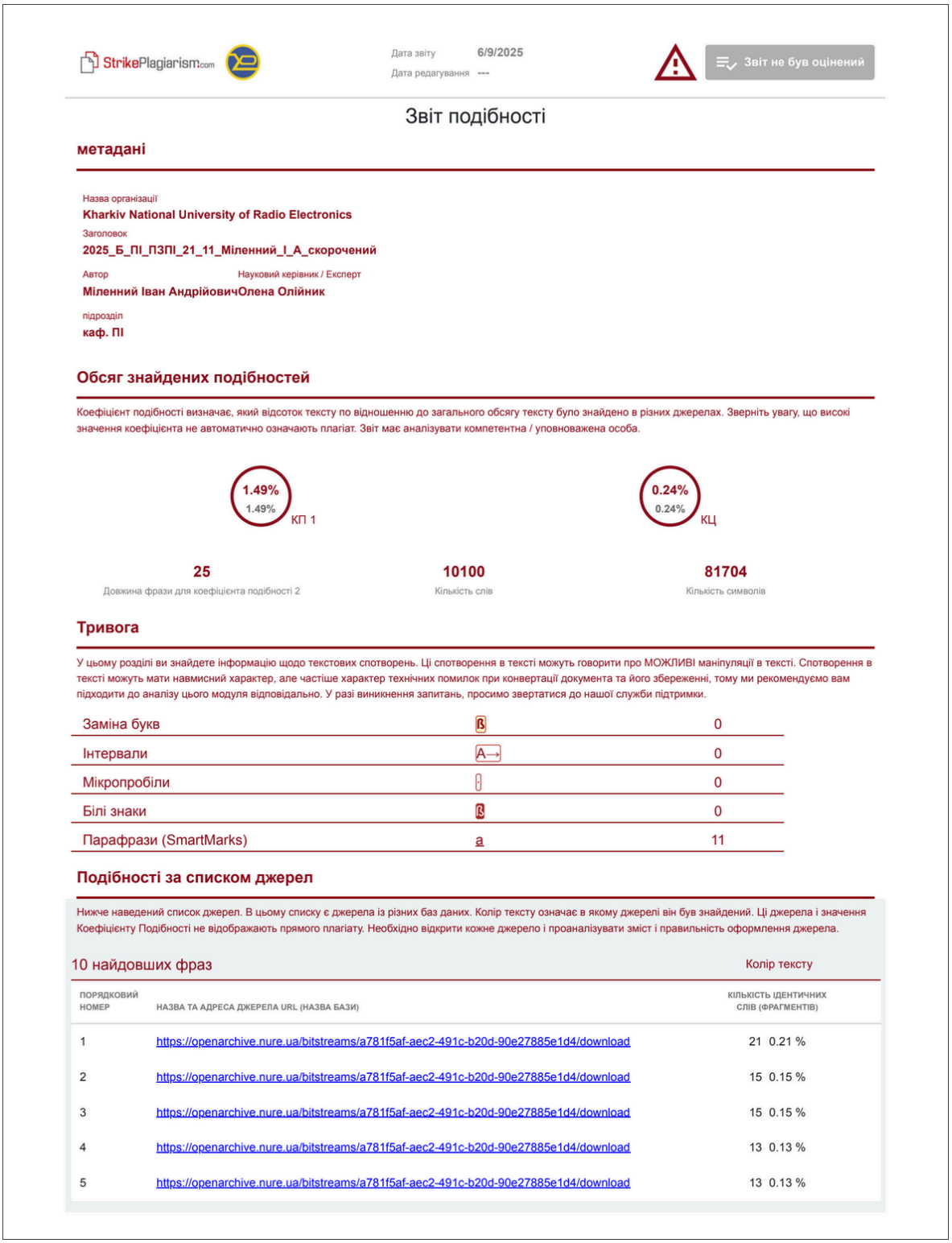
ДОДАТОК А

Звіт результатів перевірки унікальності тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 6/9/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_Б_ПІ_ПЗПІ_21_11_Міленний_І_А_скорочений
Автор Науковий керівник / Експерт
Міленний Іван АндрійовичОлена Олійник
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

1.49% 1.49% КП 1

0.24% 0.24% КЦ

25	10100	81704
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	¦	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	11

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://openarchive.nure.ua/bitstreams/a781f5af-aec2-491c-b20d-90e27885e1d4/download	21 0.21 %
2	https://openarchive.nure.ua/bitstreams/a781f5af-aec2-491c-b20d-90e27885e1d4/download	15 0.15 %
3	https://openarchive.nure.ua/bitstreams/a781f5af-aec2-491c-b20d-90e27885e1d4/download	15 0.15 %
4	https://openarchive.nure.ua/bitstreams/a781f5af-aec2-491c-b20d-90e27885e1d4/download	13 0.13 %
5	https://openarchive.nure.ua/bitstreams/a781f5af-aec2-491c-b20d-90e27885e1d4/download	13 0.13 %

6	https://openarchive.nure.ua/bitstreams/a781f5af-aec2-491c-b20d-90e27885e1d4/download	13 0.13 %
7	https://openarchive.nure.ua/bitstreams/a781f5af-aec2-491c-b20d-90e27885e1d4/download	13 0.13 %
8	https://openarchive.nure.ua/bitstreams/a781f5af-aec2-491c-b20d-90e27885e1d4/download	12 0.12 %
9	https://openarchive.nure.ua/bitstreams/a781f5af-aec2-491c-b20d-90e27885e1d4/download	8 0.08 %
10	https://openarchive.nure.ua/bitstreams/a781f5af-aec2-491c-b20d-90e27885e1d4/download	7 0.07 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з Інтернету (1.49 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://openarchive.nure.ua/bitstreams/a781f5af-aec2-491c-b20d-90e27885e1d4/download	144 (12) 1.43 %
2	https://openarchive.nure.ua/bitstreams/7c3f3dfe-e12e-465e-94fb-558e4cf87424/download	6 (1) 0.06 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

