SenSwap: An Open Liquidity Protocol for Token Swap on Solana

Whitepaper - Version 1.0
SenSwap Protocol
hi@senswap.com

***Abstract*—During the early era of the decentralized finance (DeFi), there were numerous decentralized applications (DApp) that had quickly emerged and attracted huge liquidity from the community. However, the environment of DeFi is quite a chaos because several teams built their applications separately with poor interfaces to connect other services. SenSwap (we) aims to solve the problem by unifying the liquidity and opening it up for community services. In this paper, we are going to introduce an automatic market maker (AMM) with pools of a trilogy, 3 tokens. The native utility token, SEN, is compulsory in all pools. By the property, the protocol removes several technical barriers and remarkably reduces the exchange fee regardless of the pair of tokens. Furthermore, we also provide a complete solution for developers can realize their ideas and integrate them into the SenSwap ecosystem. Eventually, the SenSwap ecosystem will be run by the community. DAO might be utilized as a democratic regime to make sure the liquidity can leverage innovative projects and grow the ecosystem.**

***Index Terms*—blockchain, decentralized finance, open liquidity protocol, automated market making.**

I. Introduction

The idea of using Automated Market Making (AMM) to build up a decentralized exchange (DEX) was initially proposed in 2016. Two years later, the first form of the idea, named Uniswap, was launched in November 2018. However, the traffic on Uniswap was quite low for the next 3 years. In August 2020, the liquidity had reached 1 billion US Dollars for the first time. Currently, Uniswap becomes a key component of the Decentralized Finance (DeFi) market that leverages numerous DeFi applications and builds up a robust DeFi ecosystem.

Uniswap is just an example that a great idea is possibly ignored for a long time before someone discovers and gives it a chance to come out. We believe that there are still many great ideas out there and need financial and technical supports to grow. Few projects have realized that limitation and tried to get around it. They built a comprehensive ecosystem with multiple existing services on the market. However, those projects are quite close and the liquidity is not open to the public. As our belief, a lot of great projects are still hidden and waiting for a chance to come out. SenSwap will bring that chance to innovative applications. A well-designed tool-chain is provided for developers to accelerate the development process in the first place. If the developers prove their applications can create meaningful impacts, they will be accepted to use the liquidity by the SenSwap community.

To implement the vision of an open liquidity protocol, an AMM service is proposed to build a liquidity market. Unlike other AMMs, the SenSwap must be carefully designed to create a universal interface that can easily integrate with other services to expand the ecosystem. Therefore, the pool of trilogy is a possible solution where a pool now contains three types of token and SEN is the compulsory token. The model of a trilogy has the following advantages:

- Reduce and maintain an upper bound of exchange fees.
- Unify all fees to SEN.
- Connect all pools via SEN.
- Provide a universal interface of liquidity.
- Parallel decentralized autonomous organizations (DAO).

Besides the merits, pools of a trilogy also have limitations such as symmetric deposits [1], ineffective token distribution, impermanent loss [2]. We have employed a number of sophisticated solutions that will be explained in detail in the following sections.

II. Related work

In 2020, Uniswap introduced version 2 in which they mentioned the concept of programmable liquidity. The concept is similar to an open liquidity concept at some points. They opened and conducted a method to implement on-chain price oracles. We can consider it as open data. Uniswap also utilizes the atomic property of Ethereum to leverage flash swaps. In short, you can ask for tokens before paying them with some conditions. If the conditions aren't met, the transaction will be rolled back. The flash swaps are a superpower for arbitrageurs to explore the price differential. [1]

It's clear that the liquidity openness on Uniswap is still limited when LP Token cannot be consumed by other financial services. In the last half of 2020, a term of yield farming is very noticeable when it accepts LP tokens to earn other tokens. However, these services are dependent on third parties without native supports from Uniswap. [3]

SushiSwap is a younger AMM, however, more innovative than existing platforms before. SushiSwap provides a comprehensive DeFi market with many services including swap, yield farming, lending & borrowing, staking, and others. All these services are built up by themselves which means they fully control the liquidity and circulate it in their platform.

