Use solana\_program::{

Account\_info::{next\_account\_info, AccountInfo},

Entrypoint,

Entrypoint::ProgramResult,

Program\_error::ProgramError,

Pubkey::Pubkey,

Program\_pack::{Pack, IsInitialized},

System\_instruction,

Sysvar::{rent::Rent, Sysvar},

};

Use solana\_sdk::{

Signature::{Keypair, Signer},

Transaction::Transaction,

Commitment\_config::CommitmentConfig,

Message::Message,

};

Use std::str::FromStr;

Struct NFT {

Name: String,

Symbol: String,

Uri: String,

}

#[derive(Accounts)]

Pub struct MintNft<’info> {

#[account(init, payer = user, space = 165)]

Nft\_account: Account<’info, NftAccount>,

#[account(mut)]

User: Signer<’info>,

Rent: Sysvar<’info, Rent>,

}

#[account]

Pub struct NftAccount {

Pub is\_initialized: bool,

Pub name: String,

Pub symbol: String,

Pub uri: String,

}

#[entrypoint]

Fn mint\_nft(program\_id: &Pubkey, accounts: &mut MintNft, name: String, symbol: String, uri: String) -> ProgramResult {

Let nft\_account = &mut accounts.nft\_account;

If nft\_account.is\_initialized() {

Return Err(ProgramError::AccountAlreadyInitialized);

}

Let tomorrow = solana\_program::clock::get()?.unix\_timestamp + 86400;

Let current\_time = solana\_program::clock::get()?.unix\_timestamp;

If current\_time < tomorrow {

Return Err(ProgramError::Custom(1000));

}

Nft\_account.is\_initialized = true;

Nft\_account.name = name;

Nft\_account.symbol = symbol;

Nft\_account.uri = uri;

Ok(())

}

Fn main() {

Let nfts = vec![

NFT {

Name: “My first NFT”.to\_string(),

Symbol: “NFT1”.to\_string(),

Uri: <https://example.com/nft1.to_string()>,

},

NFT {

Name: “My second NFT”.to\_string(),

Symbol: “NFT2”.to\_string(),

Uri: <https://example.com/nft2.to_string()>,

},

NFT {

Name: “My third NFT”.to\_string(),

Symbol: “NFT3”.to\_string(),

Uri: <https://example.com/nft3.to_string()>,

},

];

// Configure the Solana Devnet endpoint and the payer account

Let endpoint = <https://api.devnet.solana.com.to_string()>;

Let payer = Keypair::from\_base58\_string(“your\_payer\_base58\_string\_here”).unwrap();

// Create a new Solana Keypair to use as the program owner

Let program\_owner = Keypair::new();

// Compile the program Rust code into a BPF program

Let program\_data = include\_bytes!(“../target/deploy/nft.so”);

Let program\_account = Keypair::new();

Let mut transaction = Transaction::new\_with\_payer(

&[system\_instruction::create\_account(

&payer.pubkey(),

&program\_account.pubkey(),

1\_000\_000,

Program\_data.len() as u64,

&spl\_token::id(),

), system\_instruction::assign(&program\_account.pubkey(), &spl\_token