ДОДАТОК Б

Слайди презентації

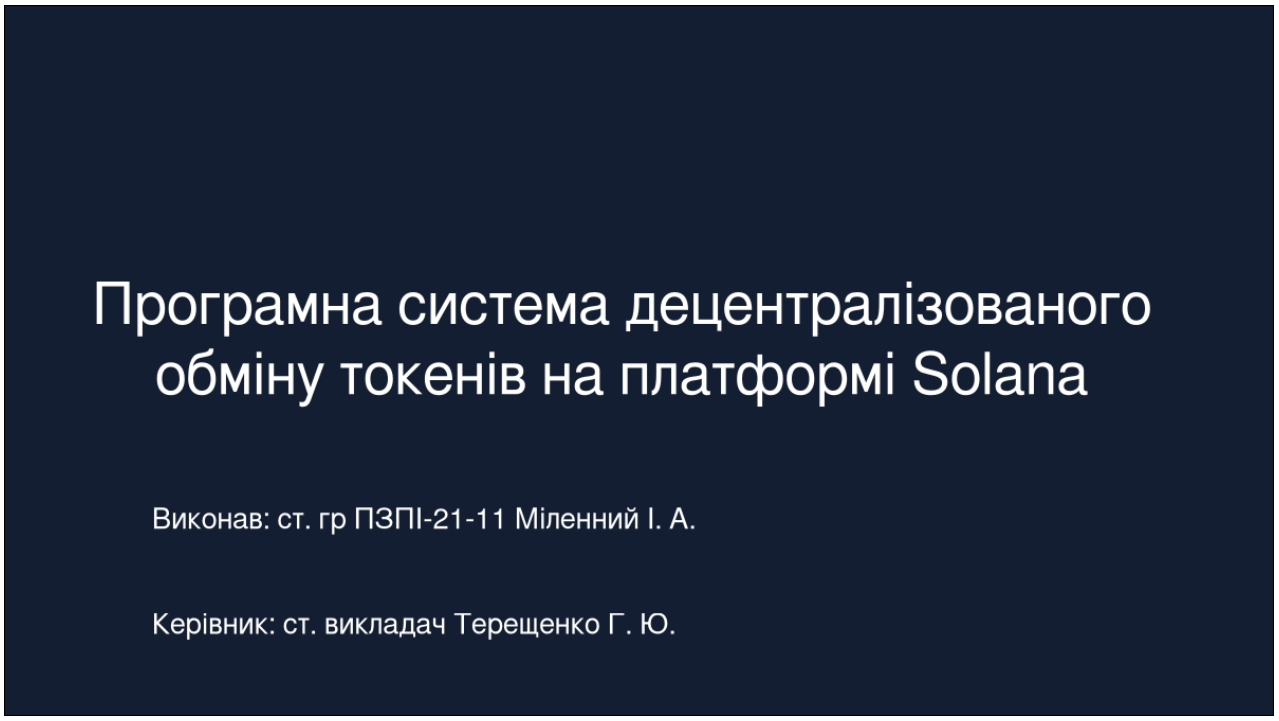


Рисунок Б.1 – Слайд №1

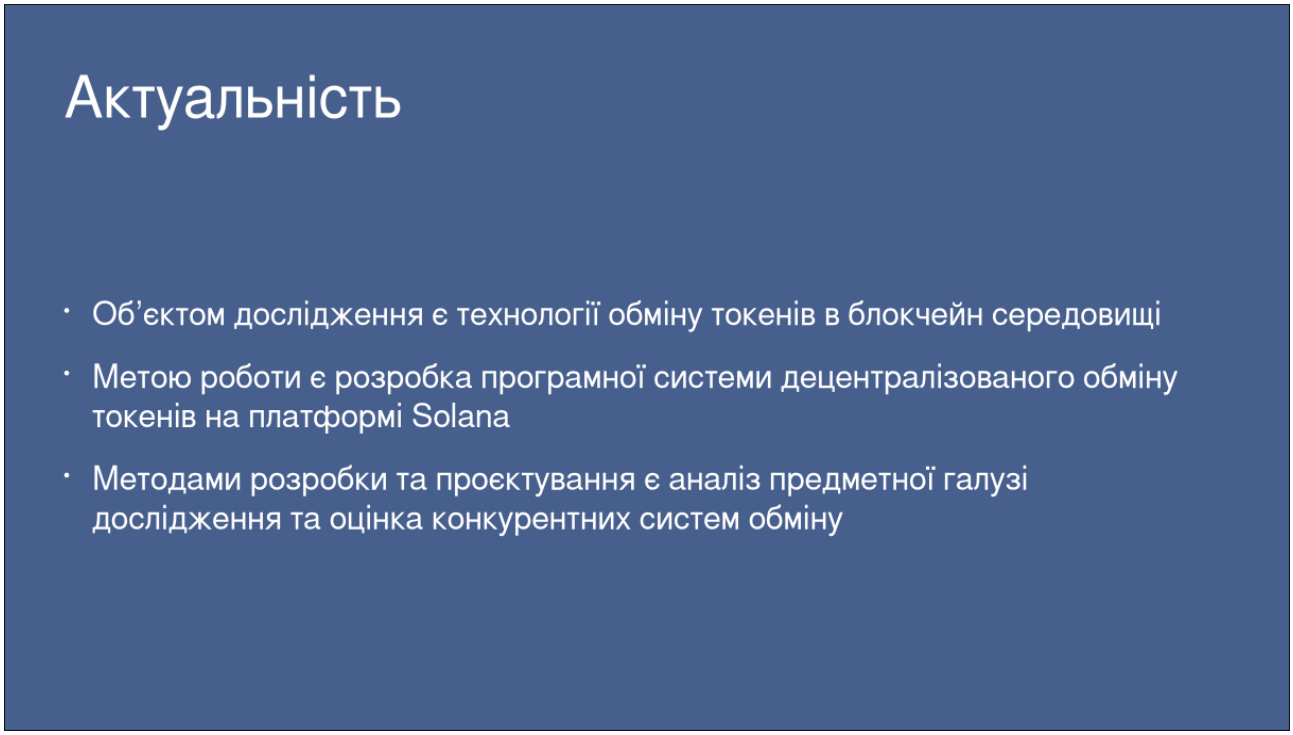


Рисунок Б.2 – Слайд №2

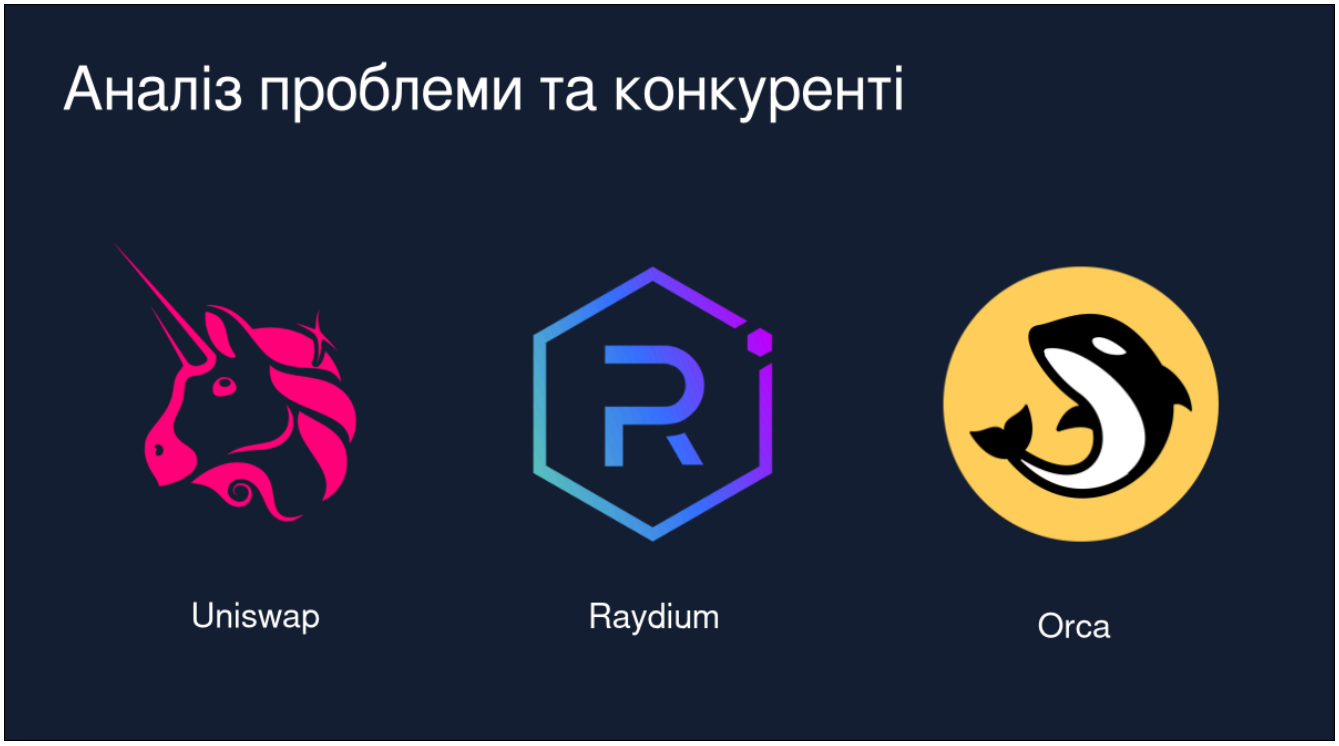


Рисунок Б.3 – Слайд №3

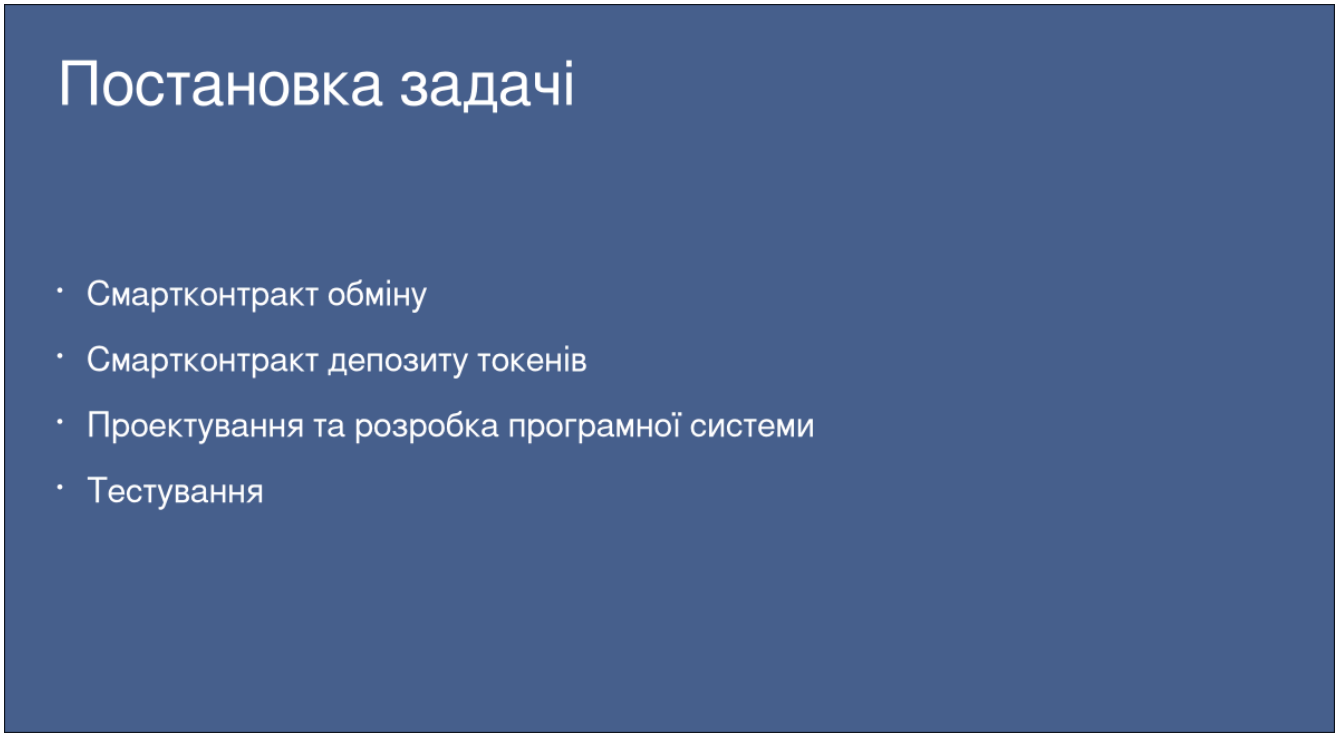


Рисунок Б.4 – Слайд №4

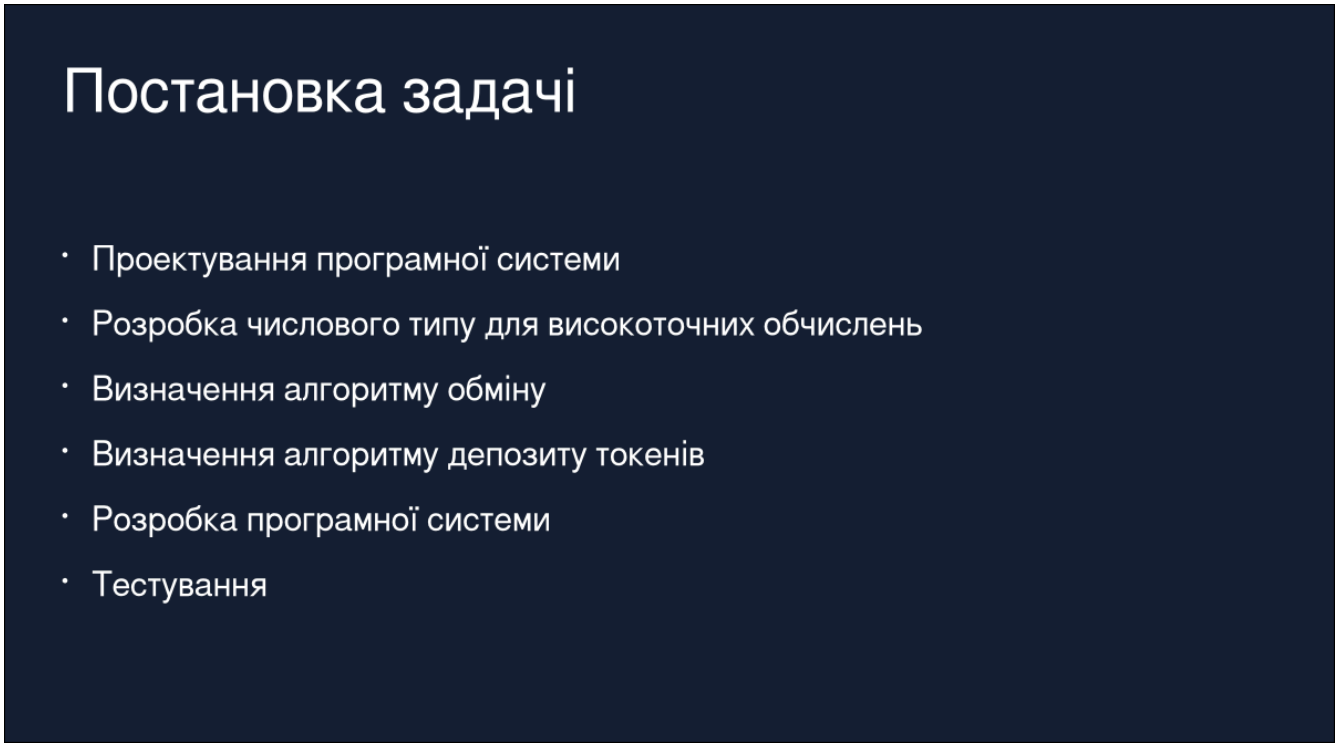


Рисунок Б.5 – Слайд №5

Рисунок Б.6 – Слайд №6

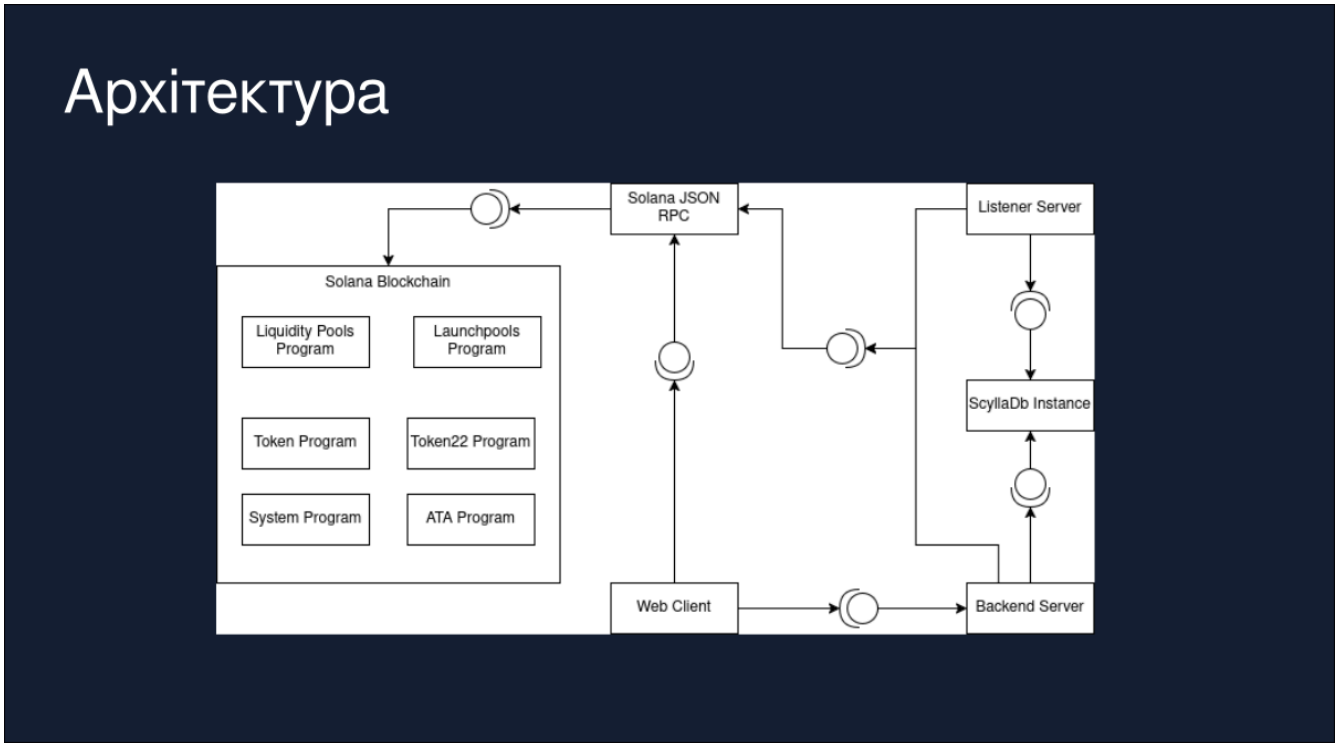


Рисунок Б.7 – Слайд №7

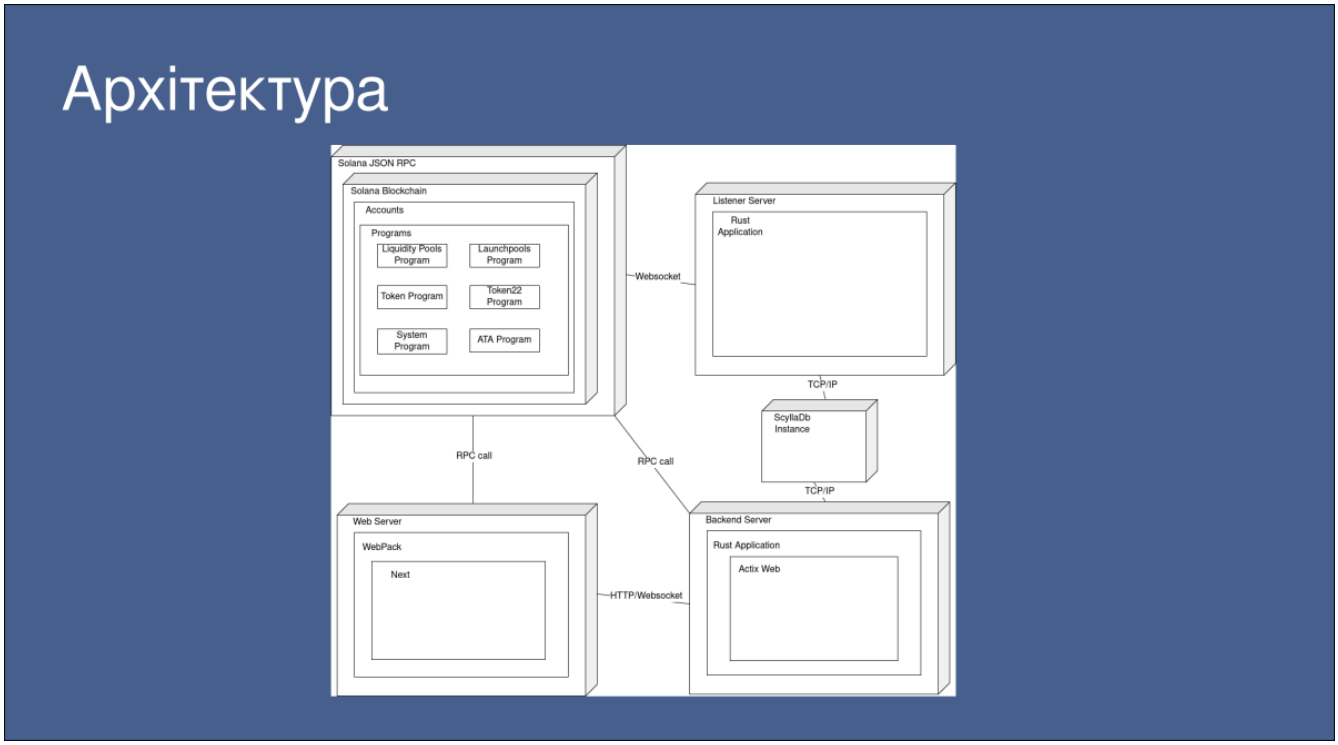


Рисунок Б.8 – Слайд №8

Архітектура

Тип Q64.128 для високоточних обчислень

```rust
pub struct Q64_128 {
    /// The internal representation
    value: U192,
}
```

Рисунок Б.9 – Слайд №9

Архітектура

Алгоритм Constant Product Market Maker

$$x \times y = k$$

$$\Delta y = \frac{k}{x + \Delta x} - y$$

Рисунок Б.10 – Слайд №10

Архітектура

Алгоритм Linear Time-Based Fixed Reward Distribution

$$r = \frac{R_{total}}{D}$$

$$\Delta RPT = \frac{r \times (t - T_{last})}{S}$$

Рисунок Б.11 – Слайд №11

Архітектура

Смартконтракт пулів ліквідності

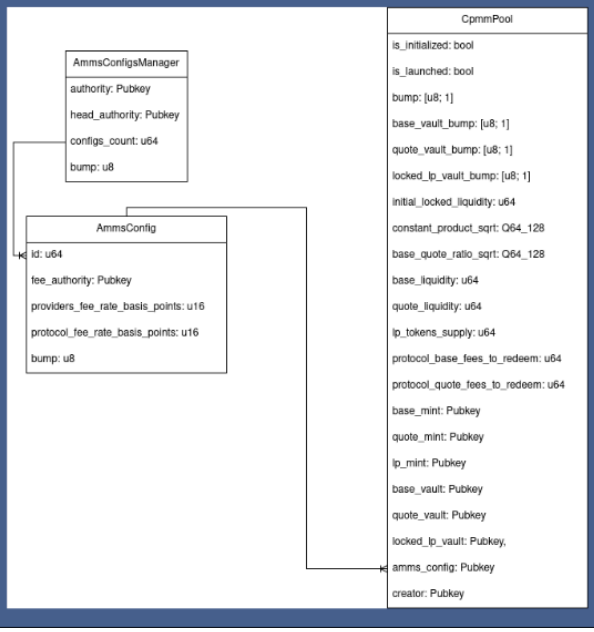


Рисунок Б.12 – Слайд №12

Архітектура

Смартконтракт лаунчпулів

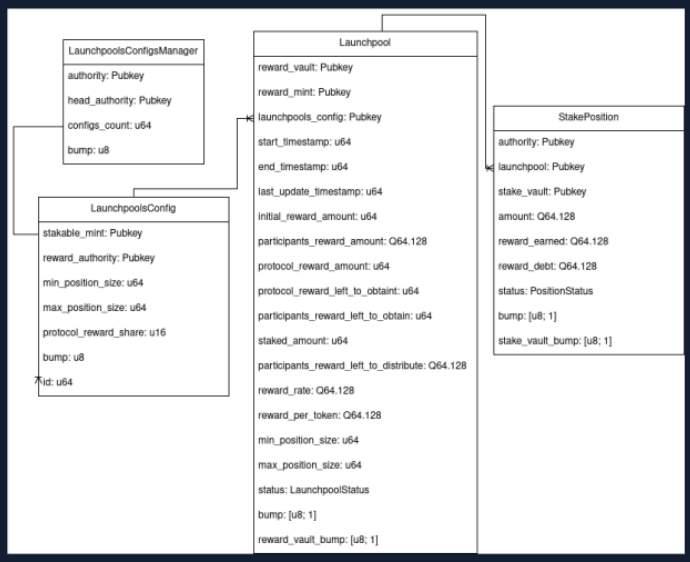


Рисунок Б.13 – Слайд №13

Діаграми прецедентів

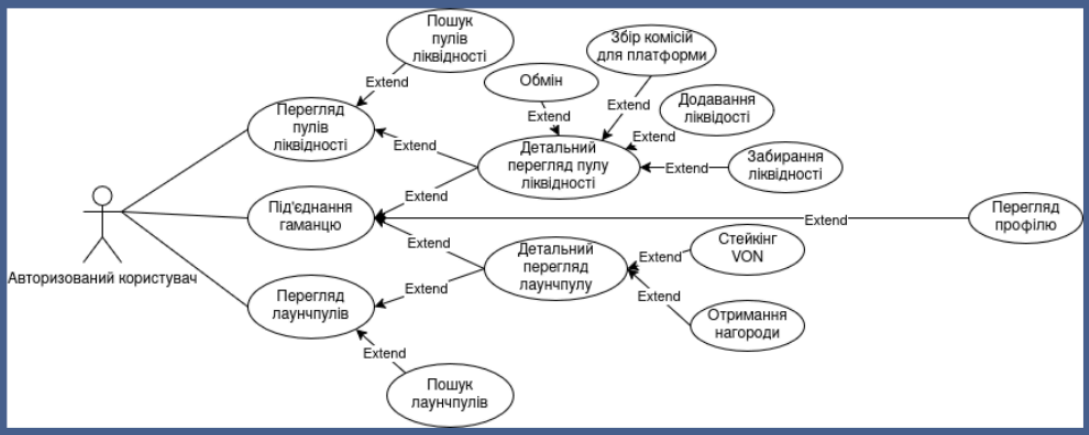
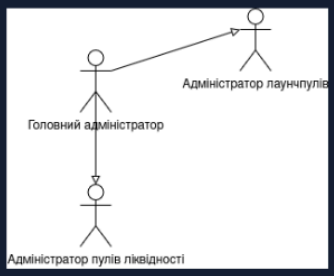
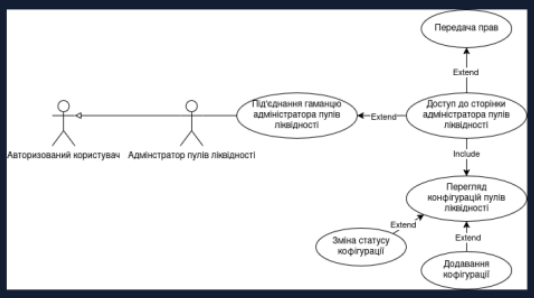


Рисунок Б.14 – Слайд №14

Діаграми прецедентів





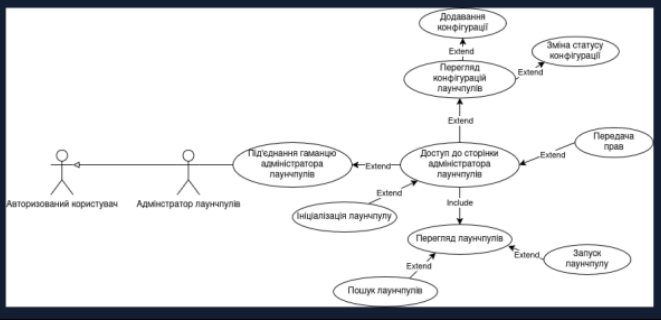


Рисунок Б.15 – Слайд №15

Приклад реалізації

```rust
pub(crate) fn handler(ctx: Context<SwapInCpAmm>, swap_amount: u64, estimated_result: u64, allowed_slippage: u64, is_in_out: bool) -> Result<()> {
    let in_transfer_instruction:Box<TransferTokensInstruction> = Box::new(x:ctx.accounts.get_in_transfer_instruction(swap_amount, is_in_out)?);

    let swap_amount_after_fee:u64 = in_transfer_instruction.get_amount_after_fee();
    let swap_payload:SwapPayload = ctx.accounts.cp_amm.get_swap_payload(
        swap_amount_after_fee,
        estimated_result,
        allowed_slippage,
        ctx.accounts.amms_config.providers_fee_rate_basis_points(),
        ctx.accounts.amms_config.protocol_fee_rate_basis_points(),
        is_in_out
    )?;
    let amount_to_withdraw:u64 = swap_payload.amount_to_withdraw();

    let out_transfer_instruction:Box<TransferTokensInstruction> = Box::new(x:ctx.accounts.get_out_transfer_instruction(swap_payload.amount_to_withdraw(), is_in_out)?);
    in_transfer_instruction.execute(None)?;
    let cp_amm_seeds:[&[u8];3] = ctx.accounts.cp_amm.seeds();
    let out_instruction_seeds: &[&[&[u8]]] = &[&cp_amm_seeds];
    out_transfer_instruction.execute(Some(out_instruction_seeds))?;

    ctx.accounts.cp_amm.swap(swap_payload);
    let cp_amm:&Box<Account<CpAmm>> = &ctx.accounts.cp_amm;
```

