Arbitrage on the Solana Blockchain

Critical Design Review

Resol

Robert Elizondo

Anu Khatri

Grant Tugwell

Veronica Wisor

Department of Computer Science

Texas A&M University

March 20th, 2022

Table of Contents

1 Introduction

Need Statement and Problem Background:

With cryptocurrency exchanges being something relatively new in the world of finance and trading, our team aims to capitalize on the emerging industry and its complexities. With there being so many Decentralized Exchanges (DEX) on the Solana network, there are often subtle price inconsistencies between these exchanges. Our aim is to take advantage of these exchanges by creating an arbitrage that will work on the Solana network. These types of programs currently exist online, but we have found that none of them have an intuitive and user-friendly interface, making it more difficult for an inexperienced or a not tech savvy trader to take advantage of the opportunity. We plan to create a profitable arbitrager and give the user an informative UI that describes the actions of the program.

Design Constraints:

With this project, the team has a few constraints that will need to be worked around to achieve the overall goal. The main constraint is the time that we have to complete the project, which is less than 10 weeks. Another constraint is the economic constraint. In order to maximize profit in an arbitrage, more money is always better because the margin of price difference between the purchase and sell is amplified by quantity. However, there is a budget of $600 that the department has given that will need to encompass all expenses, including hosting, so not much can be utilized for arbitraging. There is also a meeting constraint outside of class time. We have other classes and jobs to juggle at the same time, and we need to be able to find time to meet with each other in order to get the work done outside of class if class time is not enough. Lastly, our group has a knowledge restraint about cryptocurrency and blockchain. None of the team members have any experience in this space, but we are all very excited to learn more about this topic and make the arbitrage successful.

The team plans to discuss funding for the arbitrage during the next meeting, in order to begin making larger transactions.

Validation and Testing Procedures:

Validation and testing will be very important for the success of the project, as there will need to be lots of tuning in order to make the system profitable. There are very tight tolerances in this space and one strategy may prove to be more successful than another. The team will have to try these different strategies in order to find out which are the ones that work best for this project. This will be achieved with the wallet monitoring system and will then require analysis of the findings that it returns back to the team. Another key objective is to ensure that the program is stable over an extended period of time and is able to handle errors. This is important because in an ideal situation, the program would be running for long periods of time in order to maximize profit and utility.

The team is starting to implement these features and plan testing, as the transaction function is beginning to come together.

2 Proposed design

2.1 Updates to the proposal design

The team's strategy has definitely changed over the course of the project. Due to this subject matter being in current development and new to the entire team, there have been changes of plans. The biggest problem that we have faced in this stage of the project has been finding which AMMs should be used in order to make a successful arbitrage program. This has consumed a lot of time as there are many different AMMs to try. After testing multiple AMMs, the two that have proven to work the best with our project are Orca and Jupiter.

Another design update the team has made has to do with whether the transaction and price monitoring modules were on chain or not. We originally wanted both the transaction and price monitoring modules to be on chain. We have since realized that doing this is more difficult than originally expected. Currently, the team has focussed on developing a proof of concept for both modules that exist off chain using client side amm APIs. However, going forward, we plan to move the transaction module back on chain as originally planned. The future plan for the price monitoring module is to keep it off chain.

Both of these redesigns stem from difficulty to find documentation online that supports the goal of the project. For most of the Solana projects that the team has tried to integrate into the project, there has been a struggle to fully understand what is going on under the hood of the sample code provided. This has made it difficult to pin-point which AMMs would work for the project. This has resulted in lots of testing and verification.

One final update that was made has to do with the Solana Node being used. We originally were using the free mainnet node for RPCs to make transactions locally. However, after reaching out to some developers who work on the amms, the team found that the free node doesn't always work properly. For this reason, the team decided to upgrade to a Solana Node for RPC connections that costs $9/month. This has significantly increased the success rate of our transactions made through API calls, so this update to the original design was the best choice to make.