On Solana, Raydium is the current top DEX. Currently, they had services such as swap, staking, and farming. However, Raydium seems to share the same vision as SushiSwap when they maintain the private permission in DeFi services development. [4]

For a recap, it's worth mentioning about the constant product function (CPF) which expresses the pricing curve in the aforementioned protocols. The main idea behind seems very simple, but it has strongly proved functionality, possibility in both experimental [5] and practical. The CPF formulates the quoted price via the pair of token reserves. Giving token A, and B with corresponding reserves R_A, R_B, the algorithm will maintain the following equation,

$$R_A \times R_B = k, \tag{1}$$

where $k > 0$ is a constant defined at the initial state [6]. In other words, let's call α is the "changing rate" of R_A after an exchange (sell A, buy B for example). Then $R'_A = \frac{1}{\alpha} R_A$ induces $R'_B = \alpha R_B$, where $0 < \alpha < 1$, to maintain the constant product,

$$R_A \times R_B = R'_A \times R'_B = k. \tag{2}$$

At an arbitrary timestamp t, the quoted price $p^{(t)} = R_B^{(t)}/R_A^{(t)}$. Therefore, the first liquidity provider must deposit both tokens, A and B, with reserves that satisfy the reference market price at the initial state, $p^{(0)} = R_B^{(0)}/R_A^{(0)}$. After the pool's setup, subsequent liquidity providers must follow the quoted price by depositing both tokens accordingly. It's a problem when users usually have only one token. We call the problem Symmetric Deposits.

When trading, there exists a slippage rate s because of the CPF.

Definition 1. *Let's p be the current quoted price, p' be the next quoted price, then a slippage rate,*

$$s = \frac{p'}{p}. \tag{3}$$

Applying to the CPF,

$$s = \frac{R'_B/R'_A}{R_B/R_A} = \alpha^2 \tag{4}$$

The slippage rate is more amplified when routing. When there are no direct pools for the desired tokens, traders need to go through many middle tokens before reaching the destination. Sometimes you can't even find an available route. This problem, Swap Possibility, reduces the liquidity effectiveness and user experience.

In this paper, SenSwap takes care of three problems of an AMM:

- Swap Possibility
- Symmetric Deposits
- Liquidity Openness

By putting the native utility token named SEN into all pools, the protocol introduces Simulated Single Exposure and Simulated Mesh Exchange which solve the problems of Symmetric Deposits and Swap Possibility respectively. In practice, SenSwap will automatically run these algorithms and provide transparency to users. Regarding developers, SenSwap will open the chance to access the liquidity for people. We believe that can quickly encourage the number of public projects and increase the liquidity effectiveness. Finally, SenSwap inherited outstanding properties from Solana. We make use of quick block confirmation time based on Proof-of-History on Solana and low fee for user attraction [7].

III. Solana Programming Model

At first look, Solana blockchain has an unusual approach to develop the "smart contract" concept and call it "program". A program on Solana doesn't have its own memory, storage, or anything to store states. Solana's solution is extremely opposite to Ethereum [8] which the most adopted smart contract platform. As an interesting metaphor, we can think of Solana programs as punched-card computers, which are the very first versions of the digital computer in the 70s. Such computers need punched cards to provide inputs, algorithms, and even to write the outputs. Within the Solana programming concept, a punched card is represented by an account.

To adapt the novel model and secure user experience, DApps on Solana are required several special features including renting accounts, rent exemption, associated accounts, cross-program invocations, and many others. And pools of a trilogy is such a consequence to structure an open liquidity protocol.

IV. Pool of a Trilogy

Recall that the most well-adapted CPF is two-dimensional where $R_A \times R_B = k$. In SenSwap, the pool is no longer organized into two tokens. It's now the pool of trilogy and the CPF turns to three-dimensional where the third dimension is for SEN as in Fig. 1.

Definition 2. *For a pool of A, B, and compulsory SEN, the 3D constant product function is:*

$$R_A \times R_B \times R_{SEN} = k \tag{5}$$

, where k is a constant.