Рисунок Б.16 – Слайд №16

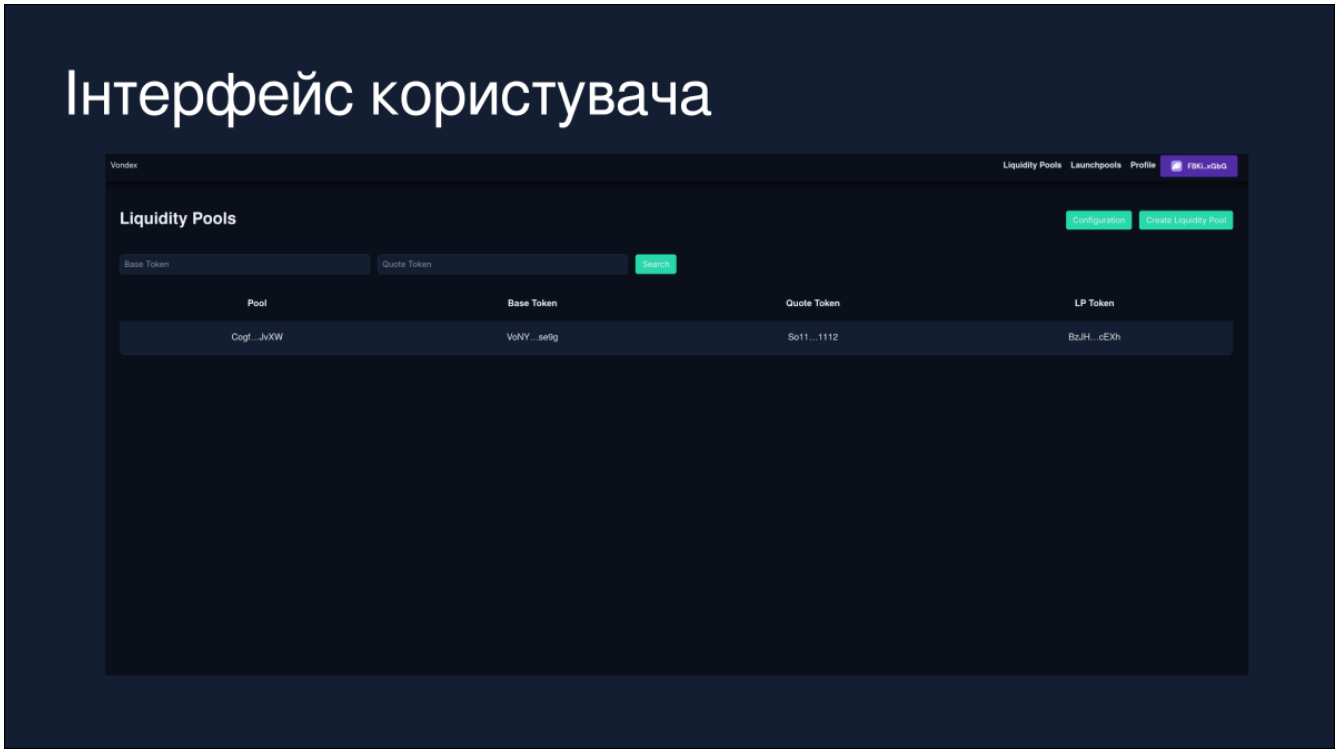


Рисунок Б.17 – Слайд №17

Рисунок Б.18 – Слайд №18

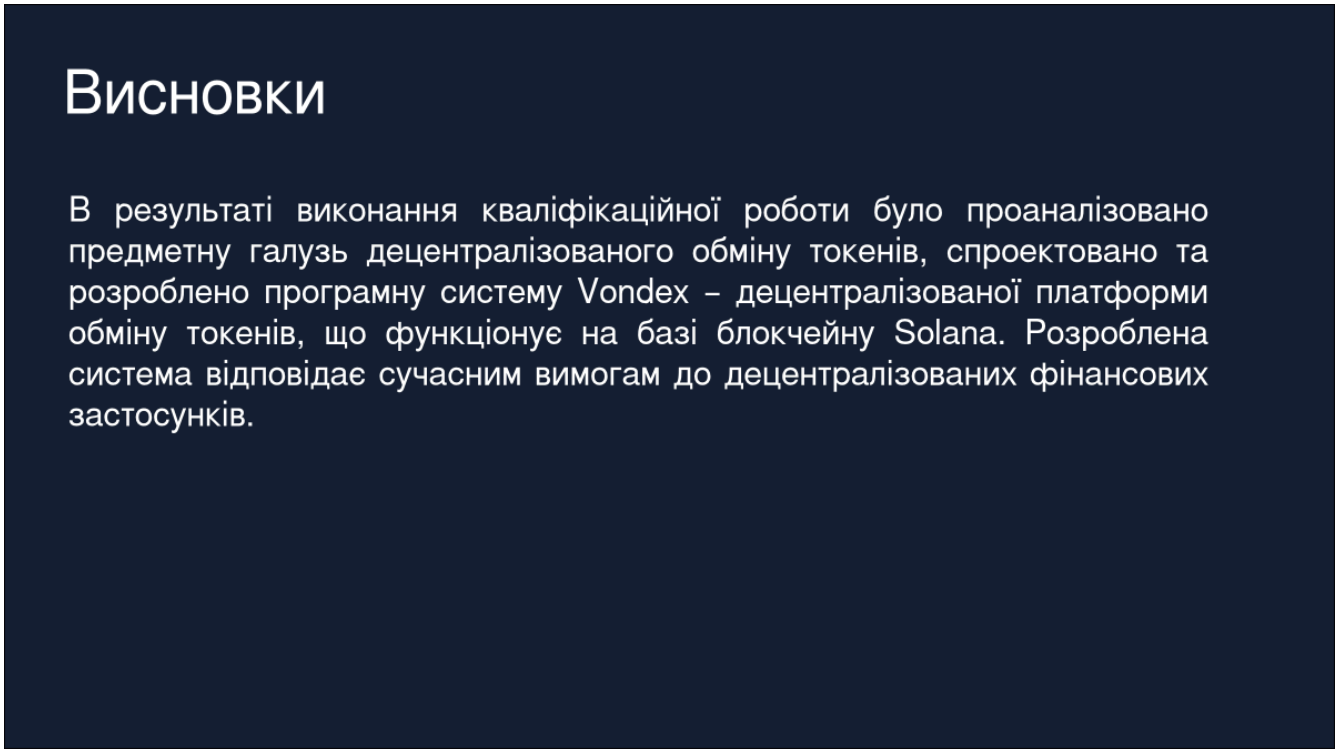


Рисунок Б.19 – Слайд №19

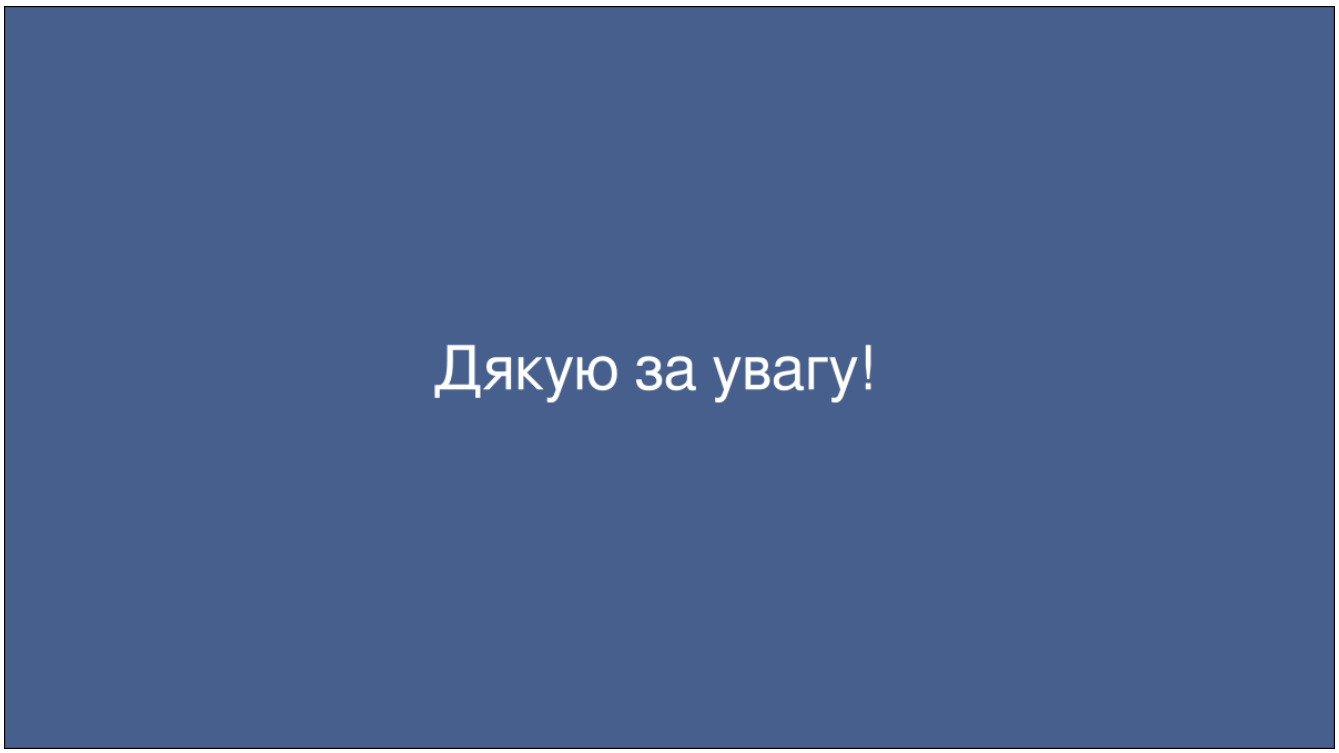


Рисунок Б.20 – Слайд №20

ДОДАТОК В

Приклад кодів програми

В.1 Реалізація типу Q64_128

```
use std::ops::{Add, Div, Mul, Sub};
use super::{U384, U192};

#[cfg(feature = "solana")]
use anchor_lang::{AnchorDeserialize, AnchorSerialize, prelude::borsh,
InitSpace};

/// Represents a fixed-point number with 64 integer bits and 128
fractional bits.
///
/// The `Q64_128` type is useful for high-precision arithmetic where
fractional values
/// need to be represented without losing precision. Internally, it
uses a 192-bit
/// unsigned integer (`U192`) to store the value, where the most
significant 64 bits
/// represent the integer part, and the least significant 128 bits
represent the fractional part.
///
///  This type provides utilities for fixed-point arithmetic,
conversions from primitive types,
/// and accessing the integer and fractional components of the value.
#[derive(Debug, Clone, Copy, PartialEq, PartialOrd, Default)]
#[cfg_attr(feature = "solana", derive(AnchorSerialize,
AnchorDeserialize, InitSpace))]
pub struct Q64_128 {
    /// The internal representation of the fixed-point value as a
192-bit unsigned integer.
    value: U192,
}

impl Q64_128 {
    /// The number of fractional bits (128).
    pub const FRACTIONAL_BITS: u32 = 128;

    /// The number of integer bits (64).
    pub const INTEGER_BITS: u32 = 64;

    /// The bitmask used to extract fractional bits.
    pub const FRACTIONAL_MASK: U192 = U192([u64::MAX, u64::MAX, 0]);

    /// The minimal fraction required for rounding.
    pub const ROUND_FRACTION: U192 = U192([0, 9223372036854775808,
0]);
    /*  const ROUND_FRACTION: U192 = U192([0,
18446744073709551360, 0]);*/

    /// The scaling factor used for converting floating-point values
to fixed-point.
```

```
    pub const FRACTIONAL_SCALE: f64 = 340282366920938463463374607431768211456.0;

    /// The maximum representable value for `Q64_128`.
    pub const MAX: Self = Q64_128::new(U192::MAX);

    /// A constant representing the value 1 in `Q64_128` format.
    pub const ONE: Self = Q64_128::from_u64(1u64);

    /// Creates a new `Q64_128` instance from a `U192` value.
    ///
    /// # Parameters
    /// - `value`: The raw 192-bit unsigned integer representing the fixed-point value.
    ///
    /// # Returns
    /// A new `Q64_128` instance.
    pub const fn new(value: U192) -> Self {
        Q64_128 { value }
    }

    /// Creates a `Q64_128` instance from an unsigned 64-bit integer.
    ///
    /// # Parameters
    /// - `value`: The integer to be converted to a fixed-point value.
    ///
    /// # Returns
    /// A `Q64_128` instance representing the integer as a fixed-point number.
    pub const fn from_u64(value: u64) -> Self {
        Self {
            value: U192([0, 0, value]),
        }
    }

    /// Creates a `Q64_128` instance from an unsigned 128-bit integer.
    ///
    /// # Parameters
    /// - `value`: The integer to be converted to a fixed-point value.
    ///
    /// # Returns
    /// A `Q64_128` instance representing the integer as a fixed-point number.
    pub const fn from_u128(value: u128) -> Self {
        let high = (value >> Self::INTEGER_BITS) as u64;
        let mid = (value << Self::INTEGER_BITS >> Self::INTEGER_BITS) as u64;
        Self {
            value: U192([0, mid, high]),
        }
    }
```

```
    /// Creates a `Q64_128` instance from high and low bits.
    ///
    /// # Parameters
    /// - `high_bits`: The 64 most significant bits representing the
integer part.
    /// - `low_bits`: The 128 least significant bits representing
the fractional part.
    ///
    /// # Returns
    /// A `Q64_128` instance combining the integer and fractional
parts.
    pub const fn from_bits(high_bits: u64, low_bits: u128) -> Self {
        Self::new(U192([
            (low_bits & (u64::MAX as u128)) as u64,
            (low_bits >> 64) as u64,
            high_bits,
        ]))
    }

    /// Converts the fixed-point value to a 64-bit unsigned integer.
    ///
    /// # Returns
    /// The integer part of the fixed-point value.
    pub fn as_u64(&self) -> u64 {
        (self.value >> Self::FRACTIONAL_BITS).as_u64()
    }

    /// Converts the fixed-point value to a 64-bit unsigned integer
with rounding.
    ///
    /// This method extracts the integer part of the fixed-point
value stored in the `value` field
    /// and rounds it up if the fractional part meets or exceeds the
rounding threshold.
    ///
    /// # Returns
    /// - A `u64` representing the rounded integer value of the
fixed-point number.
    pub fn as_u64_round(&self) -> u64 {
        let mut integer = (self.value >>
Self::FRACTIONAL_BITS).as_u64();
        if integer < u64::MAX && (self.value &
Self::FRACTIONAL_MASK >= Self::ROUND_FRACTION){
            integer += 1;
        }
        integer
    }

    /// Splits the fixed-point value into its integer and fractional
components.
    ///
    /// # Returns
    /// A tuple `(integer_part, fractional_part)`, where:
    /// - `integer_part` is the integer portion of the value.
    /// - `fractional_part` is the fractional portion as a 128-bit
unsigned integer.
    pub fn split(&self) -> (u64, u128) {
```

```
            (self.get_integer_bits(), self.get_fractional_bits())
        }

        /// Retrieves the raw 192-bit value of the fixed-point number.
        ///
        /// # Returns
        /// The raw `U192` value.
        pub(super) fn raw_value(&self) -> U192 {
            self.value
        }

        /// Extracts the integer bits from the fixed-point value.
        ///
        /// # Returns
        /// The integer part as a 64-bit unsigned integer.
        pub fn get_integer_bits(&self) -> u64 {
            (self.value >> Self::FRACTIONAL_BITS).as_u64()
        }

        /// Extracts the fractional bits from the fixed-point value.
        ///
        /// # Returns
        /// The fractional part as a 128-bit unsigned integer.
        pub fn get_fractional_bits(&self) -> u128 {
            (self.value & Self::FRACTIONAL_MASK).as_u128()
        }

        /// Checks if the value is zero.
        ///
        /// # Returns
        /// `true` if the value is zero, otherwise `false`.
        pub fn is_zero(&self) -> bool {
            self.value == U192::zero()
        }

        /// Checks if the value is one.
        ///
        /// # Returns
        /// `true` if the value is one, otherwise `false`.
        pub fn is_one(&self) -> bool {
            self.value == Q64_128::ONE.value
        }

        /// Creates a `Q64_128` instance from a `f64` value (for testing
purposes).
        ///
        /// # Parameters
        /// - `value`: A floating-point value to be converted to
`Q64_128`.
        ///
        /// # Returns
        /// An `Option<Q64_128>`:
        /// - `Some` containing the converted value if it is within the
valid range.
        /// - `None` if the value is outside the valid range for
`Q64_128`.
        ///
```

```
     /// # Valid Range
     /// - The value must be within `[0.0, 2^64)`.
     pub fn from_f64(value: f64) -> Option<Self> {
          if !(0.0..18446744073709551616.0).contains(&value) {
               return None;
          }
          let integer_part = value.floor() as u64;
          let fractional_part = ((value - integer_part as f64) *
Self::FRACTIONAL_SCALE).round() as u128;
          Some(Self::from_bits(integer_part, fractional_part))
     }

     /// Serializes the Q64_128 value to a 24-byte little-endian
array.
     pub fn to_le_bytes(&self) -> [u8; 24] {
          self.value.to_little_endian()
     }

     /// Deserializes a 24-byte little-endian array into Q64_128.
     pub fn from_le_bytes(bytes: [u8; 24]) -> Self {
          Self::new(U192::from_little_endian(&bytes))
     }

     /// Serializes the Q64_128 value to a 24-byte big-endian array.
     pub fn to_be_bytes(&self) -> [u8; 24] {
          self.value.to_big_endian()
     }

     /// Deserializes a 24-byte big-endian array into Q64_128.
     pub fn from_be_bytes(bytes: [u8; 24]) -> Self {
          Self::new(U192::from_little_endian(&bytes))
     }

}

/// Implements conversion from `U384` to `Q64_128`.
///
/// # Panic
/// - Panics if the `U384` value cannot fit into 192 bits. This occurs
when
///    the number of leading zeros in the `U384` value is less than
192.
///
/// # Parameters
/// - `value`: A `U384` value to be converted into `Q64_128`.
///
/// # Returns
/// A `Q64_128` instance representing the input `U384` value.
impl From<U384> for Q64_128 {
     fn from(value: U384) -> Self {
          if value.leading_zeros() < 192 {
               panic!("Cannot convert U384 to Q64_128");
          }
          let                          u192                          =
U192::from_big_endian(&(value.to_big_endian()[24..]));
          Self::new(u192)
     }
```

```
}

/// Implements conversion from `Q64_128` to `f64` (testing only).
///
/// # Notes
/// This implementation is provided for testing purposes to convert a
`Q64_128`
/// value into a floating-point number. It separates the integer and
fractional
/// parts and combines them to form the `f64` value.
///
/// # Parameters
/// - `value`: A `Q64_128` instance to be converted into a `f64`.
///
/// # Returns
/// A `f64` representation of the input `Q64_128`.
impl From<Q64_128> for f64 {
     fn from(value: Q64_128) -> Self {
          let (high_bits, low_bits) = value.split();
          let    fractional_part    =    (low_bits    as    f64)    /
Q64_128::FRACTIONAL_SCALE;
          high_bits as f64 + fractional_part
     }
}

/// Implements addition for `Q64_128`.
///
/// # Behavior
/// Adds two `Q64_128` values together by performing an addition on
their raw
/// `U192` representations.
///
/// # Returns
/// A new `Q64_128` instance containing the sum of the two inputs.
impl Add for Q64_128 {
     type Output = Self;

     fn add(self, rhs: Self) -> Self::Output {
          let result = self.value + rhs.value;
          Self { value: result }
     }
}

/// Implements subtraction for `Q64_128`.
///
/// # Behavior
///  Subtracts  one  `Q64_128`  value  from  another  by  performing  a
subtraction on their
/// raw `U192` representations.
///
/// # Returns
/// A new `Q64_128` instance containing the result of the subtraction.
impl Sub for Q64_128 {
     type Output = Self;

     fn sub(self, rhs: Self) -> Self::Output {
          Self {
```

```
                value: self.value - rhs.value,
            }
        }
}

/// Implements multiplication for `Q64_128`.
///
/// # Behavior
/// Multiplies two `Q64_128` values by converting them to `U384`,
performing the
/// multiplication, and then shifting the result to adjust for the
fractional bits.
///
/// # Returns
/// A new `Q64_128` instance containing the product of the two inputs.
impl Mul for Q64_128 {
        type Output = Self;

        fn mul(self, rhs: Self) -> Self::Output {
            let result = (U384::from(self) * U384::from(rhs)) >>
Q64_128::FRACTIONAL_BITS;
            result.into()
        }
}

/// Implements division for `Q64_128`.
///
/// # Behavior
/// Divides one `Q64_128` value by another by converting them to
`U384`, performing
/// the division, and adjusting for the fractional bits by shifting
the numerator.
///
/// # Panic
/// - Panics if the divisor (`rhs`) is zero to avoid division by zero.
///
/// # Returns
/// A new `Q64_128` instance containing the result of the division.
impl Div for Q64_128 {
        type Output = Self;

        fn div(self, rhs: Self) -> Self::Output {
            if rhs.is_zero() {
                panic!("Division by zero!");
            }
            let result = (U384::from(self) << Self::FRACTIONAL_BITS) /
U384::from(rhs);
            result.into()
        }
}

impl Q64_128 {
        /// Calculates the absolute difference between two `Q64_128`
values.
        ///
        /// # Parameters
        /// - `self`: The first `Q64_128` value.
```

```
    /// - `other`: The second `Q64_128` value.
    ///
    /// # Returns
    /// A new `Q64_128` instance representing the absolute
difference between `self` and `other`.
    pub fn abs_diff(self, other: Self) -> Q64_128 {
        Q64_128::new(self.value.abs_diff(other.value))
    }

    /// Performs a checked addition of two `Q64_128` values.
    ///
    /// # Parameters
    /// - `self`: The first `Q64_128` value.
    /// - `rhs`: The second `Q64_128` value to add to `self`.
    ///
    /// # Returns
    /// An `Option<Q64_128>`:
    /// - `Some(Q64_128)` containing the result if the addition does
not overflow.
    /// - `None` if the addition overflows.
    pub fn checked_add(self, rhs: Self) -> Option<Self> {
        self.value.checked_add(rhs.value).map(|res| Self { value:
res })
    }

    /// Performs a checked subtraction of two `Q64_128` values.
    ///
    /// # Parameters
    /// - `self`: The first `Q64_128` value.
    /// - `rhs`: The second `Q64_128` value to subtract from `self`.
    ///
    /// # Returns
    /// An `Option<Q64_128>`:
    /// - `Some(Q64_128)` containing the result if the subtraction
does not underflow.
    /// - `None` if the subtraction underflows.
    pub fn checked_sub(self, rhs: Self) -> Option<Self> {
        self.value.checked_sub(rhs.value).map(|res| Self { value:
res })
    }

    /// Performs a checked multiplication of two `Q64_128` values.
    ///
    /// # Parameters
    /// - `self`: The first `Q64_128` value.
    /// - `rhs`: The second `Q64_128` value to multiply with `self`.
    ///
    /// # Returns
    /// An `Option<Q64_128>`:
    /// - `Some(Q64_128)` containing the result if the
multiplication does not overflow.
    /// - `None` if the multiplication overflows.
    pub fn checked_mul(self, rhs: Self) -> Option<Self> {
        ((U384::from(self) * U384::from(rhs)) >>
Q64_128::FRACTIONAL_BITS).checked_as_q64_128()
    }
```

```
    /// Performs a checked division of two `Q64_128` values.
    ///
    /// # Parameters
    /// - `self`: The numerator `Q64_128` value.