2.2 System description

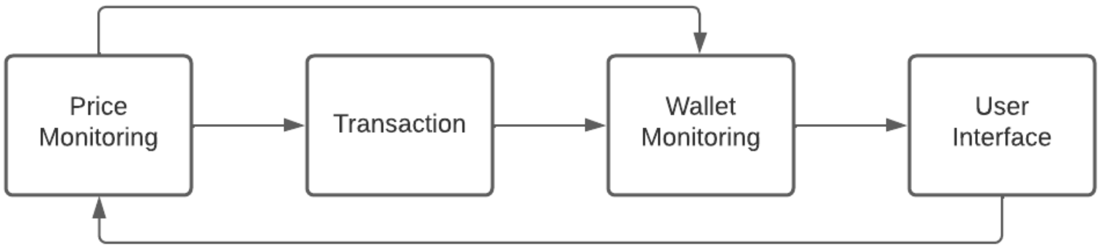


Figure 1

The Arbitrage System is made up of four high level modules: the Price Monitoring module, the Transaction module, the Wallet Monitoring module, and the User Interface module.

Price Monitoring:

A key component of any arbitrage system is to be able to track prices quickly and accurately. The windows of time that arbitrage can be successfully performed can be very small and having a system that can rapidly detect those occurrences is pivotal. The price monitoring system is hosted off chain and it is the entry point for the arbitrage process. The price monitoring system makes constant requests to the different AMMs in order to obtain the most current and accurate prices, and then it will alert the transaction function. Having the best transaction function or clever arbitrage strategy will be insignificant if the monitoring system is not on par with the rest of the system, otherwise it could be a limiting factor for profit.

Transaction:

The transaction function is the heart of any arbitrage program. The transaction function makes a buy and then immediately sells, and profit is made on the difference between the two. There are many different strategies for this and it can get quite complicated. The transaction function is hosted off chain and is able to communicate with two AMMs in order to make this possible. Typescript is used for the transaction function. This function can also be a performance bottleneck and it plays a vital role in the success of the program. If the function is too slow, it is possible to lose the opportunity for arbitrage in the time it took for the transaction to be carried out.

Wallet Monitoring:

Wallet monitoring is what will allow the team to track how the program is performing. This program will log all the transaction attempts and will then do analysis on this data in order to provide relevant and useful information to the team for tuning the transaction perimeters. This will also be used on the UI in order to display this information to the user and give performance insights. This will be done in Typescript, and will take the output of the transaction function as input. This is a key component of the program, as the team believes it will help with optimizing the transaction function and will make the UI more unique compared to those already implemented.

User Interface:

The user interface will serve as a way for the user to visualize the performance of the arbitrate program. The user interface will allow the user to begin the program and terminate, along with being able to see key information that will keep the user up to date. We find this component to be important because most of the other arbitrage programs that we have found on the market do not have a comprehensive UI and are instead used through the command line interface. This part of the project will be completed using React and will display significant graphs on the homepage for the user's convenience.

2.3 Complete module-wise specifications

Price Monitoring:

Currently, the price monitoring module is split into two different programs: price monitoring for individual AMMs, and the Jupiter price monitoring. For the demonstration, only the Jupiter price program will be used. Both of these programs are off chain and use several APIs to fetch current exchange rates.

The price monitoring for individual AMMs can fetch current exchange rates for at least SOL/USDC from Raydium, Serum, Mango, and Orca. To get a current exchange from each AMM, it calls each AMMs API.

This is currently a very simple program that decides whether a SOL -> USDC trade should be made between Serum and Orca.

The Jupiter Price Routes program uses an API call to a price aggregator called Jupiter. This is the program that will be shown during the demonstration. The API call returns several different price routes for the in and out tokens provided which are organized from most to least profitable. These price routes can be direct (in token -> out token), or can take several steps (in token -> middle token -> different middle token -> .. -> out token). Additionally, this API call can use almost any input and output tokens and will look at several different amms: Aldrin, Crema, Cropper, IDL, Lifinity, Mercurial, Raydium, Saber, Sencha, Serum, and Orca.

Our next steps for this module is to be able to use these programs to interact with our on chain program. The main issue currently that will need to be addressed is the speed of these programs. Due to these programs using API calls, the response time of this module is currently very long. We hope to decrease this response time in the future by using better time saving logic and potentially adding multithreading to make multiple calls at once.

Transaction:

The transaction module can successfully make transactions between Orca and, using the Jupiter API calls, any AMM that Jupiter supports (Aldrin, Crema, Cropper, IDL, Lifinity, Mercurial, Raydium, Saber, Sencha, Serum, and Orca). All transactions currently being executed in this module are off chain using API calls to the various AMMs.