To deposit to the pool, a liquidity provider (LP) theoretically need to divide their portfolio into three equal portions regarding value. However, LPs never worry about this due to Simulated Single Exposure which allows people to deposit even on one side (see IV-B). Especially, although the formula has a higher dimension, the formula isn't much different from the 2D CPF in trading. When a trader swap A from B for example, the third token, which is SEN in this case, will be ignored. The formula is boiled down to $R_A \times R_B = \frac{k}{R_{SEN}} = k'$ where k' is a constant as well.

Assume a trader swaps r_A for r_B, the newest state of token A would be $R'_A = R_A + r_A$. Because of the CPF, we have:

$$R'_B = \frac{R_A R_B}{R'_A} = \frac{R_A R_B}{R_A + r_A}. \tag{6}$$

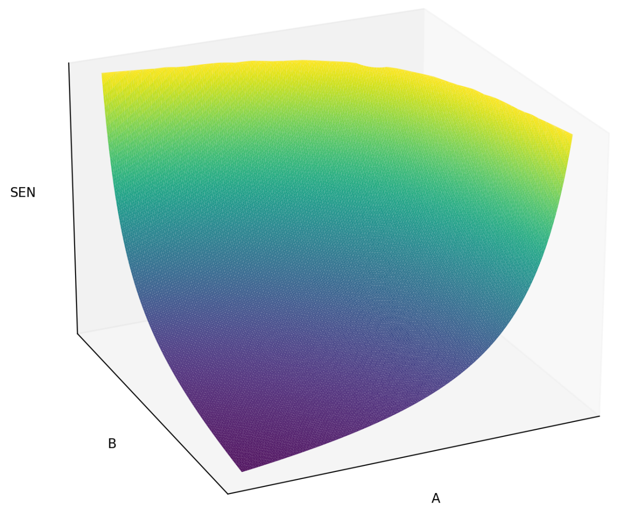

Fig. 1. A visualization of the 3D CPF.

A. Swap Possibility

Because SEN appears in all pools within SenSwap protocol. All tokens will be exchanged through SEN if no direct pool is found. Therefore, the number of routes is always less than 2. We call it Simulated Mesh Trading due to the SEN dependency. This property reminds us of Uniswap v1.

B. Symmetric Deposits

The Symmetric Deposits problem is that LPs must deposit the tokens in a pool with a predefined proportion. To remove extra actions, the algorithm in SenSwap will automatically simulate the swap on the current pool to balance the amount for LPs. Then the subsequent procedures like depositing and returning LP tokens are automated too. Because the process is just to simulate a single-sided deposit, we call it Simulated Single Exposure.

Proposition 1. *Without loss of generality, giving a pool of A, B, and SEN with the current state $\{R_A, R_B, R_{SEN}\}$, an LP deposit Δ_{SEN} to the pool and receive an amount of LP token,*

$$lpt = \Delta_{SEN} - (\delta_A + \delta_B), \tag{7}$$

where $\delta_A + \delta_B = \sqrt[3]{R_{SEN}(R_{SEN} + \Delta_{SEN})^2} - R_{SEN}$.

Proof. Let's $\Delta_{SEN} = \delta_A + \delta_B + \delta_{SEN}$ where the left term is the set of amounts of token SEN to swap for $\{r_A, r_B, r_{SEN}\}$ in the simulation. The simulation is depicted by the following table (we use blackboard characters for state owners):

$\mathbb{LP}$	$\mathbb{A}$	$\mathbb{B}$	$\mathbb{SEN}$
$\delta_A + \delta_B + \delta_{SEN}$	R_A	R_B	R_{SEN}
$\overset{\text{Swap } \delta_A \text{ for } r_A}{\underset{r_A = R_A - R'_A}{\rightleftarrows}}$	R'_A	$\vdots$	$R_{SEN} + \delta_A$
$\overset{\text{Swap } \delta_B \text{ for } r_B}{\underset{r_B = R_B - R'_B}{\rightleftarrows}}$	$\vdots$	R'_B	$R_{SEN} + \delta_A + \delta_B$
r_A & r_B & r_{SEN}	R'_A	R'_B	$R_{SEN} + \delta_A + \delta_B$