    /// - `rhs`: The denominator `Q64_128` value.
    ///
    /// # Returns
    /// An `Option<Q64_128>`:
    /// - `Some(Q64_128)` containing the result if the division is
valid.
    /// - `None` if the denominator is zero.
    pub fn checked_div(self, rhs: Self) -> Option<Self> {
        if rhs.is_zero() {
            return None;
        }
        let result = (U384::from(self) << Self::FRACTIONAL_BITS) /
U384::from(rhs);
        result.checked_as_q64_128()
    }

    /// Performs a saturating multiplication of two `Q64_128`
values.
    ///
    /// # Parameters
    /// - `self`: The first `Q64_128` value.
    /// - `rhs`: The second `Q64_128` value to multiply with `self`.
    ///
    /// # Returns
    /// A `Q64_128` instance:
    /// - The product of the two values if the multiplication does
not overflow.
    /// - `Q64_128::MAX` if the multiplication overflows.
    pub fn saturating_mul(self, rhs: Self) -> Self {
        self.checked_mul(rhs).map_or(Q64_128::MAX, |res| res)
    }

    /// Performs a saturating division of two `Q64_128` values with
overflow handling.
    ///
    /// # Parameters
    /// - `self`: The numerator `Q64_128` value.
    /// - `rhs`: The denominator `Q64_128` value.
    ///
    /// # Returns
    /// An `Option<Q64_128>`:
    /// - `Some(Q64_128)` containing the result of the division.
    /// - `None` if the denominator is zero.
    /// - `Some(Q64_128::MAX)` if the division overflows.
    pub fn saturating_checked_div(self, rhs: Self) -> Option<Self> {
        if rhs.is_zero() {
            return None;
        }
        let result = (U384::from(self) << Self::FRACTIONAL_BITS) /
U384::from(rhs);
        result.checked_as_q64_128().map_or(Some(Q64_128::MAX),
Some)
    }
```

```
}


impl Q64_128 {
      /// Calculates the square of the `Q64_128` value.
      ///
      /// # Behavior
      /// - Computes the square of the value as a fixed-point number.
      /// - Returns `None` if the result cannot fit within the bounds
of `Q64_128`.
      ///
      /// # Returns
      /// An `Option<Q64_128>`:
      /// - `Some(Q64_128)` containing the square of the value.
      /// - `None` if the result overflows.
      #[cfg(test)]
      fn square(self) -> Option<Self> {
            let square_as_u384 = self.square_as_u384();
            if self.is_zero() || self.is_one() {
                  return Some(self);
            }
            square_as_u384.q128_256_to_q64_128_round()
      }

      /// Computes the square of the `Q64_128` value as a `U384`.
      ///
      /// # Behavior
      /// - Multiplies the value by itself and returns the result as a
`U384`.
      ///
      /// # Returns
      /// A `U384` representing the square of the value.
      fn square_as_u384(self) -> U384 {
            if self.is_zero() {
                  return U384::zero();
            }
            if self.is_one() {
                  return U384::from(1u128);
            }
            U384::from(self) * U384::from(self)
      }

      /// Computes the square of the `Q64_128` value as a `u128`.
      ///
      /// # Behavior
      /// - Computes the square and rounds the result to fit within a
`u128`.
      ///
      /// # Returns
      /// A `u128` representing the square of the value.
      pub fn square_as_u128(self) -> u128 {
            let square_as_u384 = self.square_as_u384();
            if square_as_u384 == U384::one() || self.is_zero() {
                  return square_as_u384.as_u128();
            }
            square_as_u384.q128_256_to_u128_round()
      }
```

```
    /// Computes the square of the `Q64_128` value as a `u64` if
possible.
    ///
    /// # Behavior
    /// - Computes the square and rounds the result to fit within a
`u64`.
    /// - Returns `None` if the result overflows a `u64`.
    ///
    /// # Returns
    /// An `Option<u64>`:
    /// - `Some(u64)` containing the rounded square of the value.
    /// - `None` if the result overflows a `u64`.
    pub fn checked_square_as_u64(self) -> Option<u64> {
        let square_as_u384 = self.square_as_u384();
        if square_as_u384 == U384::one() || self.is_zero() {
            return Some(square_as_u384.as_u64());
        }
        square_as_u384.checked_q128_256_to_u64_round()
    }

    /// Computes the square of the `Q64_128` value as a `u64`.
    ///
    /// # Behavior
    /// - Computes the square and rounds the result to fit within a
`u64`.
    /// - Panics if the result overflows a `u64`.
    ///
    /// # Returns
    /// A `u64` representing the square of the value.
    pub fn square_as_u64(self) -> u64 {
        self.checked_square_as_u64().unwrap()
    }

    /// Computes the square root of a `u128` value and scales it to
`Q64_128`.
    ///
    /// # Parameters
    /// - `value`: A `u128` value for which the square root is to be
computed.
    ///
    /// # Returns
    /// A `Q64_128` instance representing the square root of the
input value.
    pub fn sqrt_from_u128(value: u128) -> Self {
        let scaled_value = U384::from(value) << (2 *
Self::FRACTIONAL_BITS);
        Q64_128::from(scaled_value.integer_sqrt())
    }

    /// Computes the square root of the `Q64_128` value.
    ///
    /// # Behavior
    /// - Computes the square root of the value as a fixed-point
number.
    ///
    /// # Returns
```

```
    /// A `Q64_128` instance representing the square root of the
value.
    pub fn sqrt(self) -> Self {
        let scaled_value = U384::from(self) <<
Self::FRACTIONAL_BITS;
        Q64_128::from(scaled_value.integer_sqrt())
    }

    /// Computes the square root of the division of two `Q64_128`
values.
    ///
    /// # Parameters
    /// - `q1`: The numerator `Q64_128` value.
    /// - `q2`: The denominator `Q64_128` value.
    ///
    /// # Behavior
    /// - Computes the division of `q1` by `q2` and then takes the
square root of the result.
    /// - Returns `None` if `q2` is zero.
    /// - If `q1` equals `q2`, the result is `Q64_128::ONE`.
    ///
    /// # Returns
    /// An `Option<Q64_128>`:
    /// - `Some(Q64_128)` containing the square root of the division
result.
    /// - `None` if the division is invalid (e.g., division by
zero).
    pub fn checked_div_sqrt(q1: Self, q2: Self) -> Option<Self> {
        if q2.is_zero() {
            return None;
        } else if q1 == q2 {
            return Some(Q64_128::ONE);
        }
        let division_result = U384::from(q1.checked_div(q2)?);
        let result = Q64_128::from((division_result <<
Self::FRACTIONAL_BITS).integer_sqrt());
        Some(result)
    }
}
```

В.2 Інструкції смартконтракту пулів ліквідності

```
use anchor_lang::prelude::*;

declare_id!("2M2QKXZuuERizynTpUwfD7FkdhKHWAFVKiCFGBSxXr3X");

pub mod constants;
pub mod error;
pub mod instructions;
pub mod state;

pub use instructions::*;
#[program]
pub mod liquidity_pool {
      use super::*;

      pub fn initialize_amms_configs_manager(ctx:
Context<InitializeAmmsConfigsManager>) -> Result<()>{
      msg!("Instruction: InitializeAmmsConfigsManager");
      initialize_amms_configs_manager::handler(ctx)
      }

      pub fn update_amms_configs_manager_authority(ctx:
Context<UpdateAmmsConfigsManagerAuthority>) -> Result<()>{
      msg!("Instruction: UpdateAmmsConfigsManagerAuthority");
      update_amms_configs_manager_authority::handler(ctx)
      }

      pub fn update_amms_configs_manager_head_authority(ctx:
Context<UpdateAmmsConfigsManagerHeadAuthority>) -> Result<()>{
      msg!("Instruction: UpdateAmmsConfigsManagerHeadAuthority");
      update_amms_configs_manager_head_authority::handler(ctx)
      }


      pub fn initialize_amms_config(ctx:
Context<InitializeAmmsConfig>, protocol_fee_rate_basis_points: u16,
providers_fee_rate_basis_points: u16) -> Result<()>{
      msg!("Instruction: InitializeAmmsConfig");
      initialize_amms_config::handler(ctx,
protocol_fee_rate_basis_points, providers_fee_rate_basis_points)
      }

      pub fn update_amms_config_fee_authority(ctx:
Context<UpdateAmmsConfigFeeAuthority>) -> Result<()>{
      msg!("Instruction: UpdateAmmsConfigFeeAuthority");
      update_amms_config_fee_authority::handler(ctx)
      }

      pub fn update_amms_config_providers_fee_rate(ctx:
Context<UpdateAmmsConfigProvidersFeeRate>,
new_providers_fee_rate_basis_points: u16) -> Result<()>{
      msg!("Instruction: UpdateAmmsConfigProvidersFeeRate");
      update_amms_config_providers_fee_rate::handler(ctx,
new_providers_fee_rate_basis_points)
```

```
    }

    pub fn update_amms_config_protocol_fee_rate(ctx:
Context<UpdateAmmsConfigProtocolFeeRate>,
new_protocol_fee_rate_basis_points: u16) -> Result<()>{
    msg!("Instruction: UpdateAmmsConfigProtocolFeeRate");
    update_amms_config_protocol_fee_rate::handler(ctx,
new_protocol_fee_rate_basis_points)
    }

    pub fn initialize_cp_amm(ctx: Context<InitializeCpAmm>) ->
Result<()>{
    msg!("Instruction: InitializeCpAmm");
    initialize_cp_amm::handler(ctx)
    }
    pub fn launch_cp_amm(ctx: Context<LaunchCpAmm>, base_liquidity:
u64, quote_liquidity: u64) -> Result<()>{
    msg!("Instruction: LaunchCpAmm");
    launch_cp_amm::handler(ctx, base_liquidity, quote_liquidity)
    }
    pub fn provide_to_cp_amm(ctx: Context<ProvideToCpAmm>,
base_liquidity: u64, quote_liquidity: u64) -> Result<()>{
    msg!("Instruction: ProvideToCpAmm");
    provide_to_cp_amm::handler(ctx, base_liquidity, quote_liquidity)
    }
    pub fn withdraw_from_cp_amm(ctx: Context<WithdrawFromCpAmm>,
lp_tokens: u64) -> Result<()>{
    msg!("Instruction: WithdrawFromCpAmm");
    withdraw_from_cp_amm::handler(ctx, lp_tokens)
    }
    pub fn swap_in_cp_amm(ctx: Context<SwapInCpAmm>, swap_amount:
u64, estimated_result: u64, allowed_slippage: u64, is_in_out: bool) ->
Result<()>{
    msg!("Instruction: SwapInCpAmm");
    swap_in_cp_amm::handler(ctx, swap_amount, estimated_result,
allowed_slippage, is_in_out)
    }
    pub fn collect_fees_from_cp_amm(ctx:
Context<CollectFeesFromCpAmm>) -> Result<()>{
    msg!("Instruction: CollectFeesFromCpAmm");
    collect_fees_from_cp_amm::handler(ctx)
    }
}
```

В.3 Інструкції смартконтракту лаунчпулів

```
use anchor_lang::prelude::*;
declare_id!("5M9TeHHBeAtUd956yRUW9TEULF5XqGUdcyfy74YDzXHU");

mod error;
mod instructions;
mod state;

pub use instructions::*;

#[program]
pub mod launchpool {
      use super::*;
      pub fn initialize_launchpools_config(ctx:
Context<InitializeLaunchpoolsConfig>, min_position_size: u64,
max_position_size: u64, protocol_reward_share_basis_points: u16,
duration: u64) -> Result<()>{
      msg!("Instruction: InitializeLaunchpoolsConfig");
      initialize_launchpools_config::handler(ctx, min_position_size,
max_position_size, protocol_reward_share_basis_points, duration)
      }

      pub fn initialize_launchpools_configs_manager(ctx:
Context<InitializeLaunchpoolsConfigsManager>) -> Result<()>{
      msg!("Instruction: InitializeLaunchpoolsConfigsManager");
      initialize_launchpools_configs_manager::handler(ctx)
      }

      pub fn update_launchpools_config_duration(ctx:
Context<UpdateLaunchpoolsConfigDuration>, new_duration: u64) ->
Result<()>{
      msg!("Instruction: UpdateLaunchpoolsConfigDuration");
      update_launchpools_config_duration::handler(ctx, new_duration)
      }

      pub fn update_launchpools_config_position_sizes(ctx:
Context<UpdateLaunchpoolsConfigPositionSizes>, new_min_position_size:
u64, new_max_position_size: u64) -> Result<()>{
      msg!("Instruction: UpdateLaunchpoolsConfigPositionSizes");
      update_launchpools_config_position_sizes::handler(ctx,
new_min_position_size, new_max_position_size)
      }

      pub fn update_launchpools_config_protocol_reward_share(ctx:
Context<UpdateLaunchpoolsConfigProtocolRewardShare>,
new_protocol_reward_share_basis_points: u16) -> Result<()>{
      msg!("Instruction: UpdateLaunchpoolsConfigProtocolRewardShare");
      update_launchpools_config_protocol_reward_share::handler(ctx,
new_protocol_reward_share_basis_points)
      }

      pub fn update_launchpools_config_reward_authority(ctx:
Context<UpdateLaunchpoolsConfigRewardAuthority>) -> Result<()>{
      msg!("Instruction: UpdateLaunchpoolsConfigRewardAuthority");
```

```
        update_launchpools_config_reward_authority::handler(ctx)
        }

        pub fn update_launchpools_configs_manager_authority(ctx:
Context<UpdateLaunchpoolsConfigsManagerAuthority>) -> Result<()>{
        msg!("Instruction: UpdateLaunchpoolsConfigsManagerAuthority");
        update_launchpools_configs_manager_authority::handler(ctx)
        }

        pub fn update_launchpools_configs_manager_head_authority(ctx:
Context<UpdateLaunchpoolsConfigsManagerHeadAuthority>) -> Result<()>{
        msg!("Instruction:
UpdateLaunchpoolsConfigsManagerHeadAuthority");
        update_launchpools_configs_manager_head_authority::handler(ctx)
        }

        pub fn initialize_launchpool(ctx: Context<InitializeLaunchpool>,
initial_reward_amount: u64) -> Result<()>{
        msg!("Instruction: InitializeLaunchpool");
        initialize_launchpool::handler(ctx, initial_reward_amount)
        }

        pub fn launch_launchpool(ctx: Context<LaunchLaunchpool>,
start_timestamp: u64) -> Result<()>{
        msg!("Instruction: LaunchLaunchpool");
        launch_launchpool::handler(ctx, start_timestamp)
        }

        pub fn open_stake_position(ctx: Context<OpenStakePosition>,
stake_amount: u64) -> Result<()>{
        msg!("Instruction: OpenStakePosition");
        open_stake_position::handler(ctx, stake_amount)
        }

        pub fn increase_stake_position(ctx:
Context<IncreaseStakePosition>, stake_increase_amount: u64) ->
Result<()>{
        msg!("Instruction: IncreaseStakePosition");
        increase_stake_position::handler(ctx, stake_increase_amount)
        }

        pub fn close_stake_position(ctx: Context<CloseStakePosition>) ->
Result<()>{
        msg!("Instruction: CloseStakePosition");
        close_stake_position::handler(ctx)
        }

        pub fn collect_protocol_reward(ctx:
Context<CollectProtocolReward>) -> Result<()>{
        msg!("Instruction: CollectProtocolReward");
        collect_protocol_reward::handler(ctx)
        }
}
```

