Fluidity

**eTech Challenge Product Brief**

**Shahmeer Chaudhry, Nathan Shephered, Eli Demoulin, Jolene Loo, Tony Nguyen, Kosta Tsekouras**

Fluidity is a cryptocurrency that is meant to be used. It is used as an alternative to US Dollar based Cryptocurrencies (stablecoins). Fluidity aims to reward users for spending money. Fluidity takes user's stablecoins and exchanges them for Fluid Dollars, when the user uses Fluid Dollars, they are awarded potentially a lottery prize. Fluidity is for everyone. Certainly, there are groups of people that stand to benefit more through its use, such as those who may not be able to save money easily as they're required to expend their entire paycheque to cover living expenses. However, anyone who sends or receives money using Fluidity has the potential to win a sizable amount of currency in exchange for simply using the service. As highlighted above, everyone could benefit from utilising Fluidity, although, our primary target initially would be people who are already familiar with cryptocurrencies. There are additional complexities associated with allowing non-crypto users to properly utilise our service, however these users will also become target customers further down the development timeline.

The old way of spending money was, put simply, to not spend it at all. In the past, the greatest benefit came from being able to save money for long periods of time, and grow it through interest or investments. Many people in this day and age do not have the ability to hold on to large amounts of money for long periods of time. The new way of spending money requires ordinary people to spend money frequently to cover living expenses, subscriptions, and other necessities. This, unfortunately, provides people with little room for financial growth and makes it very difficult for people to gain exposure to saving money. Fluidity aims to solve this problem by rewarding people with a chance to win a sizable amount of money, just by using the coin to continue their normal spending habits. The way that our solution would function can essentially be thought of as providing a user with entries into a lottery just for spending money as they already would on our platform. Should the user that has spent money be selected randomly as a lottery winner, the user will receive 80% of the pool of the lottery, and the merchant receiving the user's payment will receive 20% of the pool of the lottery. Not only does this stand to benefit the user, but also the merchant, for accepting Fluidity as a form of payment. No fee is taken from the user when they spend money using the platform.

The only remotely similar concept to Fluidity that is in existence is the idea of a "no loss lottery", such as PoolTogether. This functions by asking a user to deposit any amount of Ethereum, in exchange for entries into the no loss lottery. Hence the name, "no loss lottery", the user will not actually lose any of the money they use to enter into PoolTogether, and they can withdraw their entry at any time for no cost. The primary difference between PoolTogether and Fluidity is that the solution that Fluidity implements accounts for the fact that people might not have money that they are able to set aside to enter into a no loss lottery. The aspect of Fluidity where people are essentially provided entries into a no loss lottery without having to sacrifice, even temporarily, any of their money is unique to our product.