Here we don't need to swap SEN thus $r_{SEN} = \delta_{SEN}$. By the CPF (see Eq. (6)), we also know

$$R'_A = \frac{R_A R_{SEN}}{R_{SEN} + \delta_A}, \tag{8}$$

$$R'_B = \frac{R_A(R_{SEN} + \delta_A)}{R_{SEN} + \delta_A + \delta_B}. \tag{9}$$

And

$$r_A = R_A - R'_A = \frac{R_A \delta_A}{R_{SEN} + \delta_A}, \tag{10}$$

$$r_B = R_B - R'_B = \frac{R_B \delta_B}{R_{SEN} + \delta_A + \delta_B}. \tag{11}$$

To satisfy the symmetric deposits, we have a system of equations:

$$\frac{r_A}{R'_A} = \frac{r_B}{R'_B} = \frac{r_{SEN}}{R_{SEN} + \delta_A + \delta_B}. \tag{12}$$

Or,

$$\frac{\delta_A}{R_{SEN}} = \frac{\delta_B}{R_{SEN} + \delta_A} = \frac{\delta_{SEN}}{R_{SEN} + \delta_A + \delta_B}. \tag{13}$$

By transforming the system of equations, we know that

$$\delta_A + \delta_B = \sqrt[3]{R_{SEN}(R_{SEN} + \Delta_{SEN})^2} - R_{SEN}. \tag{14}$$

Therefore, $lpt = \delta_{SEN} = \Delta_{SEN} - (\delta_A + \delta_B)$. ■

C. Liquidity Openness

The most difficult for an application in the DeFi market is liquidity accumulation. Liquidity is an essential component to create a stable, reliable market, then attract users. Many DApps were struggling with these problems over the years while the liquidity was concentrated on a small number of top protocols. It's more often when the liquidity has limited access from the public that causes poor liquidity effectiveness.

Let's examine a typical example for a swap on Solana. An LP deposits a set of SPL Tokens [9] to a corresponding pool and receives an amount of LP tokens (LPT) which acknowledges their proportional contribution. In a way, LPT contents some values so they can be exchanged or mortgaged theoretically. In the early era of DeFi, this point was usually skipped and LPT couldn't be consumed. To optimize the cash flow, a concept named Yield Farming has been introduced. It brought several advantages such as expanding cash flow, diversifying services, and bootstrapping liquidity. After staking LPT into a farm, LP also receives a farming token. In similar to the root problem, no service is available for farming tokens currently.

With an alternative approach, there are a number of lending borrowing services that accept LPT as collateral to mint stable tokens. This solution is more creative and able to tackle the problem that farming has faced. A bottleneck here is that the number of accepted LPT is very limited due to assessment

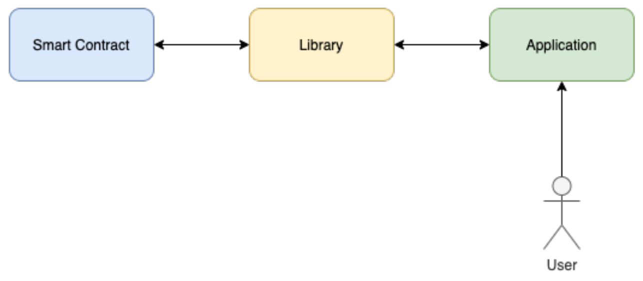

Fig. 2. The perspective of a full DApp is usually organized into three components. An application interface is for a user to interact with the DApp. A smart contract is to run the main logic. And a library helps the application interface call functions and read data from the smart contract. In some usual cases, some backend servers are built to support the application interface. But these servers are optional so we can skip in the general system.

difficulty [10]. By putting SEN in all pools, we can reduce the risk of fake tokens and accelerate the process of assessment for example. Moreover, now all fees can be converted to SEN. It's very meaningful for the foundation to realize the concept "Buy Back and Make". The foundation can allow a fraction of the fee to be converted to SEN and automatically transmitted to other services.