В.4 Скрипт для генерації клієнтів смартконтрактів

```
import {AnchorIdl, rootNodeFromAnchor} from
"@codama/nodes-from-anchor";
import {renderJavaScriptVisitor, renderRustVisitor} from
"@codama/renderers";
import liquidityPoolIdl from '../target/idl/liquidity_pool.json';
import launchpoolIdl from '../target/idl/launchpool.json';
import * as path from "node:path";
import fg from "fast-glob";
import fs from "fs-extra";
import {createFromRoot} from "codama";

async function generateClients() {
      const codamaLiquidityPool =
createFromRoot(rootNodeFromAnchor(liquidityPoolIdl as AnchorIdl));
      const codamaLaunchpool =
createFromRoot(rootNodeFromAnchor(launchpoolIdl as AnchorIdl));
      const root = "../onchain-clients"
      const clients = [
      {
            type: "JS",
            dir: `${root}/liquidity-pool/js`,
            renderVisitor: renderJavaScriptVisitor,
            codama: codamaLiquidityPool,
            name: "@liquidity-pool/js"
      },
      {
            type: "Rust",
            dir: `${root}/liquidity-pool/rust/src`,
            renderVisitor: renderRustVisitor,
            codama: codamaLiquidityPool,
            name: "liquidity-pool"
      },
      {
            type: "JS",
            dir: `${root}/launchpool/js`,
            renderVisitor: renderJavaScriptVisitor,
            codama: codamaLaunchpool,
            name: "@launchpool/js"
      },
      {
            type: "Rust",
            dir: `${root}/launchpool/rust/src`,
            renderVisitor: renderRustVisitor,
            codama: codamaLaunchpool,
            name: "launchpool"
      }
      ];
      let rust_libs_path: string[] = [];
      for (const client of clients) {
      try {
            await client.codama.accept(
                  await client.renderVisitor(client.dir)
            );
```

```
            if (client.type == "JS") {
                  await generatePackageJson(client.dir, client.name);
            } else if (client.type == "Rust") {
                  rust_libs_path.push(client.name)
                  await generateCargoToml(client.dir, client.name);
            }
            console.log(`✅ Successfully generated ${client.type}
client for ${client.name} in directory: ${client.dir}!`);
      } catch (e) {
            console.error(`Error in ${client.renderVisitor.name}:`, e);
            throw e;
      }
      }
      await generateWorkspaceCargoToml(root, rust_libs_path)
}

async function generatePackageJson(dir: string, name: string) {
      const packageJsonPath = path.join(dir, "package.json");
      const pkg = {
      name,
      version: "1.0.0",
      main: "index.ts",
      types: "index.ts",
      license: "MIT",
      private: true
      };
      fs.writeFileSync(packageJsonPath, JSON.stringify(pkg, null, 2));
      console.log(`📦 Created package.json for ${name}`);

}
async function generateCargoToml(dir: string, name: string) {
      const cargoTomlPath = path.join(`${dir}/..`, "Cargo.toml");
      const toml = `[package]
      name = "${name}"
      version = "1.0.0"
      edition = "2021"

      [lib]
      path = "src/lib.rs"

      [features]
      default = []
      fetch = ["solana-client", "solana-sdk"]

      [dependencies]
      borsh.workspace = true
      solana-program.workspace = true
      serde.workspace = true
      num-derive.workspace = true
      thiserror.workspace = true
      num-traits.workspace = true
      solana-client = { workspace = true, optional = true }
      solana-sdk = { workspace = true, optional = true }
      `;
      fs.writeFileSync(cargoTomlPath, toml);
      fs.renameSync(`${dir}/mod.rs`, `${dir}/lib.rs`);
      const content = fs.readFileSync(`${dir}/lib.rs`, 'utf-8');
```

```
      fs.writeFileSync(`${dir}/lib.rs`, '#![allow(warnings)]\n' +
content);


      const files = await fg([`${dir}/**/*.rs`]);

      for (const file of files) {
      const fullPath = path.resolve(file);
      const content = await fs.readFile(fullPath, "utf-8");

      const replaced = content.replace(/use\s+crate::generated::/g,
"use crate::");

      if (content !== replaced) {
            await fs.writeFile(fullPath, replaced, "utf-8");
      }
      }

      console.log(`📦 Created Cargo.toml for ${name}`);

}

async function generateWorkspaceCargoToml(dir: string, libs: string[])
{
      if (libs.length == 0) {
      return;
      }
      const cargoTomlPath = path.join(`${dir}`, "Cargo.toml");
      const toml = `
      [workspace]
      members = [
            ${libs.map(lib => `   "${lib}/rust"`).join(',\n')}
      ]

      [workspace.dependencies]
      borsh = "1.5.7"
      solana-program = "2.2.1"
      serde = "1.0.219"
      num-derive = "0.4.2"
      thiserror = "2.0.12"
      num-traits = "0.2.19"
      solana-client = "2.2.7"
      solana-sdk = "2.2.2"
      `;
      fs.writeFileSync(cargoTomlPath, toml);
}
generateClients();
```

В.5 Макрос для десеріалізації подій

```
use anyhow::{Result as AnyResult};
use solana_sdk::pubkey::Pubkey;

pub trait AnchorProgram: Sized + Send + Sync {
      const PUBKEY: Pubkey;
      fn try_deserialize(data: &[u8]) -> AnyResult<Self>;
}

#[macro_export]
macro_rules! define_program_events_enum {
      (
      $pubkey:expr,
      $(#[$outer:meta])*
      $program_vis:vis enum $program:ident {
            $(
                  $event:ident = $discriminator:expr
            ),+ $(,)?
      }
      ) => {
      use paste::paste;
      use anyhow::{Result as AnyResult, anyhow};
      use borsh::de::BorshDeserialize;
      use solana_sdk::pubkey::Pubkey;
      use AnchorProgram;
      $(#[$outer])*
      $program_vis enum $program{
            $(
                  $event(Box<$event>)
            ),+
      }

      impl $program{
            paste! {
                  $(
                  const [<$event _DISCRIMINATOR>]: [u8; 8] =
$discriminator;
                  )+
            }
      }

      impl AnchorProgram for $program{
            const PUBKEY: Pubkey = $pubkey;
            fn try_deserialize(data: &[u8]) -> AnyResult<Self> {
                  let discriminator_slice =
data.get(0..8).ok_or_else(|| anyhow!("Insufficient data for event
discriminator"))?;

                  paste! {
                  $(
                        if discriminator_slice == &$program::[<$event
_DISCRIMINATOR>] {
                              let event =
<$event>::try_from_slice(&data[8..])?;
```

```
                                return
Ok($program::$event(Box::new(event)));
                            }
                    )+
                    }
                    Err(anyhow!("Unknown event discriminator for {}",
stringify!($program)))
                }
        }
        }
}

define_program_events_enum! {
        LAUNCHPOOL_ID,
        #[derive(Debug)]
        pub enum LaunchpoolProgram {
        OpenStakePositionEvent = [43, 163, 16, 37, 74, 4, 209, 161],
        IncreaseStakePositionEvent = [121, 133, 109, 216, 234, 229, 196,
202],
        CloseStakePositionEvent = [100, 168, 243, 5, 211, 21, 49, 217],
        CollectProtocolRewardEvent = [205, 32, 118, 106, 76, 207, 44,
80],
        LaunchLaunchpoolEvent = [157, 245, 31, 39, 189, 53, 99, 115],
        InitializeLaunchpoolEvent = [135, 225, 199, 2, 42, 67, 97, 45],
        InitializeLaunchpoolsConfigEvent = [191, 79, 44, 239, 5, 100,
108, 4],
        UpdateLaunchpoolsConfigRewardAuthorityEvent = [41, 93, 234, 192,
147, 225, 218, 156],
        UpdateLaunchpoolsConfigProtocolRewardShareEvent = [28, 94, 107,
85, 139, 71, 180, 59],
        UpdateLaunchpoolsConfigPositionSizesEvent = [190, 75, 204, 106,
214, 22, 190, 193],
        UpdateLaunchpoolsConfigDurationEvent = [207, 214, 158, 69, 198,
68, 179, 48],
        UpdateLaunchpoolsConfigsManagerAuthorityEvent = [2, 12, 242,
131, 70, 205, 239, 249],
        UpdateLaunchpoolsConfigsManagerHeadAuthorityEvent = [58, 215,
167, 123, 90, 21, 139, 104],
        InitializeLaunchpoolsConfigsManagerEvent = [73, 78, 194, 10, 22,
3, 125, 192],
        }
}

define_program_events_enum! {
        LIQUIDITY_POOL_ID,
        #[derive(Debug)]
        pub enum LiquidityPoolProgram {
        SwapInCpAmmEvent = [167, 90, 102, 132, 142, 199, 241, 241],
        ProvideToCpAmmEvent = [169, 179, 105, 2, 40, 101, 75, 46],
        WithdrawFromCpAmmEvent = [20, 17, 220, 146, 91, 169, 183, 30],
        CollectFeesFromCpAmmEvent = [136, 202, 5, 125, 123, 107, 91,
113],
        LaunchCpAmmEvent = [185, 17, 120, 196, 33, 27, 224, 149],
        InitializeCpAmmEvent = [169, 188, 54, 67, 1, 145, 213, 80],
        UpdateAmmsConfigFeeAuthorityEvent = [145, 84, 143, 149, 33, 46,
208, 235],
```

```
      UpdateAmmsConfigProtocolFeeRateEvent = [122, 157, 87, 60, 236,
113, 198, 207],
      UpdateAmmsConfigProvidersFeeRateEvent = [182, 212, 34, 247, 179,
94, 71, 148],
      UpdateAmmsConfigsManagerAuthorityEvent = [87, 111, 229, 185, 38,
229, 136, 227],
      UpdateAmmsConfigsManagerHeadAuthorityEvent = [36, 151, 67, 108,
246, 99, 170, 92],
      InitializeAmmsConfigEvent = [138, 41, 61, 174, 151, 6, 209,
181],
      InitializeAmmsConfigsManagerEvent = [99, 45, 79, 86, 159, 151,
244, 154]
      }
}
```

B.6 Скрипт ініціалізації ScyllaDB

```
CREATE KEYSPACE IF NOT EXISTS launchpool_data
WITH replication = {
    'class': 'SimpleStrategy',
    'replication_factor': 1
    };

USE launchpool_data;

CREATE TABLE IF NOT EXISTS
initialize_launchpools_configs_manager_events
(
     signature        text PRIMARY KEY,
     timestamp        bigint,
     signer           text,
     authority        text,
     head_authority text
);

CREATE TABLE IF NOT EXISTS update_lp_cfg_mgr_auth_events
(
     signature  text PRIMARY KEY,
     timestamp  bigint,
     authority  text,
     new_authority text
);

CREATE TABLE IF NOT EXISTS update_lp_cfg_mgr_head_auth_events
(
     signature        text PRIMARY KEY,
     timestamp        bigint,
     head_authority   text,
     new_head_authority text
);

CREATE TABLE IF NOT EXISTS initialize_launchpools_config_events
(
     signature                          text,
     timestamp                          bigint,
     authority                          text,
     launchpools_config                 text PRIMARY KEY,
     reward_authority                   text,
     stakable_mint                      text,
     min_position_size                  blob,
     max_position_size                  blob,
     protocol_reward_share_basis_points smallint,
     duration                           blob,
     id                                 bigint,
);

CREATE TABLE IF NOT EXISTS update_launchpools_config_duration_events
(
     signature        text,
     timestamp        bigint,
```

```
      event_id         timeuuid,
      authority        text,
      launchpools_config text,
      new_duration     blob,
      PRIMARY KEY ((launchpools_config), event_id)
) WITH CLUSTERING ORDER BY (event_id DESC);

CREATE TABLE IF NOT EXISTS
update_launchpools_config_position_sizes_events
(
      signature              text,
      timestamp              bigint,
      event_id               timeuuid,
      authority              text,
      launchpools_config     text,
      new_min_position_size blob,
      new_max_position_size blob,
      PRIMARY KEY ((launchpools_config), event_id)
) WITH CLUSTERING ORDER BY (event_id DESC);

CREATE TABLE IF NOT EXISTS upd_lp_cfg_protocol_reward_events
(
      signature                                  text,
      timestamp                                  bigint,
      event_id                                   timeuuid,
      authority                                  text,
      launchpools_config                         text,
      new_protocol_reward_share_basis_points smallint,
      PRIMARY KEY ((launchpools_config), event_id)
) WITH CLUSTERING ORDER BY (event_id DESC);

CREATE TABLE IF NOT EXISTS upd_lp_cfg_reward_auth_events
(
      signature              text,
      timestamp              bigint,
      event_id               timeuuid,
      authority              text,
      launchpools_config    text,
      new_reward_authority text,
      PRIMARY KEY ((launchpools_config), event_id)
) WITH CLUSTERING ORDER BY (event_id DESC);

CREATE TABLE IF NOT EXISTS launchpools
(
      signature        text,
      timestamp        bigint,
      authority        text,
      launchpool       text PRIMARY KEY,
      launchpools_config text,
      reward_mint      text,
      reward_vault     text
);

CREATE TABLE IF NOT EXISTS launchpools_status
(
      launchpool text PRIMARY KEY,
      status     tinyint
```

```
);

CREATE TABLE IF NOT EXISTS stake_positions
(
      signature        text,
      timestamp        bigint,
      user             text,
      launchpool       text,
      stake_position text  PRIMARY KEY
);

CREATE TABLE IF NOT EXISTS stake_positions_by_user
(
      signature        text,
      event_id         timeuuid,
      timestamp        bigint,
      user             text,
      launchpool       text,
      stake_position text,
      PRIMARY KEY ((user), event_id)
) WITH CLUSTERING ORDER BY (event_id DESC);

CREATE TABLE IF NOT EXISTS stake_positions_by_launchpool
(
      signature        text,
      event_id         timeuuid,
      timestamp        bigint,
      launchpool       text,
      owner            text,
      stake_position text,
      PRIMARY KEY ((launchpool), event_id)
) WITH CLUSTERING ORDER BY (event_id DESC);

CREATE TABLE IF NOT EXISTS stake_positions_status
(
      stake_position text PRIMARY KEY,
      status           tinyint,
);

CREATE KEYSPACE IF NOT EXISTS liquidity_pool_data
      WITH replication = {
      'class': 'SimpleStrategy',
      'replication_factor': 1
      };

USE liquidity_pool_data;

CREATE TABLE IF NOT EXISTS init_amms_cfg_mgr_events
(
      signature        text PRIMARY KEY,
      timestamp        bigint,
      signer           text,
      authority        text,
      head_authority text
);

CREATE TABLE IF NOT EXISTS upd_amms_cfg_mgr_auth_events
```

```
(
     signature  text PRIMARY KEY,
     timestamp  bigint,
     authority  text,
     new_authority text
);

CREATE TABLE IF NOT EXISTS upd_amms_cfg_mgr_head_auth_events
(
     signature        text PRIMARY KEY,
     timestamp        bigint,
     head_authority   text,
     new_head_authority text
);

CREATE TABLE IF NOT EXISTS init_amms_config_events
(
     signature                          text,
     timestamp                          bigint,
     amms_config                        text PRIMARY KEY,
     authority                          text,
     fee_authority                      text,
     protocol_fee_rate_basis_points  smallint,
     providers_fee_rate_basis_points smallint,
     id                                 bigint
);

CREATE TABLE IF NOT EXISTS upd_amms_cfg_fee_auth_events
(
     signature        text,
     timestamp        bigint,
     event_id         timeuuid,
     authority        text,
     amms_config      text,
     new_fee_authority text,
     PRIMARY KEY ((amms_config), event_id)
) WITH CLUSTERING ORDER BY (event_id DESC);

CREATE TABLE IF NOT EXISTS upd_amms_cfg_protocol_fee_events
(
     signature                          text,
     timestamp                          bigint,
     event_id                           timeuuid,
     authority                          text,
     amms_config                        text,
     new_protocol_fee_rate_basis_points smallint,
     PRIMARY KEY ((amms_config), event_id)
) WITH CLUSTERING ORDER BY (event_id DESC);

CREATE TABLE IF NOT EXISTS upd_amms_cfg_prov_fee_events
(
     signature                          text,
     timestamp                          bigint,
     event_id                           timeuuid,
     authority                          text,
     amms_config                        text,
     new_providers_fee_rate_basis_points smallint,
```

```
      PRIMARY KEY ((amms_config), event_id)
) WITH CLUSTERING ORDER BY (event_id DESC);

CREATE TABLE IF NOT EXISTS cp_amms_keys
(
      cp_amm     text PRIMARY KEY,
      amms_config text,
      base_mint   text,
      quote_mint  text,
      lp_mint    text
);

CREATE TABLE IF NOT EXISTS uninitialized_cp_amms
(
      signature   text,
      timestamp   bigint,
      event_id   timeuuid,
      creator    text,
      cp_amm     text,
      amms_config text,
      base_mint   text,
      quote_mint  text,
      lp_mint    text,
      PRIMARY KEY ((creator), event_id, cp_amm)
) WITH CLUSTERING ORDER BY (event_id DESC);

CREATE TABLE IF NOT EXISTS launched_cp_amms
(
      signature   text,
      timestamp   bigint,
      event_id   timeuuid,
      creator    text,
      cp_amm     text PRIMARY KEY,
      amms_config text,
      base_mint   text,
      quote_mint  text,
      lp_mint    text
);

CREATE TABLE IF NOT EXISTS launched_cp_amms_by_base_or_quote
(
      signature        text,
      timestamp        bigint,
      creator          text,
      cp_amm           text,
      amms_config      text,
      base_or_quote_mint text,
      remaining_mint   text,
      lp_mint          text,
      PRIMARY KEY ((base_or_quote_mint), cp_amm)
);

CREATE TABLE IF NOT EXISTS launched_cp_amms_by_base_and_quote
(
      signature   text,
      timestamp   bigint,
      creator    text,
```

```
      cp_amm     text,
      amms_config text,
      base_mint   text,
      quote_mint  text,
      lp_mint    text,
      PRIMARY KEY ((base_mint, quote_mint), cp_amm)
);

CREATE TABLE IF NOT EXISTS cp_amms_liquidity
(
      cp_amm     text PRIMARY KEY,
      liquidity blob
);

CREATE TABLE IF NOT EXISTS trades_by_cp_amm
(
      signature        text,
      timestamp        bigint,
      event_id         timeuuid,
      swapper          text,
      cp_amm           text,
      swapped_amount  blob,
      received_amount blob,
      is_in_out        boolean,
      PRIMARY KEY ((cp_amm), event_id)
) WITH CLUSTERING ORDER BY (event_id DESC);

CREATE TABLE IF NOT EXISTS trades_by_user
(
      signature        text,
      timestamp        bigint,
      event_id         timeuuid,
      swapper          text,
      cp_amm           text,
      swapped_amount  blob,
      received_amount blob,
      is_in_out        boolean,
      PRIMARY KEY ((swapper), event_id)
) WITH CLUSTERING ORDER BY (event_id DESC);

CREATE TABLE IF NOT EXISTS collect_fees_from_cp_amm_event
(
      signature                   text,
      timestamp                   bigint,
      event_id                    timeuuid,
      cp_amm                      text,
      signer                      text,
      fee_authority               text,
      fee_authority_base_account  text,
      fee_authority_quote_account   text,
      withdrawn_protocol_base_fees  blob,
      withdrawn_protocol_quote_fees blob,
      PRIMARY KEY ((cp_amm), event_id)
) WITH CLUSTERING ORDER BY (event_id DESC);
```