With the price routes from the Jupiter API, the transaction module can use Jupiter to then execute these price routes. While this is very successful, it is very slow due to all the API calls that the transaction module makes as well as all the calls made internally with the Jupiter API.

Currently, there are two issues that will be resolved moving forward: speed and atomicity. These issues will be addressed by moving this module on chain. By moving transactions from local API calls to on-chain, we will no longer have to go through middlemen. We will instead be able to directly send and confirm the transactions on chain. Additionally, with the current design of the transaction module, if one transaction fails or if no profit is being made, we lose money. By moving this on chain, we will be able to check for failures and money loss before confirming the transaction. This atomicity of transactions will ensure a profit is always being made.

Wallet Monitoring:

The wallet monitoring system is in the early stages of development within the project. As the transaction function comes together and is able to successfully execute trades, the need for monitoring is now coming into play. The goal is for the translation function to return meta-data about the transaction attempt and this will be given to the monitoring program as input. This may look something like:

{Status: Complete, Market A: ABC Market, Market B: XYZ Market, Coin: SOL, Price A: 100.00, Price B: 99.00, Margin: 1.00, Date: 2/10/2022, Time 8:00}

The wallet monitoring system will then append this transaction to a storage system such as a database or a local file for quicker access. The system will then be able to read and query from the stored transactions and analyze the data that has been produced by these trades. Different strategies can be placed in different storage locations and then analysis can be done on them to see which produces more profit. For example, strategyA and strategyB will both be given $5 and 24 hours to run. All of the transactions and their metadata will be stored in their receptive locations and once the testing period is complete, conclusions will be drawn from the data. This will help in finding which strategy is better and perhaps lead to a combination of multiple strategies. This tuning process will be key to optimizing the profitability of the system.

This system will also play a large role in testing and validation. At the moment, the best option would be to run the program on a dedicated computer and leave it running for extended periods of time, while gathering data. This will allow the team to find and eliminate bugs or crashes that happen when running for longer periods and will make the program better at handling errors.

Lastly, this part of the project will also integrate with the user interface. The monitoring system will produce the data that will be displayed on the dashboard for the user to see. It will also produce alerts or messages that the user needs to see on the frontend.

UI:

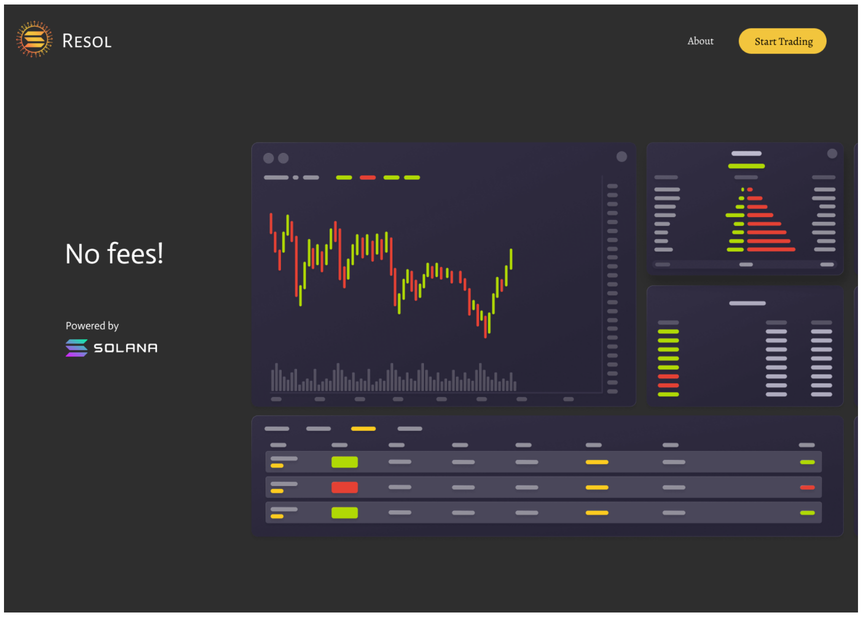


Figure 2

The UI will include a way for the user to add their wallet, as shown in Figure 3. This will be implemented with Solana's wallet adapter, which allows for the capability to add different wallets, including Phantom and Sollet. Since there is already some React UI that Solana has already implemented as part of the source code, it will be a lot easier to use for our own application as far as connecting and using the wallet that will be added by the user.