Now, SEN is regarded as a universal interface. Other services can reply on and leverage automated processes.

V. An Open Liquidity Protocol

SenSwap would like to remove the liquidity monopoly problem. Currently, a few liquidity protocols are occupying a large majority of global liquidity while other projects are thirsty for liquidity. Open liquidity is a good way to well circulate liquidity, increase effectiveness in liquidity usage, and encourage innovation from developers in the community.

Open development. In order to quickly realize ideas and delivery to users, an application must rely on many open-source frameworks, libraries. Not to mention that newcomers may put more effort into terminology understanding and technical stacks. This process could be shortened if there were full and detailed documents. Therefore, we orientate SenSwap protocol towards an open-source platform. We will provide a complete solution along with documents for developers. As a basic stack (see Fig. 2), the tool-chain will include:

- A framework to develop smart contracts on Solana.
- A library to communicate between off-chain applications and smart contracts.
- A system UI provides a consistent style for all applications.

Open liquidity. After the application development, the projects will meet the problem of liquidity and users. By integrating themselves into the SenSwap ecosystem, they can inherit the existing liquidity and the current set of users which allows the projects can go faster and win earlier. Especially, openness doesn't mean anyone, any teams can freely access and use the liquidity for their own purpose. In regard to liquidity access permission, the projects must prove their rational purpose and vision. Next, the SEN holders community will vote for favorite projects. The winners will be work directly with the SenSwap foundation to correctly integrate the services into the SenSwap ecosystem. By these strict procedures, we are able to avoid scams or ineffective projects. To realize this objective, DAO will be employed (see Fig. 3).

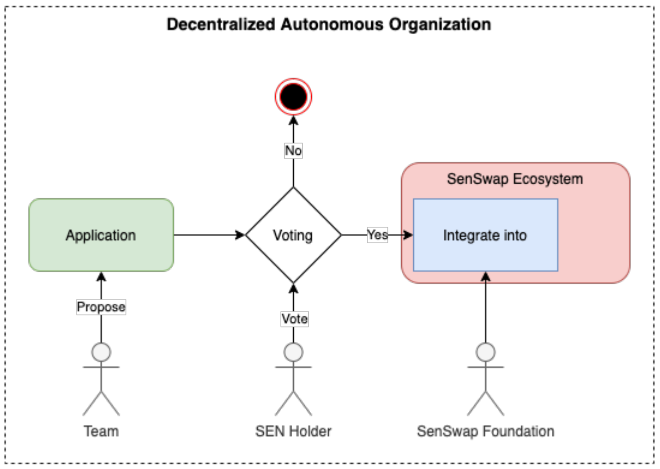

Fig. 3. An application must be accepted by the SenSwap community via voting before being integrated into the SenSwap ecosystem by the SenSwap foundation.

VI. Liquidity Bootstrapping

To encourage people to lock tokens, the protocol needs to bootstrap and build momentum for liquidity in the first place. SenSwap plans to utilize common strategies namely Farming and Auto Compound Farming. Besides, wallet integration is also good for users to have a native experience.

A. *Farming*

Roughly speaking, the SenSwap farming lets users stake their LPT to mint more SEN as rewards. Now users become farmers. The rewards are periodic and proportional to a farmer's shares. Accordingly to the actions of farmers, there are three main functions:

- Harvest: collect the rewards.
- Stake: add LPT to a pool and receive a number of corresponding shares which is equal to the number of staked LPT.
- Unstake: burn shares and remove LPT from a pool.

Since the limit of resources in the Solana programming model (see III), farming cannot naively loop all accounts and update rewards in every period. Hence, we employ debt models to reduce the number of computations. For example, instead of updating the reward of each account periodically, the protocol shall update a single parameter named pool debt to describe how much the pool is owing to farmers.