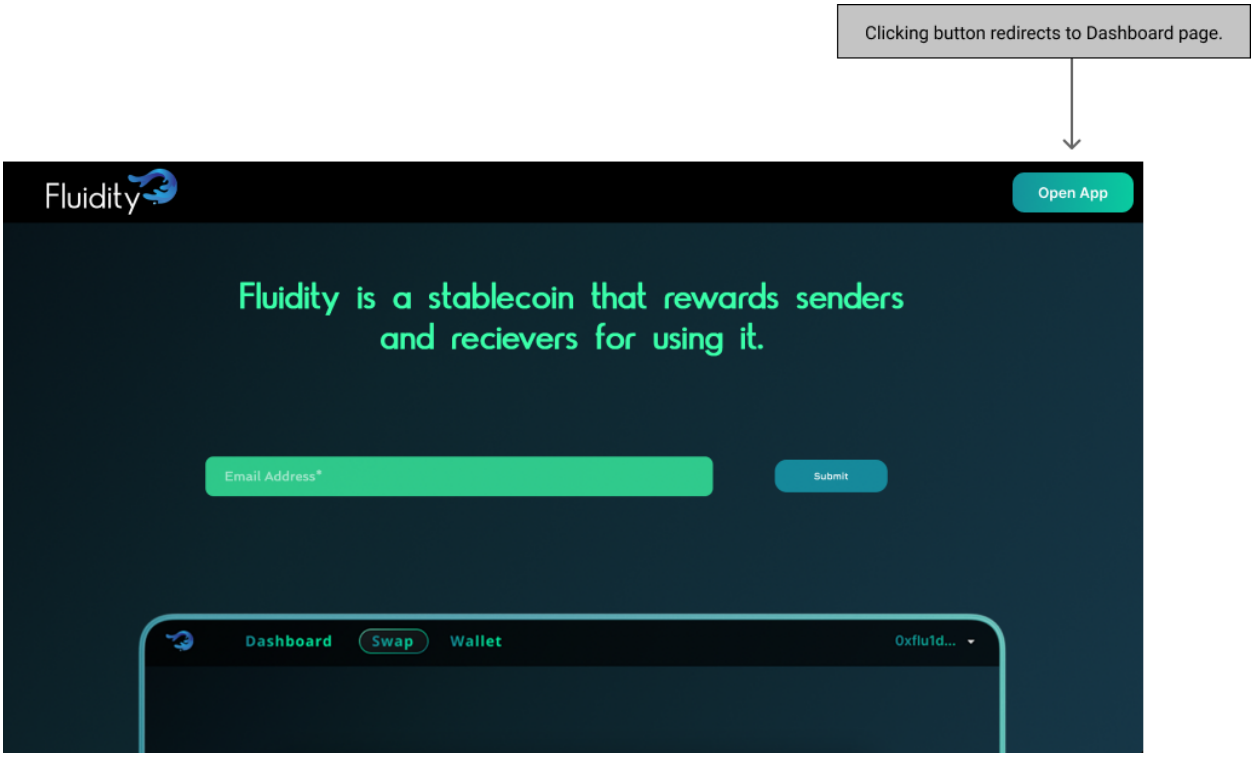
Since the Fluid Dollar can be used with any Cryptocurrency Wallet of the users choice. The main feature of Fluidity is the Backend Algorithm and Agents that invest money and reward prizes for

users to use the Fluid Dollars. This means there is not that many user facing screens as the functionality of Fluidity is in the back end. There exists a Front End where the user can create or burn Fluid Dollar and review the Prize Winners and see information about their Fluid Dollar holdings.

> Mock-up of what it will look like highlighting the design features that resulted is analysing/mapping jobs
>
> - UI Mocks