В.7 Логіка формування транзакцій

```
use crate::launchpool::context::LaunchpoolContext;
use crate::launchpool::core::instructions::{
      close_stake_position_ix, collect_protocol_reward_ix,
increase_stake_position_ix,
      initialize_launchpool_ix, initialize_launchpools_config_ix,
      initialize_launchpools_configs_manager_ix, launch_launchpool_ix,
open_stake_position_ix,
      update_launchpools_config_duration_ix,
update_launchpools_config_position_sizes_ix,
      update_launchpools_config_protocol_reward_share_ix,
      update_launchpools_config_reward_authority_ix,
update_launchpools_configs_manager_authority_ix,
      update_launchpools_configs_manager_head_authority_ix,
};
use crate::utils::clients::{ProgramContext, SolanaRpcClient};
use crate::utils::types::{build_unsigned_transaction,
UnsignedTransaction};
use anyhow::Result as AnyResult;
use solana_sdk::pubkey::Pubkey;

pub async fn initialize_launchpools_configs_manager_tx(
      context: &LaunchpoolContext,
      signer: Pubkey,
      authority: Pubkey,
      head_authority: Pubkey,
) -> AnyResult<(UnsignedTransaction, Pubkey)> {
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let (ix, launchpools_configs_manager_pubkey) =
initialize_launchpools_configs_manager_ix(signer, authority,
head_authority);
      Ok((build_unsigned_transaction(&signer, [ix], blockhash, []),
launchpools_configs_manager_pubkey))
}
pub async fn update_launchpools_configs_manager_authority_tx(
      context: &LaunchpoolContext,
      authority: Pubkey,
      new_authority: Pubkey,
) -> AnyResult<UnsignedTransaction> {
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let ix =
update_launchpools_configs_manager_authority_ix(authority,
new_authority);
      Ok(build_unsigned_transaction(&authority, [ix], blockhash, []))
}
pub async fn update_launchpools_configs_manager_head_authority_tx(
      context: &LaunchpoolContext,
      head_authority: Pubkey,
      new_head_authority: Pubkey,
) -> AnyResult<UnsignedTransaction> {
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
```

```
      let ix =

update_launchpools_configs_manager_head_authority_ix(head_authority,
new_head_authority);
      Ok(build_unsigned_transaction(
      &head_authority,
      [ix],
      blockhash,
      [],
      ))
}
pub async fn initialize_launchpools_config_tx(
      context: &LaunchpoolContext,
      authority: Pubkey,
      launchpools_configs_manager: Pubkey,
      reward_authority: Pubkey,
      stakable_mint: Pubkey,
      min_position_size: u64,
      max_position_size: u64,
      protocol_reward_share_basis_points: u16,
      duration: u64,
) -> AnyResult<(UnsignedTransaction, Pubkey)> {
      let launchpools_configs_manager_account =
context.solana_rpc_client().fetch_launchpools_configs_manager(&launchp
ools_configs_manager).await?;
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let (ix, launchpools_config_pubkey) =
initialize_launchpools_config_ix(
      authority,
      launchpools_configs_manager_account.configs_count,
      reward_authority,
      stakable_mint,
      min_position_size,
      max_position_size,
      protocol_reward_share_basis_points,
      duration,
      );
      Ok((build_unsigned_transaction(&authority, [ix], blockhash, []),
launchpools_config_pubkey))
}

pub async fn update_launchpools_config_reward_authority_tx(
      context: &LaunchpoolContext,
      authority: Pubkey,
      launchpools_config: Pubkey,
      new_reward_authority: Pubkey,
) -> AnyResult<UnsignedTransaction> {
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let ix = update_launchpools_config_reward_authority_ix(
      authority,
      launchpools_config,
      new_reward_authority,
      );
      Ok(build_unsigned_transaction(&authority, [ix], blockhash, []))
}
```

```
pub async fn update_launchpools_config_protocol_reward_share_tx(
      context: &LaunchpoolContext,
      authority: Pubkey,
      launchpools_config: Pubkey,
      new_protocol_reward_share_basis_points: u16,
) -> AnyResult<UnsignedTransaction> {
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let ix = update_launchpools_config_protocol_reward_share_ix(
      authority,
      launchpools_config,
      new_protocol_reward_share_basis_points,
      );
      Ok(build_unsigned_transaction(&authority, [ix], blockhash, []))
}

pub async fn update_launchpools_config_position_sizes_tx(
      context: &LaunchpoolContext,
      authority: Pubkey,
      launchpools_config: Pubkey,
      new_min_position_size: u64,
      new_max_position_size: u64,
) -> AnyResult<UnsignedTransaction> {
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let ix = update_launchpools_config_position_sizes_ix(
      authority,
      launchpools_config,
      new_min_position_size,
      new_max_position_size,
      );
      Ok(build_unsigned_transaction(&authority, [ix], blockhash, []))
}

pub async fn update_launchpools_config_duration_tx(
      context: &LaunchpoolContext,
      authority: Pubkey,
      launchpools_config: Pubkey,
      new_duration: u64,
) -> AnyResult<UnsignedTransaction> {
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let ix = update_launchpools_config_duration_ix(authority,
launchpools_config, new_duration);
      Ok(build_unsigned_transaction(&authority, [ix], blockhash, []))
}

pub async fn initialize_launchpool_tx(
      context: &LaunchpoolContext,
      authority: Pubkey,
      launchpools_config: Pubkey,
      reward_mint: Pubkey,
      initial_reward_amount: u64,
) -> AnyResult<(UnsignedTransaction, Pubkey)> {
      let reward_mint_account =
context.get_token_mint(&reward_mint).await?;
```

```
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let (ix, launchpool_pubkey) = initialize_launchpool_ix(
      authority,
      launchpools_config,
      reward_mint,
      *reward_mint_account.program(),
      initial_reward_amount,
      );
      Ok((build_unsigned_transaction(&authority, [ix], blockhash, []),
launchpool_pubkey))
}

pub async fn launch_launchpool_tx(
      context: &LaunchpoolContext,
      authority: Pubkey,
      launchpool: Pubkey,
      start_timestamp: u64,
) -> AnyResult<UnsignedTransaction> {
      let launchpool_keys =
context.get_launchpool_keys(&launchpool).await?;
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let ix = launch_launchpool_ix(
      authority,
      launchpool_keys.launchpools_config,
      launchpool,
      launchpool_keys.reward_mint,
      start_timestamp,
      );
      Ok(build_unsigned_transaction(&authority, [ix], blockhash, []))
}

pub async fn open_stake_position_tx(
      context: &LaunchpoolContext,
      signer: Pubkey,
      signer_stakable_account: Option<Pubkey>,
      launchpool: Pubkey,
      stake_amount: u64,
) -> AnyResult<(UnsignedTransaction, Pubkey)> {
      let launchpool_keys =
context.get_launchpool_keys(&launchpool).await?;
      let launchpools_config_keys = context

.get_launchpools_config_keys(&launchpool_keys.launchpools_config)
      .await?;
      let stakable_token_account = context
      .get_token_mint(&launchpools_config_keys.stakable_mint)
      .await?;
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let (ix, stake_position_pubkey) = open_stake_position_ix(
      signer,
      signer_stakable_account,
      launchpool_keys.launchpools_config,
      launchpool,
      launchpools_config_keys.stakable_mint,
```

```
      *stakable_token_account.program(),
      stake_amount,
      );
      Ok((build_unsigned_transaction(&signer, [ix], blockhash, []),
stake_position_pubkey))
}

pub async fn increase_stake_position_tx(
      context: &LaunchpoolContext,
      signer: Pubkey,
      signer_stakable_account: Option<Pubkey>,
      stake_position: Pubkey,
      stake_increase_amount: u64,
) -> AnyResult<UnsignedTransaction> {
      let stake_position_keys =
context.get_stake_position_keys(&stake_position).await?;
      let launchpool_keys = context
      .get_launchpool_keys(&stake_position_keys.launchpool)
      .await?;
      let launchpools_config_keys = context

.get_launchpools_config_keys(&launchpool_keys.launchpools_config)
      .await?;
      let stakable_token_account = context
      .get_token_mint(&launchpools_config_keys.stakable_mint)
      .await?;
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let ix = increase_stake_position_ix(
      signer,
      signer_stakable_account,
      launchpool_keys.launchpools_config,
      stake_position_keys.launchpool,
      launchpools_config_keys.stakable_mint,
      stake_position,
      *stakable_token_account.program(),
      stake_increase_amount,
      );
      Ok(build_unsigned_transaction(&signer, [ix], blockhash, []))
}

pub async fn close_stake_position_tx(
      context: &LaunchpoolContext,
      signer: Pubkey,
      signer_stakable_account: Option<Pubkey>,
      stake_position: Pubkey,
) -> AnyResult<UnsignedTransaction> {
      let stake_position_keys =
context.get_stake_position_keys(&stake_position).await?;
      let launchpool_keys = context
      .get_launchpool_keys(&stake_position_keys.launchpool)
      .await?;
      let launchpools_config_keys = context

.get_launchpools_config_keys(&launchpool_keys.launchpools_config)
      .await?;
```

```
      let (stakable_token_account, reward_token_account) =
tokio::try_join!(
      context.get_token_mint(&launchpools_config_keys.stakable_mint),
      context.get_token_mint(&launchpool_keys.reward_mint),
      )?;
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let ix = close_stake_position_ix(
      signer,
      signer_stakable_account,
      launchpool_keys.launchpools_config,
      stake_position_keys.launchpool,
      launchpools_config_keys.stakable_mint,
      launchpool_keys.reward_mint,
      stake_position,
      *stakable_token_account.program(),
      *reward_token_account.program(),
      );
      Ok(build_unsigned_transaction(&signer, [ix], blockhash, []))
}

pub async fn collect_protocol_reward_tx(
      context: &LaunchpoolContext,
      signer: Pubkey,
      launchpool: Pubkey,
) -> AnyResult<UnsignedTransaction> {
      let launchpool_keys =
context.get_launchpool_keys(&launchpool).await?;
      let solana_rpc_client = context.solana_rpc_client();
      let (launchpools_config_account, reward_mint_account) =
tokio::try_join!(

solana_rpc_client.fetch_launchpools_config(&launchpool_keys.launchpool
s_config),
      context.get_token_mint(&launchpool_keys.reward_mint)
      )?;
      let blockhash = solana_rpc_client.get_blockhash().await?;
      let ix = collect_protocol_reward_ix(
      signer,
      launchpools_config_account.reward_authority,
      launchpool_keys.launchpools_config,
      launchpool_keys.reward_mint,
      launchpool,
      *reward_mint_account.program(),
      );
      Ok(build_unsigned_transaction(&signer, [ix], blockhash, []))
}

use crate::liquidity_pool::context::LiquidityPoolContext;
use crate::liquidity_pool::core::instructions::*;
use crate::utils::clients::{ProgramContext, SolanaRpcClient};
use crate::utils::instructions::set_compute_budget_ix;
use crate::utils::types::{
      build_unsigned_transaction, UnsignedTransaction,
UnsignedTransactionBuilder,
};
use anyhow::Result as AnyResult;
```

```
use solana_sdk::pubkey::Pubkey;
use solana_sdk::signature::Keypair;

pub async fn initialize_amms_configs_manager_tx(
      context: &LiquidityPoolContext,
      signer: Pubkey,
      authority: Pubkey,
      head_authority: Pubkey,
) -> AnyResult<(UnsignedTransaction, Pubkey)> {
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let (ix, amms_configs_manager_pubkey) =
      initialize_amms_configs_manager_ix(signer, authority,
head_authority);
      Ok((
      build_unsigned_transaction(&signer, [ix], blockhash, []),
      amms_configs_manager_pubkey,
      ))
}
pub async fn update_amms_configs_manager_authority_tx(
      context: &LiquidityPoolContext,
      authority: Pubkey,
      new_authority: Pubkey,
) -> AnyResult<UnsignedTransaction> {
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let ix = update_amms_configs_manager_authority_ix(authority,
new_authority);
      Ok(build_unsigned_transaction(&authority, [ix], blockhash, []))
}
pub async fn update_amms_configs_manager_head_authority_tx(
      context: &LiquidityPoolContext,
      head_authority: Pubkey,
      new_head_authority: Pubkey,
) -> AnyResult<UnsignedTransaction> {
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let ix =
update_amms_configs_manager_head_authority_ix(head_authority,
new_head_authority);
      Ok(build_unsigned_transaction(
      &head_authority,
      [ix],
      blockhash,
      [],
      ))
}
pub async fn initialize_amms_config_tx(
      context: &LiquidityPoolContext,
      authority: Pubkey,
      amms_configs_manager: Pubkey,
      fee_authority: Pubkey,
      protocol_fee_rate_basis_points: u16,
      providers_fee_rate_basis_points: u16,
) -> AnyResult<(UnsignedTransaction, Pubkey)> {
      let amms_config_manager_account = context
      .solana_rpc_client()
```

```
      .fetch_amms_configs_manager(&amms_configs_manager)
      .await?;
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let (ix, amms_config_pubkey) = initialize_amms_config_ix(
      authority,
      amms_config_manager_account.configs_count,
      fee_authority,
      protocol_fee_rate_basis_points,
      providers_fee_rate_basis_points,
      );
      Ok((
      build_unsigned_transaction(&authority, [ix], blockhash, []),
      amms_config_pubkey,
      ))
}
pub async fn update_amms_config_fee_authority_tx(
      context: &LiquidityPoolContext,
      authority: Pubkey,
      amms_config: Pubkey,
      new_fee_authority: Pubkey,
) -> AnyResult<UnsignedTransaction> {
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let ix = update_amms_config_fee_authority_ix(authority,
amms_config, new_fee_authority);
      Ok(build_unsigned_transaction(&authority, [ix], blockhash, []))
}

pub async fn update_amms_config_protocol_fee_rate_tx(
      context: &LiquidityPoolContext,
      authority: Pubkey,
      amms_config: Pubkey,
      new_protocol_fee_rate_basis_points: u16,
) -> AnyResult<UnsignedTransaction> {
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let ix = update_amms_config_protocol_fee_rate_ix(
      authority,
      amms_config,
      new_protocol_fee_rate_basis_points,
      );
      Ok(UnsignedTransactionBuilder::new()
      .recent_blockhash(blockhash)
      .instruction(ix)
      .payer(&authority)
      .build())
}

pub async fn update_amms_config_providers_fee_rate_tx(
      context: &LiquidityPoolContext,
      authority: Pubkey,
      amms_config: Pubkey,
      new_providers_fee_rate_basis_points: u16,
) -> AnyResult<UnsignedTransaction> {
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
```

```
      let ix = update_amms_config_providers_fee_rate_ix(
      authority,
      amms_config,
      new_providers_fee_rate_basis_points,
      );
      Ok(UnsignedTransactionBuilder::new()
      .recent_blockhash(blockhash)
      .instruction(ix)
      .payer(&authority)
      .build())
}
pub async fn create_cp_amm_tx(
      context: &LiquidityPoolContext,
      signer: Pubkey,
      base_mint: Pubkey,
      quote_mint: Pubkey,
      amms_config: Pubkey,
      base_liquidity: u64,
      quote_liquidity: u64
) -> AnyResult<(UnsignedTransaction, Pubkey)> {
      let lp_mint_keypair = Keypair::new();
      let solana_rpc_client = context.solana_rpc_client();
      let (amms_config_account, base_mint_account, quote_mint_account)
= tokio::try_join!(
      solana_rpc_client.fetch_amms_config(&amms_config),
      context.get_token_mint(&base_mint),
      context.get_token_mint(&quote_mint),
      )?;
      let blockhash = solana_rpc_client.get_blockhash().await?;
      let mut ixs = vec![set_compute_budget_ix(300_000)];
      let (create_ixs, cp_amm_pubkey) = create_cp_amm_ixs(
      signer,
      &lp_mint_keypair,
      amms_config_account.fee_authority,
      base_mint,
      quote_mint,
      amms_config,
      *base_mint_account.program(),
      *quote_mint_account.program(),
      base_liquidity,
      quote_liquidity
      );
      ixs.extend(create_ixs);
      Ok((
      build_unsigned_transaction(&signer, ixs, blockhash,
[&lp_mint_keypair]),
      cp_amm_pubkey,
      ))
}
pub async fn initialize_cp_amm_tx(
      context: &LiquidityPoolContext,
      signer: Pubkey,
      base_mint: Pubkey,
      quote_mint: Pubkey,
      amms_config: Pubkey,
) -> AnyResult<(UnsignedTransaction, Pubkey)> {
      let lp_mint_keypair = Keypair::new();
```

```
      let solana_rpc_client = context.solana_rpc_client();
      let (amms_config_account, base_mint_account, quote_mint_account)
= tokio::try_join!(
      solana_rpc_client.fetch_amms_config(&amms_config),
      context.get_token_mint(&base_mint),
      context.get_token_mint(&quote_mint),
      )?;
      let blockhash = solana_rpc_client.get_blockhash().await?;
      let (ix, cp_amm_pubkey) = initialize_cp_amm_ix(
      signer,
      &lp_mint_keypair,
      amms_config_account.fee_authority,
      base_mint,
      quote_mint,
      amms_config,
      *base_mint_account.program(),
      *quote_mint_account.program(),
      );
      Ok((
      build_unsigned_transaction(&signer, [ix], blockhash,
[&lp_mint_keypair]),
      cp_amm_pubkey,
      ))
}
pub async fn launch_cp_amm_tx(
      context: &LiquidityPoolContext,
      creator: Pubkey,
      cp_amm: Pubkey,
      base_liquidity: u64,
      quote_liquidity: u64,
) -> AnyResult<(UnsignedTransaction, Pubkey)> {
      let cp_amm_keys = context.get_cp_amm_keys(&cp_amm).await?;
      let (lp_mint_account, base_mint_account, quote_mint_account) =
tokio::try_join!(
      context.get_token_mint(&cp_amm_keys.lp_mint),
      context.get_token_mint(&cp_amm_keys.base_mint),
      context.get_token_mint(&cp_amm_keys.quote_mint),
      )?;
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let mut ixs = vec![set_compute_budget_ix(250_000)];
      let (launch_ixs, cp_amm_pubkey) = launch_cp_amm_ixs(
      creator,
      cp_amm_keys.amms_config,
      cp_amm,
      cp_amm_keys.base_mint,
      cp_amm_keys.quote_mint,
      cp_amm_keys.lp_mint,
      *base_mint_account.program(),
      *quote_mint_account.program(),
      *lp_mint_account.program(),
      base_liquidity,
      quote_liquidity,
      );
      ixs.extend(launch_ixs);
      Ok((build_unsigned_transaction(
      &creator,
```

```
      ixs,
      blockhash,
      [],
      ), cp_amm_pubkey))
}
pub async fn provide_to_cp_amm_tx(
      context: &LiquidityPoolContext,
      signer: Pubkey,
      cp_amm: Pubkey,
      base_liquidity: u64,
      quote_liquidity: u64,
) -> AnyResult<UnsignedTransaction> {
      let cp_amm_keys = context.get_cp_amm_keys(&cp_amm).await?;
      let (lp_mint_account, base_mint_account, quote_mint_account) =
tokio::try_join!(
      context.get_token_mint(&cp_amm_keys.lp_mint),
      context.get_token_mint(&cp_amm_keys.base_mint),
      context.get_token_mint(&cp_amm_keys.quote_mint),
      )?;
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
      let ixs = provide_to_cp_amm_ixs(
      signer,
      cp_amm_keys.amms_config,
      cp_amm,
      cp_amm_keys.base_mint,
      cp_amm_keys.quote_mint,
      cp_amm_keys.lp_mint,
      *base_mint_account.program(),
      *quote_mint_account.program(),
      *lp_mint_account.program(),
      base_liquidity,
      quote_liquidity,
      );
      Ok(build_unsigned_transaction(
      &signer,
      ixs,
      blockhash,
      [],
      ))
}

pub async fn withdraw_from_cp_amm_tx(
      context: &LiquidityPoolContext,
      signer: Pubkey,
      cp_amm: Pubkey,
      lp_tokens: u64,
) -> AnyResult<UnsignedTransaction> {
      let cp_amm_keys = context.get_cp_amm_keys(&cp_amm).await?;
      let (lp_mint_account, base_mint_account, quote_mint_account) =
tokio::try_join!(
      context.get_token_mint(&cp_amm_keys.lp_mint),
      context.get_token_mint(&cp_amm_keys.base_mint),
      context.get_token_mint(&cp_amm_keys.quote_mint),
      )?;
      let blockhash =
context.solana_rpc_client().get_blockhash().await?;
```

```
        let ixs = withdraw_from_cp_amm_ixs(
        signer,
        cp_amm_keys.amms_config,
        cp_amm,
        cp_amm_keys.base_mint,
        cp_amm_keys.quote_mint,
        cp_amm_keys.lp_mint,
        *base_mint_account.program(),
        *quote_mint_account.program(),
        *lp_mint_account.program(),
        lp_tokens,
        );
        Ok(build_unsigned_transaction(
        &signer,
        ixs,
        blockhash,
        [],
        ))
}

pub async fn swap_in_cp_amm_tx(
        context: &LiquidityPoolContext,
        signer: Pubkey,
        cp_amm: Pubkey,
        swap_amount: u64,
        estimated_result: u64,
        allowed_slippage: u64,
        is_in_out: bool,
) -> AnyResult<UnsignedTransaction> {
        let cp_amm_keys = context.get_cp_amm_keys(&cp_amm).await?;
        let (base_mint_account, quote_mint_account) = tokio::try_join!(
        context.get_token_mint(&cp_amm_keys.base_mint),
        context.get_token_mint(&cp_amm_keys.quote_mint),
        )?;
        let blockhash =
context.solana_rpc_client().get_blockhash().await?;
        let ixs = swap_in_cp_amm_ixs(
        signer,
        cp_amm_keys.amms_config,
        cp_amm,
        cp_amm_keys.base_mint,
        cp_amm_keys.quote_mint,
        *base_mint_account.program(),
        *quote_mint_account.program(),
        swap_amount,
        estimated_result,
        allowed_slippage,
        is_in_out,
        );
        Ok(build_unsigned_transaction(
        &signer,
        ixs,
        blockhash,
        [],
        ))
}
```

```
pub async fn collect_fees_from_cp_amm_tx(
      context: &LiquidityPoolContext,
      signer: Pubkey,
      cp_amm: Pubkey,
) -> AnyResult<UnsignedTransaction> {
      let cp_amm_keys = context.get_cp_amm_keys(&cp_amm).await?;
      let solana_rpc_client = context.solana_rpc_client();
      let (amms_config_account, base_mint_account, quote_mint_account)
= tokio::try_join!(
      solana_rpc_client.fetch_amms_config(&cp_amm_keys.amms_config),
      context.get_token_mint(&cp_amm_keys.base_mint),
      context.get_token_mint(&cp_amm_keys.quote_mint),
      )?;
      let blockhash = solana_rpc_client.get_blockhash().await?;
      let ix = collect_fees_from_cp_amm_ix(
      signer,
      amms_config_account.fee_authority,
      cp_amm_keys.amms_config,
      cp_amm,
      cp_amm_keys.base_mint,
      cp_amm_keys.quote_mint,
      *base_mint_account.program(),
      *quote_mint_account.program(),
      );
      Ok(build_unsigned_transaction(&signer, [ix], blockhash, []))
}
```