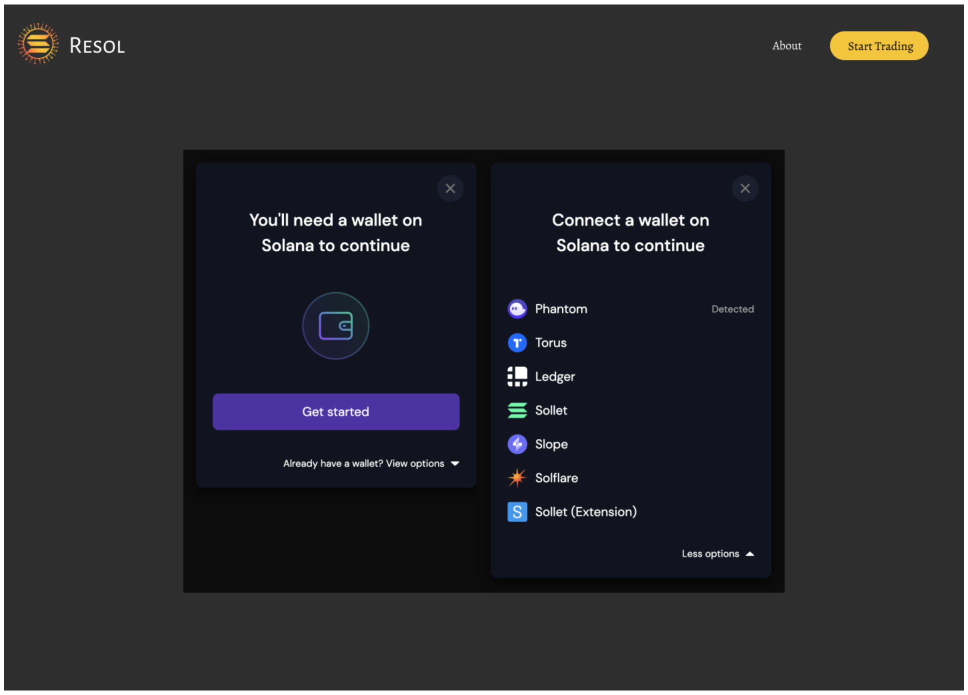

RESOL
About
Start Trading
You'll need a wallet on Solana to continue
Get started
Already have a wallet? View options
Connect a wallet on Solana to continue
Phantom
Detected
Torus
Ledger
Sollet
Slope
Solflare
Sollet (Extension)
Less options

Figure 3

There will also be a display of prices that will be pulled from the price monitoring from the individual AMMs or from Jupiter that is being done on the backend. Additionally, there will be visualizations, such as graphs and charts that display performance metrics from the arbitrage. Some important metrics to display will be, total profit, number of transactions, number of completed transactions, number of failed transactions, daily, weekly, and monthly profit breakdown, and breakdown of which cryptocurrencies are producing the most profit. Once the user has added their waller, it is anticipated that the dashboard will be similar to Figure 4.

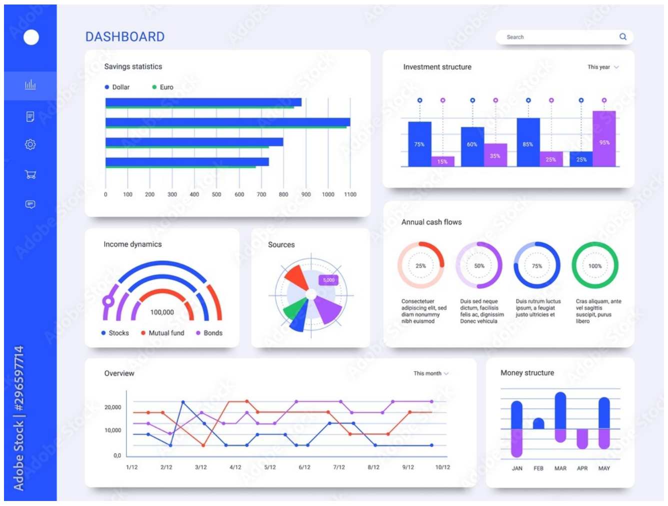


Figure 4

3 Project management

Anu is the team leader, so she creates the weekly agendas and reports so that the professor and the TA are informed on a weekly basis. Additionally, she coordinates the weekly meetings, as well as takes minutes for each meeting so that the rest of the members can also look back at them in the future for reference. She is doing technical reporting due to her experience through previous projects. She also will track the progress of each member's work, so there is proper coordination and progress in the overall project. She will work on the user interface once the command line application is working so that customers can easily add their wallet, track the markets, and start/stop arbitrage.