Proposition 2. *All farmer's actions including harvest, stake, and unstake can be generalized by an order of fully harvest, fully unstake, fully stake where*

- *Fully Harvest: havest all reward to a personal account.*
- *Fully Unstake: unstake all LPT by returning all shares.*
- *Fully Stake: stake a specific LPT and receive a number of shares. Fully harvest is just the same as stake.*

Giving a pool created at timestamp t_0 with the periodic reward $\mathfrak{p}$ and n farmers, the i-th farmer state is $\{\mathfrak{r}_i, \mathfrak{d}_i\}$ where $\mathfrak{r}_i$ is i-th farmer's shares and $\mathfrak{d}_i$ is farmer debt ($\frac{SEN}{periods \cdot shares}$); And the poll state is $\{\mathfrak{R}, \mathfrak{D}\}$ where $\mathfrak{R} = \sum_{i=0}^{n-1} \mathfrak{r}_i$ is total shares and $\mathfrak{D}$ is pool debt ($\frac{SEN}{shares}$). In the following equations, the pool has lived for t periods and we use prime to represent for subsequent notations.

Fully Harvest. This action doesn't change all debts and shares. Then the i-th farmer will harvest $\mathfrak{s}_i$ SEN,

$$\mathfrak{s}_i = \mathfrak{p}t\mathfrak{r}_i - \mathfrak{d}_i + \mathfrak{D}\mathfrak{r}_i, \tag{15}$$

$$\mathfrak{d}'_i = \mathfrak{d}_i + \mathfrak{s}_i = (\mathfrak{p}t + \mathfrak{D})\mathfrak{r}_i, \tag{16}$$

$$\mathfrak{r}'_i = \mathfrak{d}'_i,\ \mathfrak{R}' = \mathfrak{R},\ \mathfrak{D}' = \mathfrak{D}. \tag{17}$$

Fully Unstake. This action must be after fully harvest, and unstake all $\mathfrak{r}'_i$,

$$\mathfrak{r}'_i = 0,\ \mathfrak{d}'_i = 0, \tag{18}$$

$$\mathfrak{R}' = \mathfrak{R} - \mathfrak{r}'_i, \tag{19}$$

$$\mathfrak{D}' = \mathfrak{D} + (\mathfrak{p} - \mathfrak{p}')t. \tag{20}$$

Fully Stake. This action must be after fully unstake, and stake $\mathfrak{r}_i$,

$$\mathfrak{r}'_i = \mathfrak{r}_i, \tag{21}$$

$$\mathfrak{d}'_i = (\mathfrak{p}'t + \mathfrak{D}')\mathfrak{r}_i, \tag{22}$$

$$\mathfrak{R}' = \mathfrak{R} + \mathfrak{r}_i, \tag{23}$$

$$\mathfrak{D}' = \mathfrak{D} + (\mathfrak{p} - \mathfrak{p}')t. \tag{24}$$

For example, you have staked and owned 1 share. You wish to stake 1 more token to own 2 shares. This desire can be generalized by a pattern:

1) Fully harvest.
2) Fully unstake 1 token.
3) Fully stake 2 tokens.

B. Auto Compound Farming

It's clear to see that farming is not quite optimized when the rewards aren't re-invested to earn more profit. In the future, SenSwap will develop an automated process of compound interest for relevant tokens.

C. Wallet Integration

Wallet integration is an attempt to increase SenSwap brand awareness and expand the SenSwap community. Because of the convenient libraries in V, wallets can quickly support SenSwap services as native functions. As a result, the wallet can acquire new features with a cheap development cost, attract more users, and increase traffic. About the protocol, SenSwap can approach the current set of people using the wallets. SenSwap is also able to introduce its services without changing user behaviors much.