В.8 Інтеграція веб-клієнта з Solana

```
"use client";

import React, {createContext, useContext, useMemo} from "react";
import { WalletContextState} from "@solana/wallet-adapter-react";
import {Connection, PublicKey, TransactionSignature,
VersionedTransaction} from "@solana/web3.js";
import {Transaction} from "@solana/kit";
import {SendTransactionOptions} from "@solana/wallet-adapter-base";

type SolanaWalletContext = {
      walletData: WalletData | undefined,
      connection: Connection
};
type WalletData = {
      publicKey: PublicKey;
      signTransaction: <T extends VersionedTransaction |
Transaction>(tx: T) => Promise<T>;
      sendTransaction: (
      tx: VersionedTransaction | Transaction,
      connection: Connection,
      options?: SendTransactionOptions
      ) => Promise<TransactionSignature>;
};
const SolanaContext = createContext<SolanaWalletContext |
undefined>(undefined);

export const SolanaContextWrapper = ({wallet, connection, children,}:
{
      wallet: WalletContextState;
      connection: Connection;
      children: React.ReactNode;
}) => {
      const getWalletData = (wallet: WalletContextState): WalletData |
undefined => {
      let publicKey = wallet.publicKey;
      let signTransaction = wallet.signTransaction;
      let sendTransaction = wallet.sendTransaction;
      if (!publicKey || !signTransaction || !wallet.connected) return
undefined;
      return {
            publicKey,
            signTransaction,
            sendTransaction
      } as WalletData
      };


      const walletData = useMemo(() => getWalletData(wallet), [wallet,
connection]);

      return (
      <SolanaContext.Provider value={{walletData, connection}}>
            {children}
      </SolanaContext.Provider>
```

```
      );
};

export const usePublicKey = () => {
      const context = useContext(SolanaContext);
      if (!context) {
      throw new Error("SolanaContext is not initialized.");
      }
      const wallet = context.walletData;
      return wallet?.publicKey;
};
export const useSignAndSendTransaction = () => {
      const context = useContext(SolanaContext);
      if (!context) {
      throw new Error("SolanaContext is not initialized.");
      }
      const wallet = context.walletData;
      if (!wallet) {
      throw new Error("Wallet is not connected.");
      }

      return async <T extends Transaction>(
      transaction: T,
      options?: SendTransactionOptions
      ): Promise<TransactionSignature> => {
      const signed = await wallet.signTransaction(transaction);
      return await wallet.sendTransaction(signed, context.connection,
options);
      };
};
export const useSolanaWalletContext = () => {
      const context = useContext(SolanaContext);
      if (!context) {
      throw new Error("SolanaWalletContext is not initialized.");
      }
      return context;
};

export const useIsWalletConnected = (): boolean => {
      const context = useContext(SolanaContext);
      return !!context?.walletData;
};

'use client'

import dynamic from 'next/dynamic'
import {
  ConnectionProvider,
  WalletProvider,
} from '@solana/wallet-adapter-react'
import { WalletModalProvider } from '@solana/wallet-adapter-react-ui'
import React, { ReactNode, useMemo} from 'react'
import {PhantomWalletAdapter} from "@solana/wallet-adapter-phantom";
import {WalletButtonProvider} from
"@/components/solana/wallet-button-provider";
import {SolanaWalletProvider} from
"@/components/solana/solana-wallet-provider";
```

```
import {Connection} from "@solana/web3.js";

require('@solana/wallet-adapter-react-ui/styles.css')

export const WalletButton = dynamic(async () => (await
import('@solana/wallet-adapter-react-ui')).WalletMultiButton, {
  ssr: false,
})
export const SolanaWalletAdapterProvider = ({endpoint, children }:
{endpoint: string, children: ReactNode}) => {
  const wallets = useMemo(
      () => [
      new PhantomWalletAdapter(),
      ],
      // eslint-disable-next-line react-hooks/exhaustive-deps
      [endpoint]
  );
  const connection = new Connection(endpoint);
  return (
      <ConnectionProvider endpoint={endpoint}>
      <WalletProvider wallets={wallets} autoConnect>
            <WalletModalProvider>
            <WalletButtonProvider>
                  <SolanaWalletProvider connection={connection}>
                        {children}
                  </SolanaWalletProvider>
            </WalletButtonProvider>
            </WalletModalProvider>
      </WalletProvider>
      </ConnectionProvider>
  );
};

"use client"

import React, {ReactNode} from "react";
import {useWallet} from "@solana/wallet-adapter-react";
import '@solana/wallet-adapter-react-ui/styles.css';
import {SolanaContextWrapper} from
"@/components/solana/solana-context-wrapper";
import {Connection} from "@solana/web3.js";

export const SolanaWalletProvider = ({connection, children,}:
{connection: Connection, children: ReactNode; }) => {
      const wallet = useWallet();
      return (
      <SolanaContextWrapper wallet={wallet} connection={connection}>
            {children}
      </SolanaContextWrapper>
      );
};

"use client"

import {createContext, useContext} from "react";

const WalletButtonContext = createContext<(() => void) | null>(null);
```

```
export const WalletButtonProvider: React.FC<{ children:
React.ReactNode }> = ({ children }) => {
      const triggerWalletButtonClick = () => {
      const button = document.querySelector('.wallet-adapter-button');
      if (button) {
            (button as HTMLButtonElement).click();
      } else {
            console.warn("Wallet button not found");
      }
      };

      return (
      <WalletButtonContext.Provider value={triggerWalletButtonClick}>
            {children}
      </WalletButtonContext.Provider>
      );
};
export const useWalletButton = () => {
      const context = useContext(WalletButtonContext);
      if (!context) {
      throw new Error("useWalletButton has to be used inside
WalletButtonProvider");
      }
      return context;
};
```

ДОДАТОК Г

Чек-листи тестування

Таблиця Г.1 – Чек-лист для модульного тестування Q64_128

Checklist			
Тестувальник:	Міленний Іван	Дата:	28.05.2025
Програмний додаток:	Програмна система децентралізованого обміну токенів на платформі Solana (Бібліотека утиліт)		
Перевірка			
1	Створення екземпляру Q64_128		
2	Перевірка серіалізації та десеріалізації Q64_128 через Anchor		
3	Конверсія з u64 у Q64_128		
4	Конверсія з f64 у Q64_128		
5	Конверсія з Q64_128 у u64		
6	Комбінована перевірка: f64 → Q64_128 → u64 → f64		
7	Відділення цілої та дробової частини		
8	Додавання		
9	Віднімання		
10	Множення		
11	Ділення		
12	Обчислення абсолютної різниці		
13	Додавання з перевіркою		
14	Віднімання з перевіркою		
15	Множення з перевіркою		

Продовження Таблиці Г.1

16	Ділення з перевіркою

17	Зведення в квадрат з перевіркою
18	Квадрат 128-бітного цілого в Q64_128
19	Fuzz: конверсія f64 → Q64_128 → f64
20	Fuzz: корінь 128-бітного числа та його зворотнє перетворення
21	Fuzz: корінь з Q64_128 та його зворотнє перетворення
22	Fuzz: додавання з перевіркою
23	Fuzz: віднімання з перевіркою
24	Fuzz: ділення з перевіркою
25	Fuzz: множення з перевіркою

Таблиця Г.2 – Чек-лист для інтеграційного тестування смартконтракту пулів ліквідності

Checklist			
Тестувальник:	Міленний Іван	Дата:	29.05.2025
Програмний додаток:	Програмна система децентралізованого обміну токенів на платформі Solana (Смартконтракт пулів ліквідності)		
Перевірка			
1	Ініціалізація менеджера конфігурацій без авторизації має завершитися помилкою		
2	Ініціалізація менеджера конфігурацій із некоректним головним авторитетом має завершитися помилкою		
3	Ініціалізація менеджера		

Продовження Таблиці Г.2

4	Повторна ініціалізація менеджера конфігурацій має завершитися помилкою
5	Оновлення авторитету менеджера конфігурацій без прав має завершитися помилкою

6	Оновлення авторитету менеджера конфігурацій автором
7	Оновлення авторитету менеджера конфігурацій головним авторитетом
8	Оновлення головного авторитету менеджера конфігурацій без прав має завершитися помилкою
9	Оновлення головного авторитету менеджера конфігурацій
10	Ініціалізація конфігурації без авторизації має завершитися помилкою
11	Ініціалізація конфігурації з перевищеними комісіями має завершитися помилкою
12	Ініціалізація конфігурації зі стороннім менеджером має завершитися помилкою
13	Ініціалізація конфігурації головним авторитетом
14	Ініціалізація конфігурації
15	Повторна ініціалізація конфігурації має завершитися помилкою
16	Оновлення авторитету комісії без авторизації має завершитися помилкою
17	Оновлення авторитету комісії конфігурації зі стороннім менеджером має завершитися помилкою
18	Оновлення авторитету комісії головним авторитетом

Продовження Таблиці Г.2

19	Оновлення авторитету комісії
20	Оновлення ставки протокольної комісії без авторизації має завершитися помилкою
21	Оновлення ставки протокольної комісії зі стороннім менеджером має завершитися помилкою
22	Оновлення ставки протокольної комісії головним авторитетом

23	Оновлення ставки протокольної комісії
24	Оновлення ставки протокольної комісії з перевищенням ліміту має завершитися помилкою
25	Оновлення ставки комісії провайдерів без авторизації має завершитися помилкою
26	Оновлення ставки комісії провайдерів зі стороннім менеджером має завершитися помилкою
27	Оновлення ставки комісії провайдерів головним авторитетом
28	Оновлення ставки комісії провайдерів
29	Оновлення ставки комісії провайдерів з перевищенням ліміту має завершитися помилкою
30	Ініціалізація пулу ліквідності без фінансування має завершитися помилкою
31	Ініціалізація пулу ліквідності з однаковими токенами має завершитися помилкою
32	Ініціалізація пулу ліквідності з некоректним fee authority має завершитися помилкою
33	Ініціалізація пулу ліквідності зі сторонньою конфігурацією має завершитися помилкою

Продовження Таблиці Г.2

34	Ініціалізація пулу ліквідності з некоректним LP mint має завершитися помилкою
35	Ініціалізація пулу ліквідності з некоректним токеновим акаунтом у ролі сховища має завершитися помилкою
36	Ініціалізація пулу ліквідності з токеном, що має freeze authority, має завершитися помилкою
37	Ініціалізація пулу ліквідності з токеном, що має заборонене розширення (Permanent Delegate), має завершитися помилкою
38	Ініціалізація пулу ліквідності з токеном та токеном 2022

39	Повторна ініціалізація пулу ліквідності має завершитися помилкою
40	Ініціалізація пулу ліквідності з двома токенами
41	Ініціалізація пулу ліквідності з токеном та токеном 2022 з дозволеним розширенням TransferFeeConfig
42	Запуск пулу ліквідності з недостатнім балансом базового токена має завершитися помилкою
43	Запуск пулу ліквідності з недостатнім балансом котирувального токена має завершитися помилкою
44	Запуск пулу ліквідності підписантом, який не є творцем пулу ліквідності, має завершитися помилкою
45	Запуск пулу ліквідності зі звичайним токеном та токеном 2022
46	Повторний запуск пулу ліквідності має завершитися помилкою

Продовження Таблиці Г.2

47	Запуск пулу ліквідності з початковою ліквідністю меншою за заблоковану в 4 рази має завершитися помилкою
48	Запуск пулу ліквідності з двома токенами
49	Запуск пулу ліквідності з токенами та токеном 2022 з TransferFeeConfig
50	Поповнення пулу ліквідності з некоректним співвідношенням токенів має завершитися помилкою
51	Поповнення пулу ліквідності з двома токенами
52	Поповнення пулу ліквідності з токеном та токеном 2022 з TransferFeeConfig
53	Обмін в пулу ліквідності з перевищенням прослизання має завершитися помилкою

54	Обмін в пулу ліквідності, який вичерпує базову ліквідність, має завершитися помилкою
55	Обмін у пулу ліквідності з двома токенами (base → quote)
56	Обмін у пулу ліквідності з двома токенами (quote → base)
57	Обмін у пулу ліквідності з токеном 2022 та TransferFeeConfig
58	Виведення ліквідності з пулу ліквідності при недостатньому балансі LP-токенів має завершитися помилкою
59	Виведення ліквідності з пулу ліквідності з токеном та токеном 2022
60	Збір комісії з пулу ліквідності з некоректним fee authority має завершитися помилкою

Продовження Таблиці Г.2

61	Збір комісії з пулу ліквідності з токеном 2022 та TransferFeeConfig
62	Збір комісії з пулу ліквідності, якщо комісія = 0, має завершитися помилкою