Veronica is responsible for the AMM monitoring module due to her previous experience in multi-threading. This will continue to be developed so that it may be put on chain for faster, more up-to-date monitoring so that transactions are more likely to be profitable.

Grant is still the finance expert since he has the most experience and knowledge in cryptocurrency. He has been responsible for connecting with AMMs and creating transactions to swap tokens. He has also been the point of contact for other members of the team when they are unsure of cryptocurrency concepts.

Robert is responsible for testing and validation. Hosting will be done by Robert due to previous experience with AWS. He will also aid with UI.

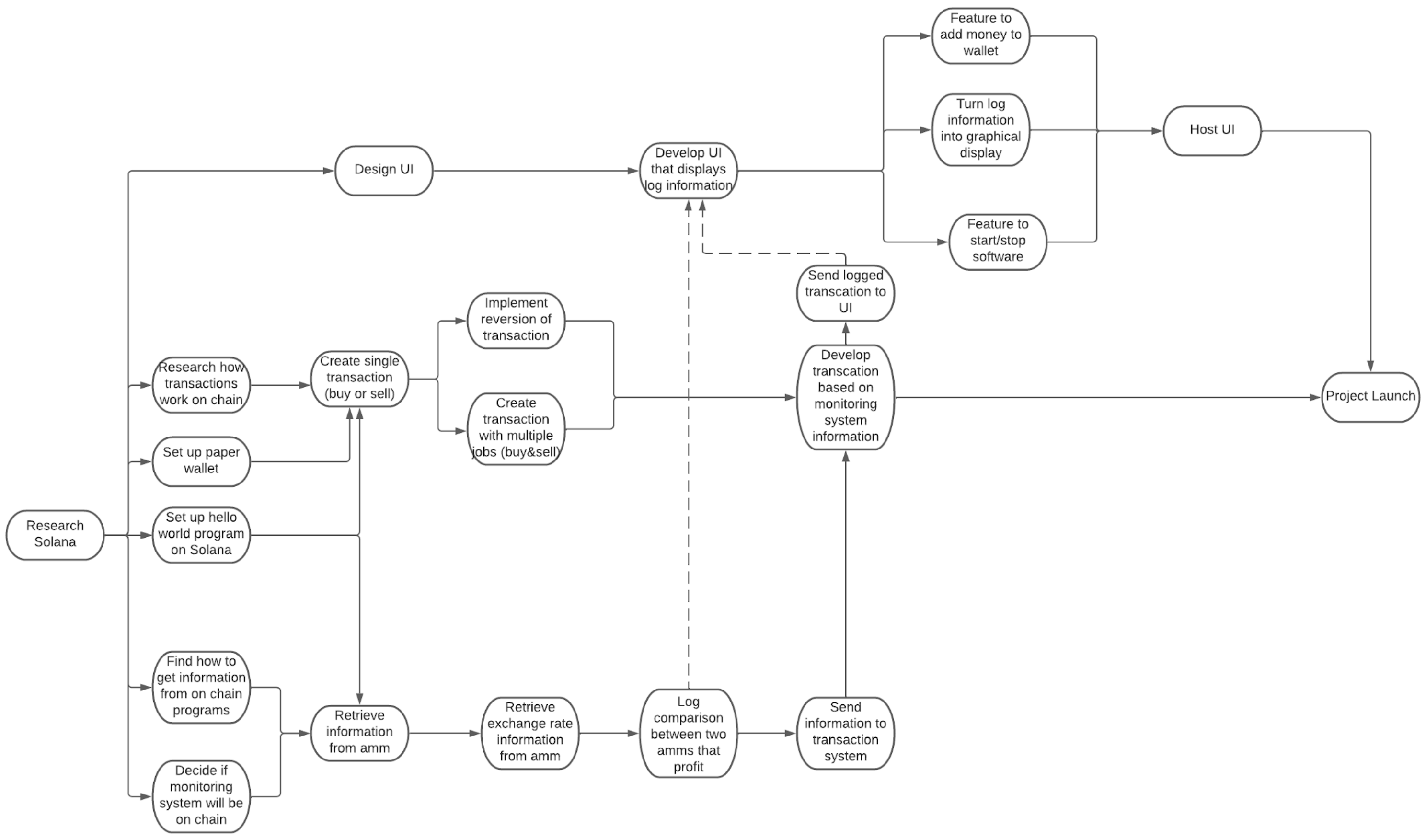
Everyone is responsible for the correct use of version control through Git for safe development of the application.