*UI Mockup*

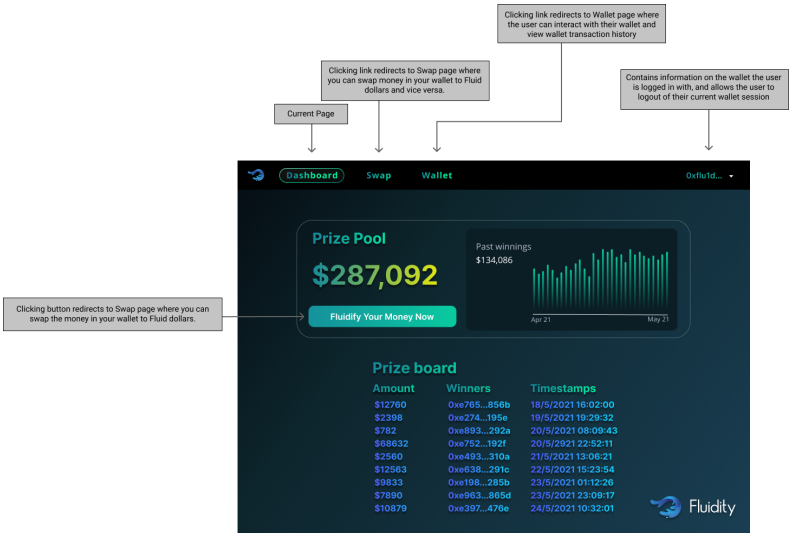
Landing Page ~



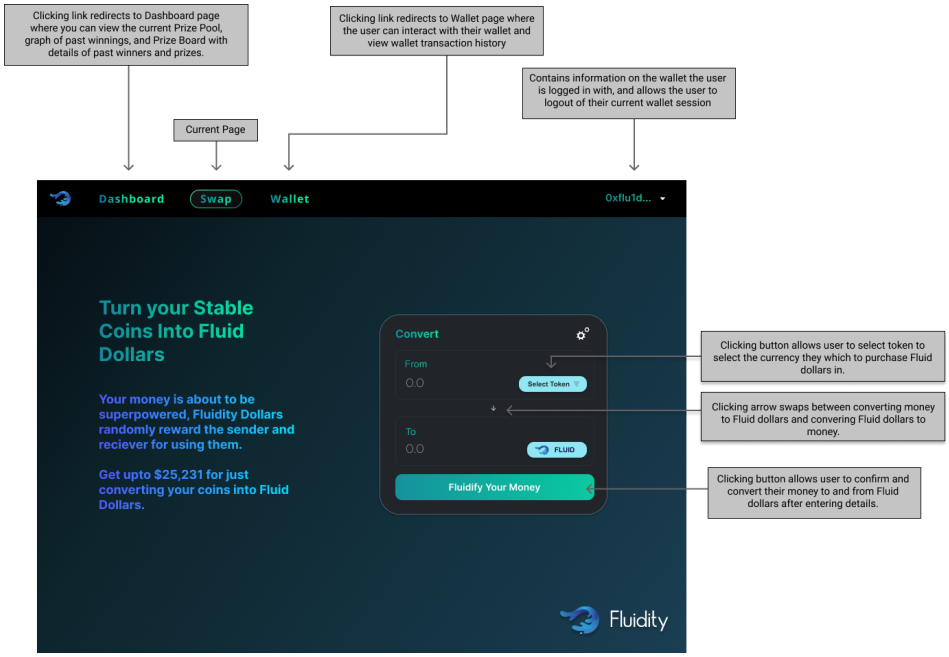
Fluidity

When the user is logged in with their wallet ~

## Dashboard Page

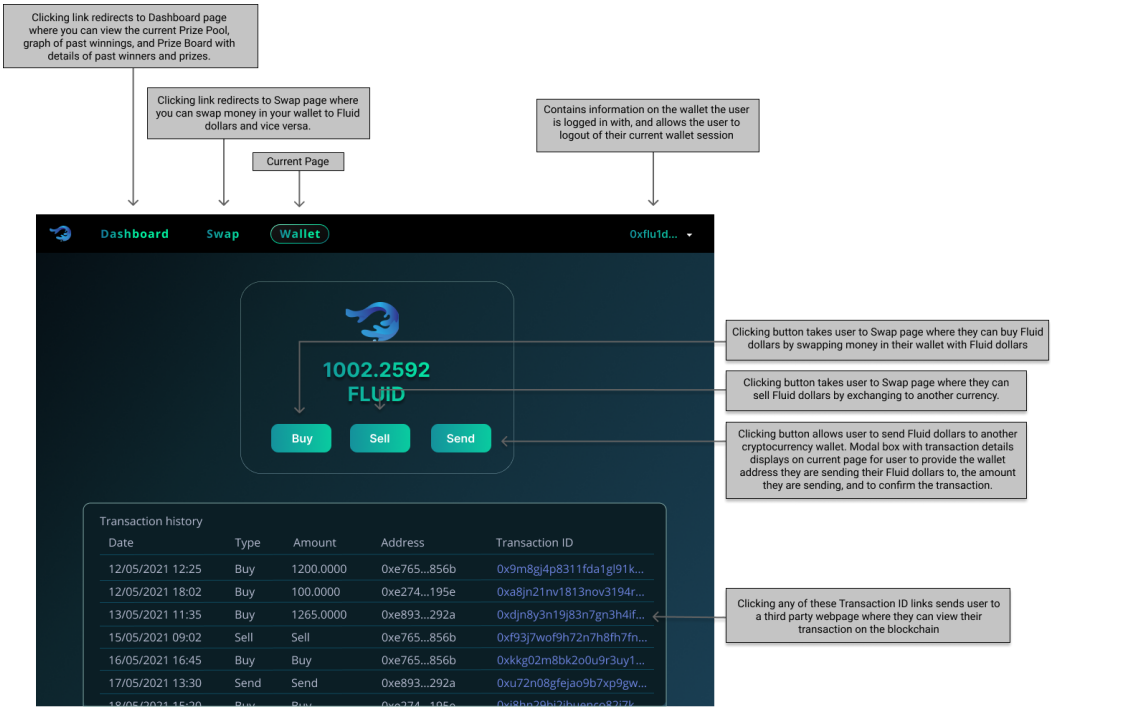


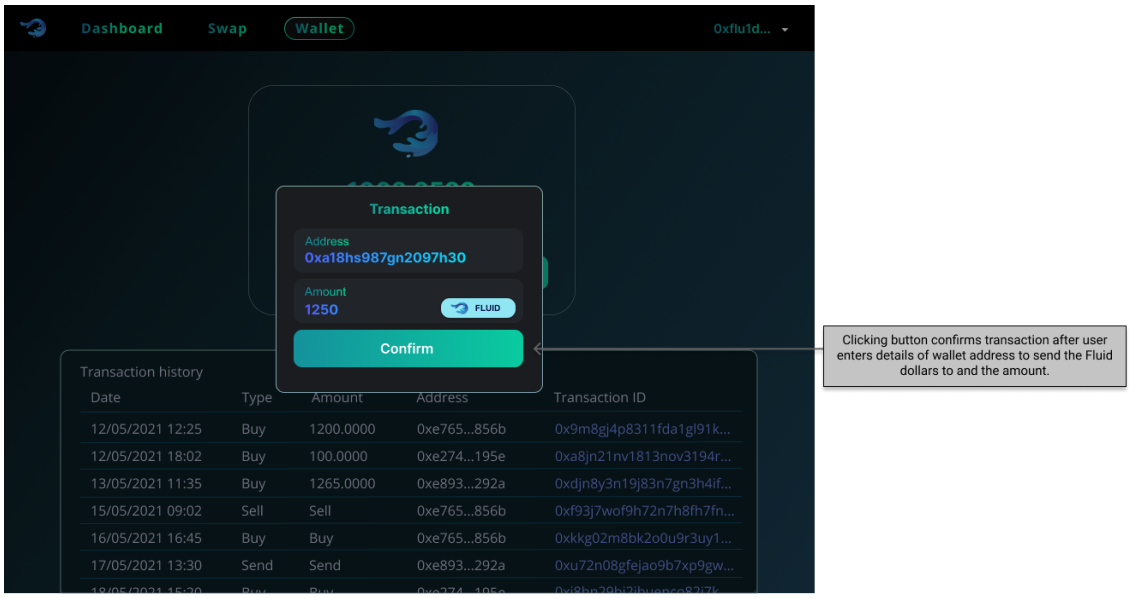

## Swap Page



Fluidity

## Wallet Page

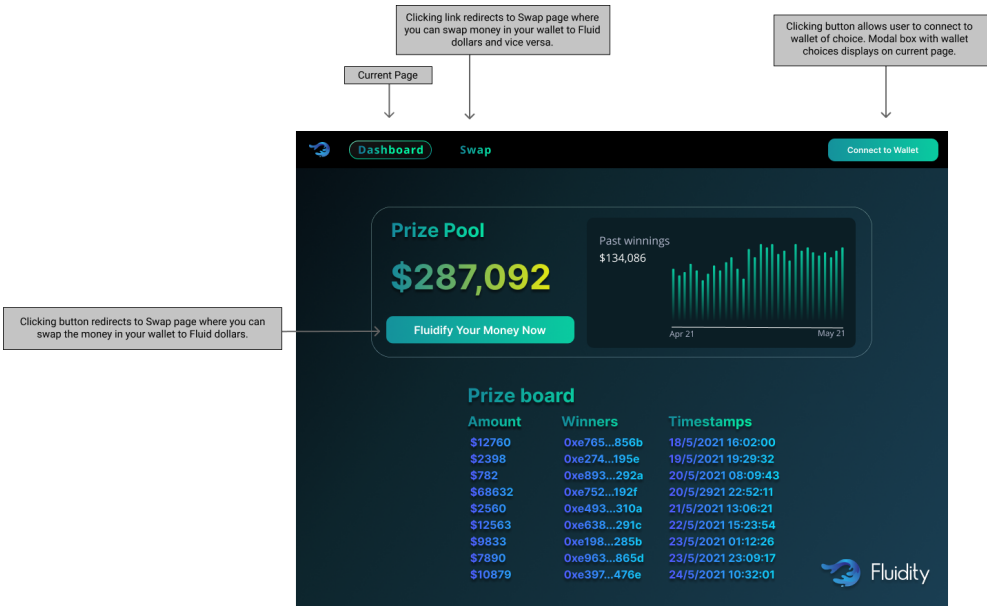


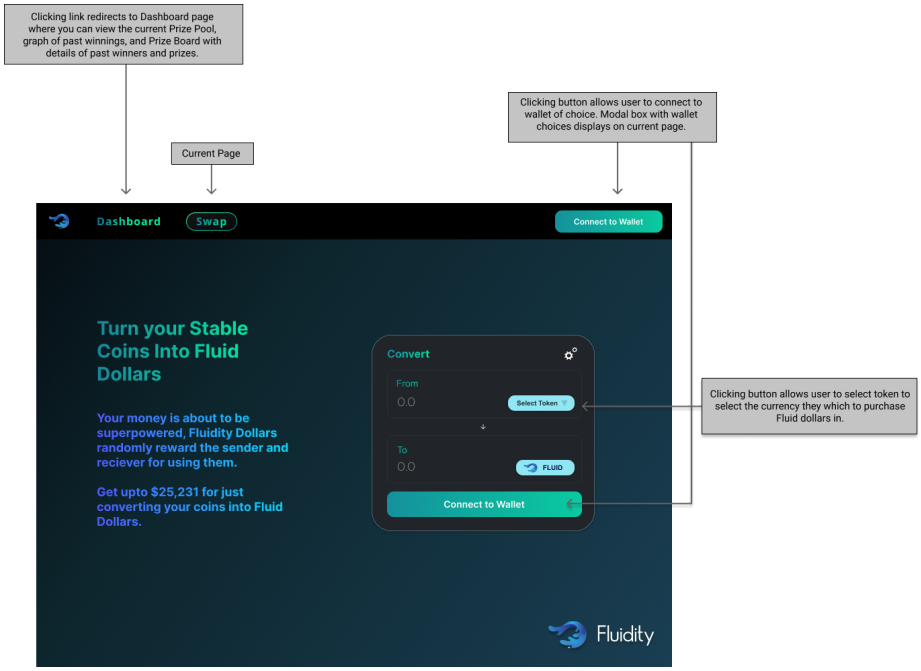
Fluidity

When the user is logged out with their wallet ~

## Dashboard Page




## Swap Page

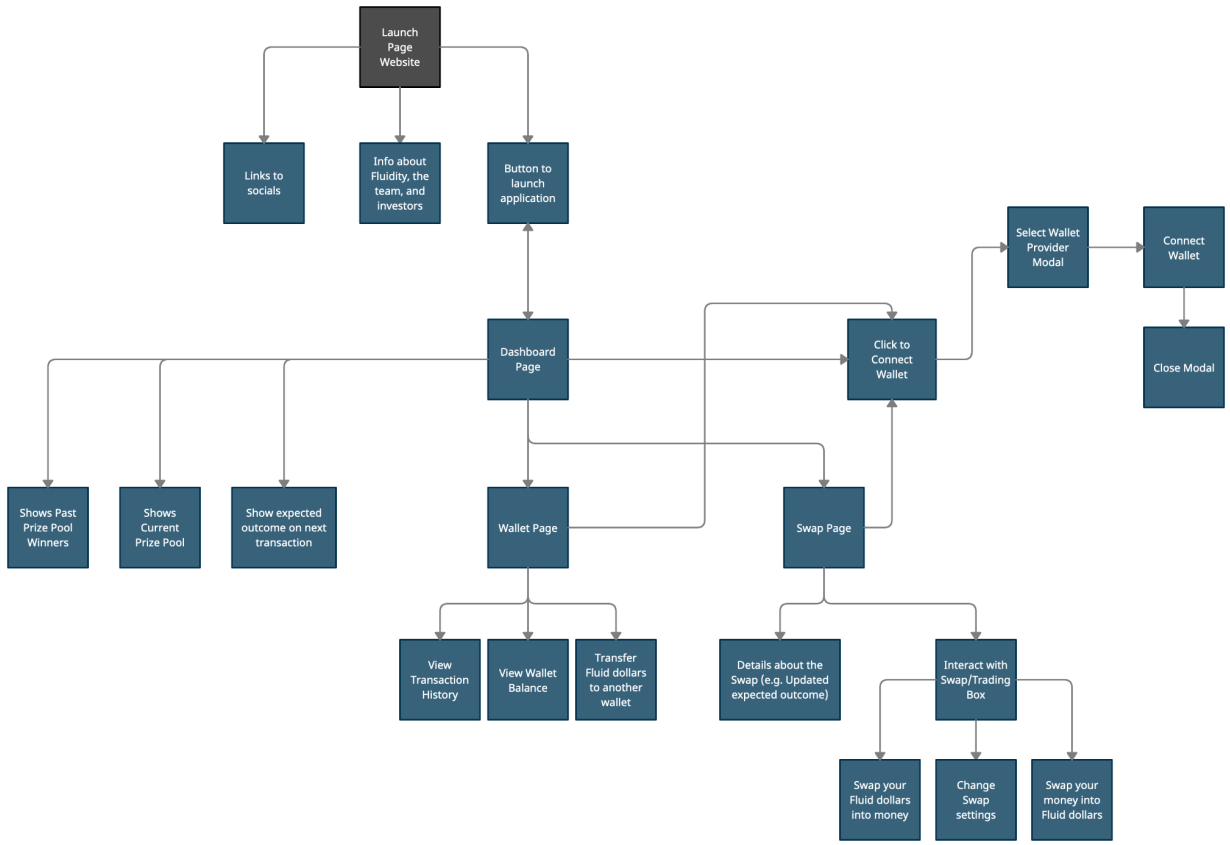


Fluidity

https://app.creately.com/diagram/8ADgsEQn8sZ/edit
https://www.figma.com/file/DxARWR0MvC60s53RQFnGHu/Fluidity-app-website?node-id=274%3A6182

Fluidity

