Solana Intro Dapp

Nick Carpinito, Grant Edens

Our project

Since our assignments in this class have been with Ethereum, we thought it would be a good exercise to do a project on the Solana blockchain, another platform that supports decentralized applications.

Today we will be using an IDE to deploy a Solana smart contract to a dev net. This contract will display text and will be accessible through an API. We will also create a frontend application on a local web server.

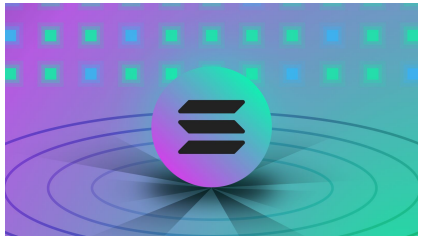
Tradeoffs between Solana/Ethereum

In 2017, Solana creator Anatoly Yakovenko published the Solana white paper, envisioning a “metaverse where [everything is] protocol based”. He aimed to utilize a consensus mechanism called proof-of-history which allows for faster transaction speeds than Ethereum. Eventually, Solana launched in 2020 backed by a non-profit called the Swiss Solana foundation.

Since Ethereum was launched several years earlier, Solana developers have not yet had the chance to create as many decentralized applications as those that are available on Ethereum. But Solana has still garnered a lot of interest as alternative to Ethereum. In November 2021, Solana reached a peak market cap of $74 billion. But even today, Solana has a market cap of over $4.5 billion (possibly undervalued?).

https://coinmarketcap.com/currencies/solana/
https://solana.com/solana-whitepaper.pdf

Advantages of Solana



While Ethereum has developed the most robust critical mass for Layer 1 solutions, Solana offers advantages that Ethereum does not.

Solana utilizes a proof-of-history consensus mechanism, put most simply as “a series of computational steps to determine the cryptographical time between two events”. This consensus is faster than proof–of-stake or proof-of-work because proof-of-history “can reduce messaging overhead in a Byzantine Fault Tolerant replicated state machine, resulting in sub-second finality times”.

Because of this faster transaction speed, Solana is more scalable than Ethereum. Also, the gas fees for Solana contracts are less than those on Ethereum. These factors make Solana a viable option for video game and NFT-related use cases

https://beincrypto.com/learn/solana-vs-ethereum/

https://www.yahoo.com/video/why-solana-founder-anatoly-yakovenko-224935383.html

Rust Programming Language

The primary smart contract language of Solana is called Rust. Unlike Solidity, Rust is a multi paradigm language, making it more complicated but also more flexible. Luckily we already know the code we will need.

Rust can easily be installing using the command-line by entering “curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh” and then selecting “1” to proceed with the installation.

We then use "$HOME/.cargo/env" to configure Solana and Rust as environment variables in our shell.

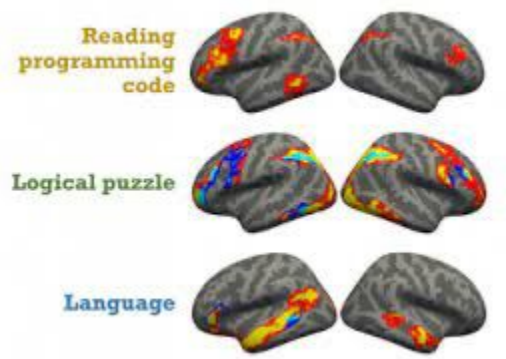
Our Rust contract

```
use solana_program::{
    account_info::AccountInfo,
    entrypoint,
    entrypoint::ProgramResult,
    pubkey::Pubkey,
    msg,
};

entrypoint!(process_instruction);

pub fn process_instruction(
    program_id: &Pubkey,
    accounts: &[AccountInfo],
    instruction_data: &[u8]
) -> ProgramResult {

    msg!("Hello, world!");

    Ok(())
}
```

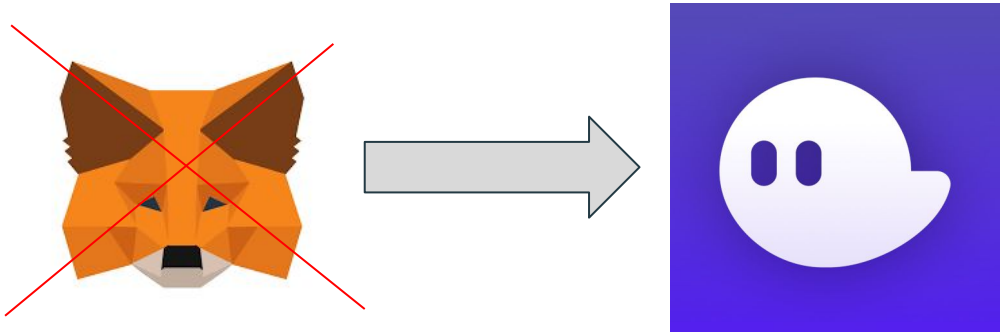
//Tells Solana CLI to use the Solana Programs package we installed.

//Specifies the endpoint to connect with

//Uses basic function structure to create a function that can print to the program log



Phantom Wallet



For our solidity projects we used Metamask but for this Solana project we will use a Solana wallet called Phantom wallet. Phantom is the most widely used wallet in the Solana ecosystem and is planning in integrating Ethereum and Polygon in the near future.

Go to https://phantom.app/ and follow the instructions to install.

Github and Git

Github is a common framework for source control in programming. This is how we “Clone” the necessary frameworks and packages to make our code work without building everything from scratch.

We chose to write our instructions as a guide on Github and pull the front-end components from the repository hosted at https://github.com/johnvsnagendra/solana-smart-contract-helloWorld to test our contract once we deploy it.

Git is most commonly used via the command line using the syntax `git clone <LINK>`.

But we will need an API key…

Moralis API

Moralis is a website specializing in “providing world-class APIs to developers across the globe, allowing companies and projects of all sizes to seamlessly integrate blockchain into their solutions”.

For this project we will create a free Moralis account in order to generate our API key.

You will have to complete Moralis onboarding, and after you can click "Web3 APIs" to display your key.

https://moralis.io/

Our Github Repository

https://github.com/meta-lite/solana-hello-world/blob/main/README.md

Feel free to try this on your own and let us know if you have any issues!