Cheshire Terminal x Privy

Powered by Cheshire: Multi-Modal AI Agent with Base and Solana Blockchain Integration

Cheshire Terminal is a groundbreaking platform that combines advanced AI capabilities with blockchain technology, creating a seamless bridge between natural language processing, audio interactions, and Web3 functionality.

🚀 Innovation

This project introduces several first-of-its-kind integrations:

1. **Unified AI Stack**
    - Combines XAI's Grok models for vision and text processing
    - Integrates OpenRouter for access to multiple AI models
    - Features ElevenLabs for voice synthesis
    - Utilizes Deepgram for real-time speech recognition
2. **Multi-Chain Authentication**
    - Privy authentication for Web2 user management
    - Coinbase Digital Platform (CDP) integration for institutional-grade wallet management
    - Helius integration for Solana blockchain interactions
    - Base network integration for Ethereum L2 functionality
3. **Real-Time Processing Pipeline**
    - Redis-powered caching layer for high-performance data access
    - Supabase backend for structured data storage
    - WebSocket integration for real-time updates

    - Twilio integration for phone-based interactions

🌟 Key Features

- **Voice-Enabled AI Interactions**: Users can speak naturally with the AI agent through voice commands
- **Cross-Chain Asset Management**: View and manage assets across multiple blockchains
- **Real-Time Transaction Monitoring**: Track blockchain transactions with instant notifications
- **Multi-Modal Content Generation**: Generate text, images, and audio responses
- **Institutional-Grade Security**: Enterprise-level security through CDP integration
- **Phone-Based Authentication**: Secure authentication through phone verification

🛠️ Technical Architecture

AI Layer

- XAI Service: Handles vision and text processing using Grok models
- OpenRouter Service: Provides access to multiple AI models
- Deepgram Service: Manages real-time speech recognition
- ElevenLabs Service: Handles voice synthesis

Blockchain Layer

- CDP Service: Manages institutional wallet operations
- Helius Service: Handles Solana blockchain interactions
- Basescan Service: Provides Base network explorer functionality
- Network Configuration: Manages blockchain network settings

Data Layer

- Redis Service: Handles caching and real-time data
- Supabase Service: Manages structured data storage
- WebSocket Service: Enables real-time updates

🚀 Getting Started

1. **Environment Setup**

```
# Clone the repository
git clone [repository-url]
cd [project-directory]

# Install dependencies
yarn install
```

2. **Configure Environment Variables**
   Create a .env file with the following:

```
# Privy Authentication
NEXT_PUBLIC_PRIVY_APP_ID=your_privy_app_id
PRIVY_APP_SECRET=your_privy_secret

# AI Services
VITE_XAI_API_KEY=your_xai_key
VITE_XAI_VISION_MODEL=grok-2-vision-1212
VITE_XAI_TEXT_MODEL=grok-2
VITE_OPEN_ROUTER_API_KEY=your_openrouter_key
VITE_OPEN_ROUTER_MODEL=your_model
VITE_ELEVEN_LABS_API_KEY=your_elevenlabs_key
VITE_ELEVEN_LABS_AGENT_ID=your_agent_id
VITE_DEEPGRAM_API_KEY=your_deepgram_key

# Blockchain Services
VITE_CDP_CLIENT_ID=your_cdp_client_id
VITE_CDP_API_KEY_NAME=your_cdp_key_name
```

```
VITE_CDP_API_KEY_PRIVATE_KEY=your_cdp_private_key
VITE_HELIUS_API_KEY=your_helius_key
VITE_HELIUS_RPC_URL=your_helius_rpc_url
VITE_BASESCAN_API_KEY=your_basescan_key
VITE_NETWORK_ID=base-sepolia

# Communication Services
VITE_TWILIO_ACCOUNT_SID=your_twilio_sid
VITE_TWILIO_AUTH_TOKEN=your_twilio_token
VITE_TWILIO_API_KEY=your_twilio_key
VITE_TWILIO_API_KEY_SECRET=your_twilio_secret
VITE_TWILIO_PHONE_NUMBER=your_twilio_number
VITE_AI_AGENT_PHONE_NUMBER=your_agent_number

# Database Services
VITE_SUPABASE_URL=your_supabase_url
VITE_SUPABASE_ANON_KEY=your_supabase_key
VITE_REDIS_URL=your_redis_url
```

3. **Start the Development Server**

```
yarn dev
```

💡 Usage Examples

Voice Interaction

```
// Initialize voice chat
const voiceChat = new VoiceChat();
await voiceChat.start();

// The AI agent will respond through voice and can:
// - Answer questions
// - Execute blockchain transactions
// - Generate images
// - Provide real-time market data
```

Asset Management

```
// View cross-chain assets
const assets = await heliusService.getAssetsByOwner(userAddress);
const baseTransactions = await basescanService.getTransactionsByAddress
(userAddress);

// Execute transactions
const transaction = await cdpService.createTransaction({
  to: recipientAddress,
  value: amount,
});
```

AI Content Generation

```
// Generate multi-modal content
const textResponse = await xaiService.textCompletion({
  prompt: "Explain blockchain technology",
});

const imageAnalysis = await xaiService.analyzeImage({
  image: imageData,
  prompt: "Describe this NFT",
});
```

🔒 Security Considerations

- All API keys are securely stored in environment variables
- CDP integration provides institutional-grade security
- Phone verification adds an extra layer of authentication

- Real-time transaction monitoring for suspicious activity

🤝 Contributing

We welcome contributions! Please see our Contributing Guide for details.

📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

🙏 Acknowledgments

- XAI team for Grok model access
- Coinbase for CDP integration
- Helius for Solana integration
- Base team for L2 support
- ElevenLabs for voice synthesis
- Deepgram for speech recognition
- OpenRouter for AI model access
- Privy for authentication
- Supabase for database
- Redis for caching
- Twilio for communication

Contributing to Multi-Modal AI Agent

We love your input! We want to make contributing to this project as easy and transparent as possible, whether it's:

- Reporting a bug
- Discussing the current state of the code
- Submitting a fix

- Proposing new features
- Becoming a maintainer

Development Process

We use GitHub to host code, to track issues and feature requests, as well as accept pull requests.

1. Fork the repo and create your branch from `main`.
2. If you've added code that should be tested, add tests.
3. If you've changed APIs, update the documentation.
4. Ensure the test suite passes.
5. Make sure your code lints.
6. Issue that pull request!

Any Contributions You Make Will Be Under the MIT Software License

In short, when you submit code changes, your submissions are understood to be under the same MIT License that covers the project. Feel free to contact the maintainers if that's a concern.

Report Bugs Using GitHub's Issue Tracker

We use GitHub issues to track public bugs. Report a bug by opening a new issue; it's that easy!

Write Bug Reports with Detail, Background, and Sample Code

Great Bug Reports tend to have:

- A quick summary and/or background
- Steps to reproduce
  - Be specific!

    - Give sample code if you can.
- What you expected would happen
- What actually happens
- Notes (possibly including why you think this might be happening, or stuff you tried that didn't work)

Use a Consistent Coding Style

- Use TypeScript for type safety
- 2 spaces for indentation rather than tabs
- You can try running `yarn lint` for style unification

License

By contributing, you agree that your contributions will be licensed under its MIT License.

References

This document was adapted from the open-source contribution guidelines for Facebook's Draft.