***App Map***




**Expected development costs and timelines to reach the minimal viable product**

We have estimated that it will take between 2 to 4 months to launch a functioning Minimal Viable Product. This will allow us to hit our functional requirements and make the application usable.

Most of the expenses that would be accrued are expenses related to development costs and legal costs. There will be little to no marketing costs and likely will be done organically.

We have been awarded two grants, roughly $50,000 AUD each that will allow us to have a runway to build our MVP and at latest by August. Although it is to be noted, these grants are also unlocked as we hit deliverables so there is a possibility that we may not have access to the capital when needed, but our timeline does aim to follow our expected burn rate and hence we should unlock our tranches as we need the funding.

We estimate that we will spend roughly $17,500 AUD +- ($5,000) a month on salaries paying 4 to 5 employees, part time (15-20 hours a week). There is also a one time cost of $32,500 to pay for legal expenses with respect to incorporating the companies and protecting the interests of the

team members, stakeholders and investors. These Legal Fees are essential for launching the MVP due to the nature of our product. Here are the charts that are modelling our expected burn rate that project that our runway will last us until roughly August. It is possible that we may need to source another grant to be able to go beyond August, however we are optimistic of achieving our goal of building the MVP, within the expected runway we have. It is to be noted that the initial spike in April/May is due to foreseen legal expenses. We have modeled three scenarios which we believe may potentially play out in the cost and timeline of the MVP.

## Fluidity MVP Potential Launch Scenarios

| Case | Launch Month +- 14 days | Cost of MVP +- $10,000 |
|---|---|---|
| Optimistic Case | Late July | $85,000 |
| Average Case | Mid August | $100,000 |
| Pessimistic Case | Late September | $125,000 |