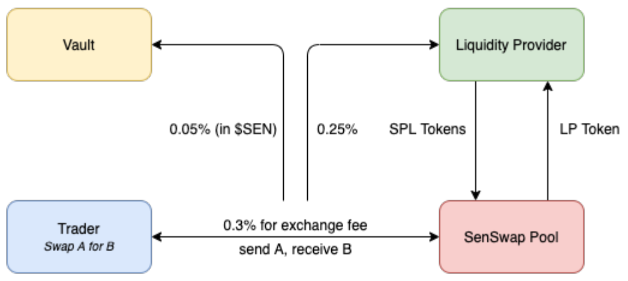

Fig. 4. The fee is organized into two components. The 0.25% is compulsory which represents the interest for LPs. The 0.05% is optional and would be transferred to a vault. This part will be faded if the asked token is SEN.

VII. Exchange Fee

Because the pricing curve is the same as the 2D CPF, the impermanent loss (IL) is obvious to SenSwap too. A fee model is employed to mitigate the IL problem. Furthermore, the SenSwap fee model has learned from centralized exchanges when the native token becomes a discount factor (see Fig. 4).

In summary, if the traders ask for SEN, the fee down from 0.3% to 0.25%.

VIII. Token Use Case

SEN is the native utility token and the heart of the SenSwap protocol. SEN has the following use cases:

Discount Factor. Swapping to SEN can reduce the exchange fee from 0.3% to 0.25%.

Buy Back And Make. SEN is designed as a fee collector. With any swapped pairs, the fee can be easily transformed to SEN.

Protocol Entrepôt. SEN connects all pools in the SenSwap protocol and makes sure any pairs of tokens possible to swap.

Governance. To propose an on-chain improvement, participants need to stake SEN in order to vote and reach a consensus. The ratified proposals will be implemented and deployed by the foundation team.

IX. Tokenomics

Total Supply	1,000,000,000 SEN
Community Growth	25%
Liquidity Mining	25%
Seed	3%
Strategic	15%
IDO	2%
Foundation Reserves	5%
$\$C98$ Holders	5%
Team	20%
Total	100%

X. Conclusions

SenSwap pursues an aspiration beyond DeFi. SenSwap heads to an Open Finance that increases accessibility, optimizes the liquidity effectiveness, and brings more value to users. Also, it creates a supportive environment for developers to innovate. Meaningful projects can instantly access the technical resource and the liquidity which are essential for their success. The more open, the more innovative.

References

[1] H. Adams, N. Zinsmeister, and D. Robinson, “Uniswap v2 core,” 2 2020. [Online]. Available: URl:https://uniswap.org/whitepaper.pdf.

[2] N. Hindman, “Beginner’s Guide to (Getting Rekt by) Impermanent Loss,” 11 2020. [Online]. Available: https://blog.bancor.network/beginners-guide-to-getting-rekt-by-impermanent-loss-7c9510cb2f22

[3] K. Rapoza, “DeFi ‘Yield Farming’: How To Get DeFi Yield, And Why Invest In It,” 06 2021. [Online]. Available: https://www.forbes.com/sites/kenrapoza/2021/06/06/defi-yield-farming-what-is-it-how-should-you-invest-in-it/?sh=5786f4052193

[4] Raydium Team", “Raydium Protocol Litepaper,” 03 2021. [Online]. Available: https://raydium.io/Raydium-Litepaper.pdf

[5] G. Angeris, H.-T. Kao, R. Chiang, C. Noyes, and T. Chitra, “An analysis of uniswap markets,” *Cryptoeconomic Systems Journal*, 2019.

[6] H. Adams, “Uniswap Whitepaper,” 11 2018. [Online]. Available: https://hackmd.io/@HaydenAdams/HJ9jLsfTz?type=view

[7] A. Yakovenko, “Solana: A new architecture for a high performance blockchain,” 2018.

[8] G. Wood *et al.*, “Ethereum: A secure decentralised generalised transaction ledger,” *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[9] solana-labs", “solana-labs/solana-program-library,” 06 2020. [Online]. Available: https://github.com/solana-labs/solana-program-library

[10] D. Lau, D. Lau, and S. Teh, *How to DeFi*. Independently Published, 2020. [Online]. Available: https://books.google.nl/books?id=g6WazQEACAAJ