to a new file named `.env`.
    - Open `.env` and configure the following variables:
        - `CI_ENVIRONMENT`: Set to `development`.
        - `app.baseURL`: The base URL of your application (e.g., `http://localhost:8080/`).
        - **Database:** Configure `database.default.hostname`, `database.default.database`, `database.default.username`, and `database.default.password`.
        - **API Keys:** Provide your secret keys for `PAYSTACK_SECRET_KEY`, `GEMINI_API_KEY`, `recaptcha.siteKey`, and `recaptcha.secretKey`.

    - **Email:** Set up your SMTP server details under the `email.*` variables for sending verification and password reset emails.
5. **Install Frontend Assets:** Download the TinyMCE community edition and place its contents in the `public/assets/tinymce/` directory to enable the rich text editor.
6. **Database Migration:** Run the migrations to create the necessary database tables:
   ```
   php spark migrate
   ```
7. **Run the Application:** Start the development server:
   ```
   php spark serve
   ```

4. Application Architecture

The project strictly follows the **Model-View-Controller (MVC)** architectural pattern, extended with a **Service layer** to adhere to modern software design principles and the project's internal coding standards (`clinerules.md`).

- **`app/Controllers`:** Controllers are the entry point for user requests. They are kept lean and are responsible for orchestrating the application flow, validating input, and calling upon models and services to perform business logic.
- **`app/Models`:** Models are responsible for all database interactions.
- **`app/Entities`:** Entities are object-oriented representations of database table rows.
- **`app/Views`:** Views handle all presentation logic, built on Bootstrap 5.
- **`app/Libraries` (Services):** This directory contains the core business logic.
  - **`TokenService.php` (New):** A new service dedicated to Natural Language Processing. It encapsulates the pipeline for converting raw text into a clean, stemmed, and stop-word-free list of keywords (tokens). It now includes a pre-processing step to strip all HTML tags, ensuring only plain text is tokenized.
  - Other services like `PaystackService`, `GeminiService`, `CryptoService`, and the sophisticated `MemoryService` encapsulate interactions with external APIs and complex application logic.
- **`app/Helpers`:** Contains simple, stateless procedural functions that are globally available.
- **`app/Config`:** Centralizes all application configuration, including a custom `AGI.php` file that fine-tunes the behavior of the AI's memory system and the NLP stop words list.
- **`app/Filters`:** These classes act as middleware to protect routes.
- **`app/Database`:** Contains all database migrations.

5. Database Schema

The database schema is defined by the following migration files and tables:

- **`users`**: Stores user credentials, profile information, and account balance.
- **`payments`**: A log of all payment transactions made through Paystack.
- **`prompts`**: Enables users to save and reuse frequently used AI prompts.

- **`interactions`**: The core of the AI's long-term memory. Each row stores a conversational turn, including raw text, relevance score, vector embedding, and **now stores a cleaner, stemmed list of keywords** generated by the `TokenService`.
- **`entities`**: Acts as a knowledge graph for the AI's memory.
- **`user_settings`**: Stores user-specific preferences.
- **`campaigns`**: Stores reusable email campaign templates.

6. Key Features and Modules

6.1. Authentication (`AuthController`)

- Secure registration, login, and password reset flows.

6.2. User Dashboard & Account Management (`HomeController`, `AccountController`)

- Personalized dashboard and transaction history.

6.3. Admin Panel (`AdminController`, `CampaignController`)

- User management, financial oversight, and mass email campaign functionality.

6.4. Payment System (`PaymentsController`, `PaystackService`)

- Secure payment processing via Paystack.

6.5. Services

A. Crypto Data Service (`CryptoController`, `CryptoService`)

- Real-time balance and transaction queries for Bitcoin and Litecoin.

B. Gemini AI Studio (`GeminiController`, `MemoryService`, `EmbeddingService`, `TokenService`)

- **Rich Text & Multimedia Prompting:** Allows users to submit formatted HTML and various media files to the Gemini API.
- **Assistant Mode (Conversational Memory):** A highly advanced feature powered by `MemoryService`.
  - **Context Retrieval:** Before sending a prompt, the service performs a **hybrid search** of past interactions. It combines a **vector search** (for semantic similarity) with a **keyword search** (for lexical relevance) to retrieve the most relevant memories.
  - **Improved Keyword Extraction:** The keyword extraction process now uses the `TokenService` to perform a robust NLP pipeline. The first step in this pipeline is to **strip all HTML tags** from the user's input, ensuring that only clean text is processed. This is followed by tokenization, stop word removal, and stemming to produce highly relevant keywords for memory retrieval.
  - **Prompt Augmentation:** This retrieved context is prepended to the user's current prompt, providing the AI with a rich conversational history.

    - **Memory Update:** After receiving a response, the system learns by rewarding the relevance of used context, decaying unused memories, and saving the new interaction with its own vector embedding and processed keywords.
    - **Persistent Setting:** The user's choice to enable or disable Assistant Mode is saved to their profile.
- **Token-Based Billing:** The system calculates the exact cost of each query based on token usage.
- **Prompt Management:** Provides a UI for users to save, load, and delete their favorite prompts.

6.6. Public-Facing Pages & SEO

- Includes public landing pages, legal pages, a contact form, and a dynamically generated sitemap for search engine optimization.

7. Documentation and Best Practices (`clinerules.md`)

The project is governed by a strict set of internal rules documented in `.clinerules.md`, which ensures a high standard of code quality, security, and maintainability. This includes standards for the MVC-S architecture, named routing, the Post/Redirect/Get pattern, security mandates, and a unified frontend workflow.