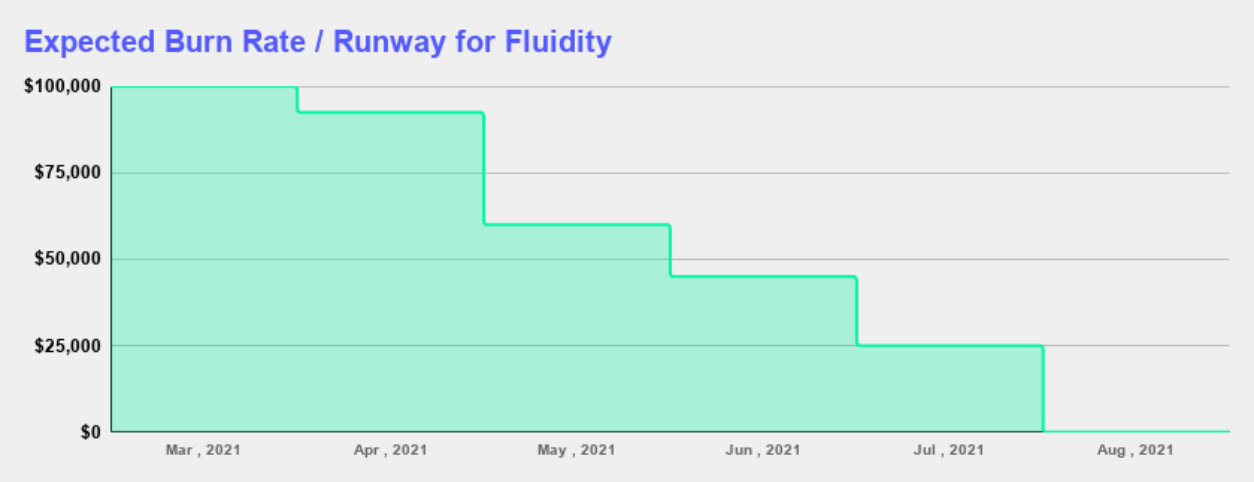



Figure 1: Expected Cash Flow Burn Rate on Fluidity based on Average Case estimation.

The system largely functions through “smart contracts”, meaning that anyone can interact with it using their own code and accounts, independently of us or our website. The resulting architecture is essentially a modular backend and frontend system, where the backend consists of contracts that allow for swaps and the reward mechanism, and the frontend (website) interacts with the backend to send and receive information. The backend has several elements:
The swap mechanism will receive a user’s external tokens, mint equivalent Fluid tokens that are given to the user, then deposit their tokens into an investment agent. The investment agent is a smart contract that invests tokens into a Decentralised Protocol, thereby earning yield which generates liquidity for the reward mechanism. The reward mechanism system describes how a

transaction is chosen to win, and how its reward is paid out in a way that cannot be influenced by users of the system. This is the most complex aspect of the system design, as it has to be written by us (as opposed to using an existing external system), and must be robust in its interface to avoid the possibility of rigging or other abuse.

From a user's perspective, the application has similar functionality - swapping, rewarding, and showing account details. However, since most of the system complexity occurs on the backend, the frontend is far simpler. The account dashboard allows a user to navigate between system functions, as well as providing them their address, transaction history, and notifications when they receive a reward for spending. Swapping will involve an interface where a user can specify their currency and amount, then see the result of their interaction. Since transactions take place independently from the user frontend, it will track any rewards they receive and notify them by sending an email and displaying a notification on the dashboard.

**Technical Considerations**

The part of the system that requires the most technical consideration is the reward mechanism. This mechanism has to be built from the ground up, and is the most vulnerable to undue influence from users. A key problem here is randomness, and how to most effectively select a random transaction. In computing, randomness can be a difficult problem, as computers are incapable of generating "random" information on their own and instead utilise pseudo random algorithms that generate sequences of numbers based on some starting value, known as the "seed". The problem with this is that if someone knows the seed, they can reproduce the exact sequence of "random" numbers that will be produced, allowing for them to in this case work out exactly which transaction will be selected to win and then performing a transaction at that time. As a solution to this problem, we are considering the use of various systems that will allow us to obtain true randomness from some external source, then select winning transactions from a pool by using an algorithm that can be verified but not easily predicted. This has the joint benefit of not only reducing the risk of rigged rewards, but also increasing user trust in the system by verifying that we have not modified the result.

As mentioned above, serious consideration must be placed on the design of the system to ensure a robust interoperability between its numerous components and prevent outside influence. Each component in the system must be designed firstly with an interface in mind via which it interacts with every other component, then designed in a way that matches its interface. As part of this, a large consideration is paid to the security of the system. Smart Contract Security is a significant consideration for Fluidity in the long term. Currently there is greater than $100 Billion AUD locked in Decentralised Finance products, and this year alone hundreds of millions have been lost to hacks and exploits. Although we have no custody of the assets, we are providing systems that allow for complex transactions to occur on behalf of our users, including interfacing with investment agents and rewarding agents, so it is important to look for security issues not just at each level of the system, but between each stage of interaction and broadly between all

components. Hence we eventually would need to get Technical Security Audits on our code to ensure they are not vulnerable to exploits. These are not necessary for the MVP, but need to be considered as we built towards the MVP and done as soon as possible.