The team has been managing its progress through taking notes of all meetings, including brainstorming sessions, standups, etc. The meeting notes are added to the Git repository each time. There are progress checks on Monday, Wednesday, and Sunday. We have struggled in figuring out each person's tasks, so people are floating between tasks, so a way we are working on solving this problem is having clearer delegation during our meetings. Additionally, we will make better use of the JIRA board so that it is easier to track progress of each of the members rather than relying on memory or having to look back at meeting notes. Hopefully, this will help each member have a better idea of their responsibilities, so that we can progress better with the project.

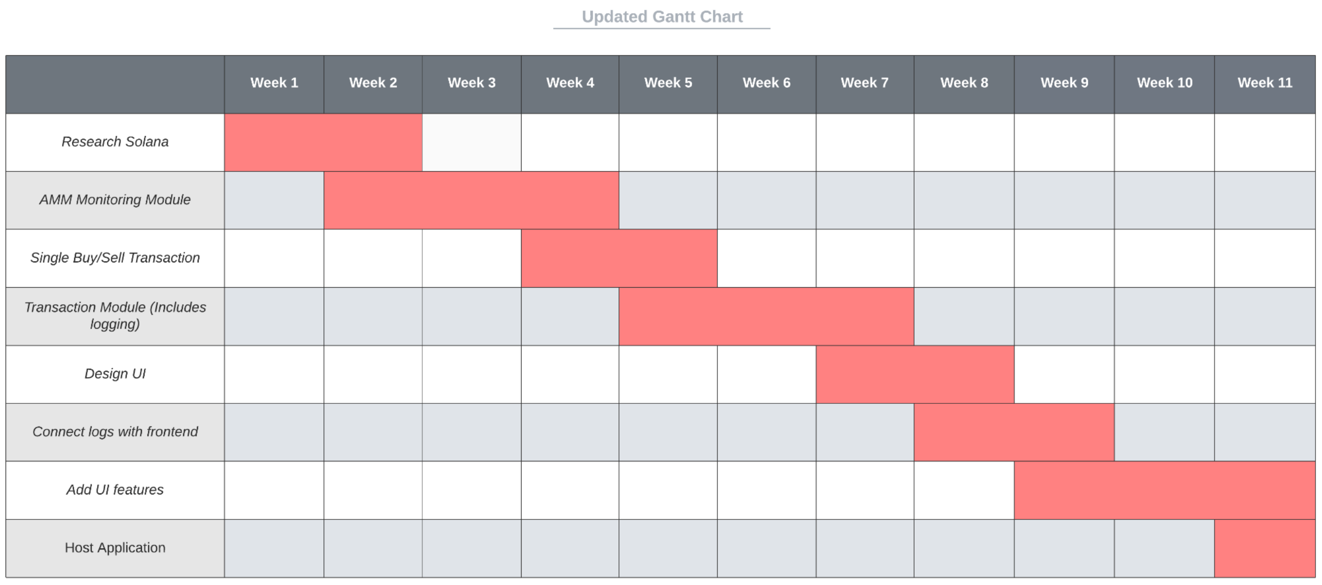
Each member also has their own individual logs to track their own progress for the course of the project.

3.1 Updated implementation schedule

Pert Chart


Research Solana
Design UI
Develop UI that displays log information
Feature to add money to wallet
Turn log information into graphical display
Feature to start/stop software
Host UI
Project Launch
Research how transactions work on chain
Set up paper wallet
Set up hello world program on Solana
Create single transaction (buy or sell)
Implement reversion of transaction
Create transaction with multiple jobs (buy&sell)
Send logged transcation to UI
Develop transcation based on monitoring system information
Find how to get information from on chain programs
Decide if monitoring system will be on chain
Retrieve information from amm
Retrieve exchange rate information from amm
Log comparison between two amms that profit
Send information to transaction system