Таблиця Г.3 – Чек-лист для інтеграційного тестування смартконтракту лаунчпулів

Checklist			
Тестувальник:	Міленний Іван	Дата:	30.05.2025
Програмний додаток:	Програмна система децентралізованого обміну токенів на платформі Solana (Смартконтракт лаунчпулів)		
Перевірка			
1	Ініціалізація менеджера конфігурацій без авторизації має завершитися помилкою		

2	Ініціалізація менеджера конфігурацій із некоректним головним авторитетом має завершитися помилкою
3	Ініціалізація менеджера
4	Повторна ініціалізація менеджера конфігурацій має завершитися помилкою
5	Оновлення авторитету менеджера конфігурацій без прав має завершитися помилкою
6	Оновлення авторитету менеджера конфігурацій автором
7	Оновлення авторитету менеджера конфігурацій головним авторитетом
8	Оновлення головного авторитету менеджера конфігурацій без прав має завершитися помилкою

Продовження Таблиці Г.3

9	Оновлення головного авторитету менеджера конфігурацій
10	Ініціалізація конфігурації без авторизації має завершитися помилкою
11	Ініціалізація конфігурації з перевищеною винагородою має завершитися помилкою
12	Ініціалізація конфігурації з нульовою тривалістю має завершитися помилкою
13	Ініціалізація конфігурації з нульовим мінімальним розміром позиції має завершитися помилкою
14	Ініціалізація конфігурації з максимальним розміром позиції менше мінімального має завершитися помилкою
15	Ініціалізація конфігурації з шкідливим менеджером конфігурацій, має завершитися помилкою
16	Ініціалізація конфігурації зі стейк-токеном, що має

	freeze authority, має завершитися помилкою
17	Ініціалізація конфігурації зі стейк-токеном, що має заборонене розширення, має завершитися помилкою
18	Ініціалізація конфігурації головним авторитетом
19	Ініціалізація конфігурації авторитетом
20	Повторна ініціалізація конфігурації має завершитися помилкою

Продовження Таблиці Г.3

21	Оновлення авторитету винагороди без авторизації має завершитися помилкою
22	Оновлення авторитету винагороди з шкідливим менеджером конфігурацій має завершитися помилкою
23	Оновлення авторитету винагороди головним авторитетом
24	Оновлення авторитету винагороди авторитетом
25	Оновлення ставки винагороди протоколу провайдерів без авторизації має завершитися помилкою
26	Оновлення ставки винагороди протоколу зі стороннім менеджером має завершитися помилкою
27	Оновлення ставки винагороди протоколу головним авторитетом
28	Оновлення ставки винагороди протоколу винагороди авторитетом
29	Оновлення ставки винагороди протоколу з перевищенням ліміту має завершитися помилкою
30	Оновлення тривалості без авторизації має завершитися помилкою

31	Оновлення тривалості зі стороннім менеджером має завершитися помилкою
32	Оновлення тривалості головним авторитетом
33	Оновлення тривалості авторитетом
34	Оновлення тривалості рівної 0 має завершитися помилкою
35	Оновлення розмірів позиції без авторизації має завершитися помилкою

Продовження Таблиці Г.3

36	Оновлення розмірів позиції зі стороннім менеджером має завершитися помилкою
37	Оновлення розмірів позиції головним авторитетом
38	Оновлення розмірів позиції авторитетом
39	Оновлення розмірів позиції з мінімальним розміром 0 має завершитися помилкою
40	Оновлення розмірів позиції з максимальним розміром менше ніж з мінімальним має завершитися помилкою
41	Ініціалізація лаунчпулу без авторизації, має завершитись помилкою
42	Ініціалізація лаунчпулу з невідповідним токеном та сховищем, має завершитись помилкою
43	Ініціалізація лаунчпулу з невідповідним токеном та сховищем, має завершитись помилкою
44	Ініціалізація лаунчпулу з токеном, що має freeze authority, має завершитись помилкою
45	Ініціалізація лаунчпулу з токеном, що заборонене розширення, має завершитись помилкою
46	Ініціалізація лаунчпулу без нагороди має завершитись помилкою
47	Ініціалізація лаунчпулу головним авторитетом

48	Ініціалізація лаунчпулу авторитетом
49	Повторна ініціалізація лаунчпулу має завершитись помилкою
50	Запуск порожнього лаунчпулу має завершитись помилкою

Продовження Таблиці Г.3

51	Запуск лаунчпулу без авторизації має завершитись помилкою
52	Запуск неініціалізованого лаунчпулу має завершитись помилкою
53	Запуск лаунчпулу зі стартом в минулому має завершитись помилкою
54	Запуск лаунчпулу головним авторитетом
55	Запуск лаунчпулу авторитетом
56	Повторний запуск лаунчпулу має завершитись помилкою
57	Відкриття позиції в незапущеному лаунчпулі має завершитись помилкою
58	Відкриття позиції до початку лаунчпулу має завершитись помилкою
59	Відкриття позиції з розміром більше дозволеного має завершитись помилкою
60	Відкриття позиції з розміром менше дозволеного має завершитись помилкою
61	Відкриття першої позиції
62	Відкриття позиції
63	Збільшення неіснуючої позиції має завершитись помилкою
64	Збільшення позиції без авторизації має завершитись

	помилкою

Продовження Таблиці Г.3

65	Збільшення позиції на суму більше ніж дозволено має завершитись помилкою
66	Збільшення першої позиції
67	Збільшення позиції
68	Збір нагород протоколу в незавершеному лаунчпулі має завершитись помилкою
69	Закриття позиції в незавершеному лаунчпулі має завершитись помилкою
70	Закриття не відкритої позиції має завершитись помилкою
71	Закриття позиції без авторизації має завершитись помилкою
72	Закриття першої позиції
73	Повторне закриття позиції має завершитись помилкою
74	Закриття позиції
75	Збір нагороди без авторизації має завершитись помилкою
76	Збір нагороди протоколу
77	Повторний збір нагороди має завершитись помилкою

ДОДАТОК Д

Стаття з конференції MIT@AIS-2025

Децентралізована Система Торгівлі Токенами з Високоточними Обчисленнями

Іван Міленний[a], Гліб Терещенко[a] та Ірина Кириченко[a]

[a] *Харківський національний університет радіоелектроніки, пр. Науки, 14, Харків, 61166, Україна.*

Анотація

Сучасні децентралізовані біржі на блокчейні Solana, що побудовані на алгоритмах автоматизованих маркет-мейкерів, покладаються на арифметику з фіксованою точністю для забезпечення обчислень у смартконтрактах. У більшості популярних протоколів, таких як Raydium, застосовується формат чисел Q64.64, який у ряді сценаріїв виявляється недостатнім для точного представлення дробових значень, особливо при низьких обсягах ліквідності або великого дисбалансу. Це створює ризик похибок, що, в свою чергу, може призводити до втрат при обмінах. У даній роботі представлено підхід до реалізації AMM-пулу з підтримкою формату Q64.128, який забезпечує значно вищу точність фінансових обчислень. Такий підхід дозволить мінімізувати втрати точності під час критичних операцій.

Ключові слова

децентралізовані фінанси, блокчейн, обмін токенів, високоточні обчислення, смартконтракти, Solana

1. Вступ

Децентралізовані біржі стали ключовим інструментом у сфері фінансових технологій на базі блокчейну. Вони дозволяють користувачам здійснювати обмін цифрових активів без участі централізованих посередників, спираючись на смартконтракти та математичні алгоритми ціноутворення. Найпоширенішим із таких алгоритмів є Constant Product Market Maker [1], впроваджений у протоколі Uniswap і пізніше адаптований у Raydium [2], Orca та інших. Однією з критичних характеристик AMM є точність обчислень. Для забезпечення арифметики з плаваючою точкою в умовах обмеженого середовища смартконтрактів, більшість протоколів використовують формат Q64.64 – число з 64-бітною цілою та 64-бітною дробовою частинами. Хоча цей формат забезпечує достатню точність у типових сценаріях, він має суттєві обмеження при роботі з екстремальними значеннями, наприклад, великим дисбалансом ліквідності або дуже малим обсягом ліквідності. У таких умовах навіть незначні обчислювальні похибки можуть призводити до втрат при обміні токенів, спотворення курсу або несправедливого розподілу комісій. Uniswap згодом перейшов до формату чисел Q64.96, однак цей тип є не надто зручним для використання в середовищі Solana [3]. Його несиметрична структура (64 біти для цілої частини і 96 – для дробової) ускладнює виконання базових арифметичних операцій на рівні смарт-контрактів і призводить до зростання обчислювальних витрат. Як альтернатива, можна розглядати BigInt (uint256), який активно використовується в Ethereum [4]. Проте і цей підхід не є оптимальним для Solana: 256-бітне число є надмірно великим, адже в контексті токенів на Solana розмір цілої частини обмежений до 64 біт, а використання додаткових 192 біт для дробової частини є неефективним і надто витратним. Більш збалансованим рішенням для Solana є використання формату Q64.128, який займає 192 біти – 64 біти для цілої частини і 128 біт для дробової. Такий розподіл не є надлишковим, забезпечує достатню точність для фінансових обчислень і водночас залишається сумісним із архітектурними обмеженнями Solana.

MIT@AIS'2025s: 1st International Scientific and Practical Conference "Modern Information Technologies and Artificial Intelligence Systems", May 19–22, 2025, Kharkiv-Yaremche, Ukraine
EMAIL: ivan.milennyi@nure.ua (Ivan Milennyi); hlib.tereshchenko@nure.ua (Glib Tereshchenko); iryna.kyrychenko@nure.ua (Iryna Kyrychenko)
ORCID: 0009-0000-3692-8896 (Ivan Milennyi); 0000-0001-8731-2135 (Glib Tereshchenko); 0000-0002-7686-6439 (Iryna Kyrychenko)

Метою цієї роботи є дослідження переваг використання формату Q64.128 у пулах ліквідності на блокчейні Solana.

2. Опис задачі

Алгоритм Constant Product Market Maker, який лежить в основі роботи більшості децентралізованих пулів ліквідності, реалізований за допомогою наведених рівнянь

$$x * y = k, \quad x \div y = P, \tag{1}$$

де: x і y – резерви активів у пулі, k – сталий продукт, P а це співвідношення між резервами.

Отримані параметри зберігаються у вигляді квадратних коренів з самих себе для забезпечення більшої точності при подальших розрахунках. Прикладом таких розрахунків є зворотнє обчислення кількості одного активу, якщо баланс іншого був змінений

$$x_{new} = \sqrt{k} \div \sqrt{P_{new}}, \; y_{new} = \sqrt{k} * \sqrt{P_{new}}, \tag{2}$$

де: x_{new} та y_{new} є оновленими резервами активів, а P_{new} це співвідношення між резервами після зміни одного з балансів.

Описані формули забезпечують цілісність пулу та коректну роботу алгоритму Constant Product Market Maker. Оскільки система виконує обчислення з коренями дробових чисел, навіть незначна похибка може спричинити спотворення результатів обміну або невідповідність нових резервів. Тому впровадження числового формату, здатного виконувати такі обчислення без втрати точності в усьому допустимому діапазоні 64-бітних цілих чисел, є надзвичайно важливим.

3. Особливості чисельного представлення

Q64.64 дозволяє виконувати розрахунки з достатньою точністю у більшості випадків у екстремальних сценаріях, зокрема, при роботі з надмалими або надвеликими значеннями (наприклад, у співвідношеннях резервів 1 до $2^{64} – 1$), навіть одна додаткова бітова похибка може призвести до повної втрати значення.

З метою усунення цих обмежень у реалізації було використано розширений формат Q64.128, у якому дробова частина представлена 128 бітами. Це дозволяє підтримувати стабільність при множенні та діленні без втрати значущих розрядів та гарантувати зворотність операцій.

Для реалізації обчислень з такою точністю використовується структура U192, що складається з трьох 64-бітних чисел, та бітова арифметика [5]. Усі основні операції – множення, ділення, корінь реалізовуються з урахуванням переповнення, нормалізації та перевірки коректності. Застосування формату Q64.128 забезпечує надійну математичну основу для AMM-пулів.

4. Результати

Використання розширеного формату Q64.128 у реалізації AMM-пулу дозволяє досягти суттєвого підвищення точності обчислень без значних додаткових витрат.

Приклад рівняння

$$x_{init} / y * y = x_{res}, \tag{3}$$

де: x і y – цілі числа в діапазоні від 1 до $2^{64} - 1$, x_{init} дорівнює x_{res}.

В Таблиці 1 наведені результати з граничними значеннями x та y.

Як видно з Таблиці 1, у випадках з граничними або великими значеннями змінних, формат Q64.64 демонструє суттєві похибки або втрату точності, у той час як Q64.128 дає правильне значення, зберігаючи оберненість обчислень.

При проведенні порівняльного аналізу було виявлено низку практичних переваг:

- Підвищена стабільність при граничних значеннях
- Гарантована оберненість операцій
- Зменшення ризику експлойтів

Таблиця 1
Приклади обчислень розрахунків з Q64.64 та Q64.128

x_{init}	y	x_{res} для Q64.64	x_{res} для Q64.128
1	2^{60}	0	1
5	2^{62}	4	5
1435661	2^{40}	1435660	1435661
$2^{63}-1$	$2^{62}-1$	2^{63}	$2^{63}-1$

Проте варто зазначити, що формат Q64.128 має і певний недолік порівняно з Q64.64 – він вимагає на 50% більше пам'яті та супроводжується вищими накладними витратами на обчислення. Хоча в більшості практичних сценаріїв роботи пулів ліквідності це не має істотного впливу на продуктивність, невелике зростання обчислювальної складності може відображатися на розмірі комісії за обмін.

Таким чином, формат Q64.128 має значні переваги порівняно з Q64.64. Це дозволяє створювати надійні AMM-протоколи, здатні працювати у сценаріях з високими вимогами до точності обчислень.

5. Реалізація

Для реалізації децентралізованої системи торгівлі токенами з високоточними обчисленнями використовується сучасний інструментарій, орієнтований на високу швидкодію та зручність розробки. Платформою для побудови та розгортання смартконтракту [7] є блокчейн Solana, який забезпечує високу пропускну здатність та дешеві комісії. Логіка смартконтракту реалізована мовою Rust [6] з використанням фреймворку Anchor, що надає високорівневу абстракцію для роботи з акаунтами, інструкціями та перевірками в межах програм на Solana. Клієнтська частина системи побудована з використанням Next.js, що дозволяє створити вебінтерфейс. Для взаємодії з блокчейном використовується бібліотека @solana/web3.js, яка дозволяє підключення до гаманців, формування і підпис транзакцій, а також безпосередній виклик інструкцій Anchor-програм. Діаграма компонентів для описаної системи наведена на Рисунку 1.

Обраний технологічний стек забезпечує не лише гнучкість та масштабованість, а й дозволяє зосередитися на безпечності логіки та обчислень, що особливо важливо для реалізації AMM-протоколів нового покоління.

6. Висновок

У роботі розглянуто проблему обмеженої точності обчислень у децентралізованих автоматизованих маркет-мейкерах, що реалізовані на блокчейні Solana. Було показано, що формат Q64.64, який широко використовується у сучасних AMM-платформах, має критичні обмеження у сценаріях з граничними значеннями ліквідності. Для подолання цієї проблеми запропоновано використання розширеного формату Q64.128, що дозволяє зберігати точність навіть у випадках, коли традиційний підхід призводить до втрат. Описана система забезпечує більш стабільні, зворотні та безпечні фінансові обчислення, що підтверджується порівнянням з існуючими реалізаціями. Використаний інструментарій – Rust, Anchor, Solana, Next.js дозволяє не лише досягти високої продуктивності, а й забезпечити гнучкість подальшого розвитку.

Отримані результати свідчать про доцільність впровадження високоточних обчислень у структуру AMM-протоколів і відкривають перспективи для подальшої оптимізації як в ончейн, так і в офчейн-компонентах децентралізованих фінансових систем. У рамках подальших досліджень доцільно зосередитися на розробці ефективних алгоритмів обчислення квадратного

кореня з фіксованою точністю, оскільки ця операція використовується для розрахунку ціни та константного продукту в AMM-протоколах.

7. Література

[1] Uniswap Developers. Uniswap V3 Development Book. 2017. URL: https://uniswapv3book.com/milestone_0/constant-function-market-maker.html.

[2] Raydium Developers. Hybrid AMM Developer Documentation. 2020. URL: https://github.com/raydium-io/raydium-docs/blob/master/dev-resources/raydium-hybrid-amm-dev-doc.pdf.

[3] A. Яковенко. Solana Whitepaper. 2019. URL: https://solana.com/solana-whitepaper.pdf.

[4] B. Бутерін. Ethereum Whitepaper. 2014. URL: https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf.

[5] P. Хайд. Write Great Code. 2020. URL: https://cdn.ttgtmedia.com/searchEnterpriseLinux/downloads/WGC_Chapter_3.pdf.

[6] C. Клабник, К. Ніколс. The Rust Programming Language. 2023. URL: https://doc.rust-lang.org/book/.

[7] І. Кириченко, Г. Терещенко, І. Груздо. «Застосування симетричних алгоритмів в блокчейні». Біоніка інтелекту, Харків: ХНУРЕ, № 1 (94), с. 33–39, 2020. doi:10.30837/bi.2020.1(94).11

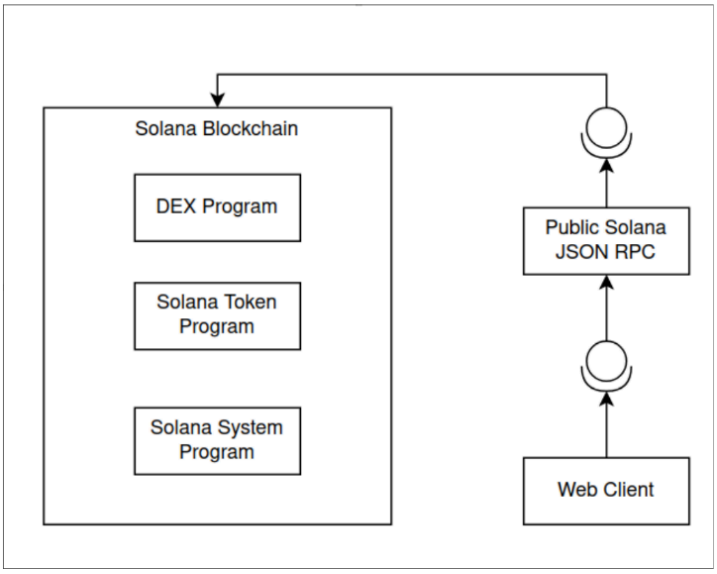


Рисунок 1: Діаграма компонентів