Gantt Chart

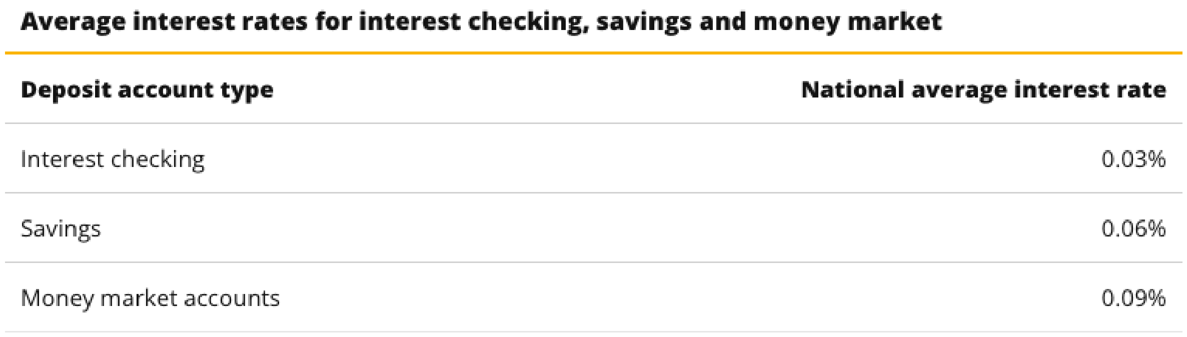


Updated Gantt Chart

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11
Research Solana	🟥	🟥									
AMM Monitoring Module		🟥	🟥	🟥							
Single Buy/Sell Transaction				🟥	🟥						
Transaction Module (Includes logging)					🟥	🟥	🟥				
Design UI							🟥	🟥			
Connect logs with frontend								🟥	🟥		
Add UI features									🟥	🟥	🟥
Host Application											🟥

3.2 Updated validation and testing procedures

With our project, it is quite simple to quantify how successful the project is. The overall goal of the project is to produce profit. The amount of money that the project produces will depend on the amount of money that it is given and it will vary greatly. This means that the way that we will have to quantify how successful the project will be should be percentage based. When it comes to getting users to put money into the project, we need to make sure that their money is better off in our arbitrage system, rather than just sitting in a bank account and accruing interest. The numbers that we have to compete with are as follows:



Average interest rates for interest checking, savings and money market

Deposit account type	National average interest rate
Interest checking	0.03%
Savings	0.06%
Money market accounts	0.09%

This is the goal that we have to aim for as we cannot expect users to place their hard earned money into our project if it is not even better off than just sitting in your bank account. We do want to produce more profit than these amounts but this is the baseline amount that we need to strive for in order to acquire users. A stretch goal will be to compete with some other long term investment strategies and their return rates, such as buying and holding certain cryptocurrencies, stocks or ETFs. This is a stretch goal and probably out of scope of this project and the time that we have allotted but that is the ultimate objective for any arbitrage system.

In order to demonstrate our project, we will show a video of the program running and making the transactions. This will show that the minimal viable product for this project has been met and the team is making strides towards the main objective. This video will show that the team has completed the first phase of the project, and the next steps will be to focus on testing and monitoring the transaction function in order to better focus on profitability. It will also show that the project is moving towards the command line interface type solutions that already exist and that we are now ready to begin designing and implementing a user interface that will solve a large part of the need statement.

A working prototype at this stage will be able to make transactions and monitor the prices on different AMMs. Once optimization and tuning come into play with the transaction function, the goal will be set at the average annual bank account interest rates that are found in the image above. Another key component to quantify the success of this project when it comes to meeting the need statement will be user interface. We need to ensure that the interface is simple and user friendly, unlike many of the existing arbitrage programs on the market today. This is a bit more tricky to quantify the success for and one of the ways we plan to do this is to interview users and ask for feedback. The team is aiming for ease of use while still providing all the necessary information that the user might want to know about how their money is being occupied. These are the questions that we will want to ask the test users and hopefully we can obtain users who will be using the product for an extended period of time and actually have money in the program, in order to get more accurate and representative feedback. When users have a position in a product or have spent money on something, they are much more likely to be honest about the performance and design of the product, as opposed to test users who are simply answering questions.

3.3 Updated division of labor and responsibilities

Anu:

- Finalize UI Features (Week of 3/21)
  - Talk to rest of team and determine what the user should be able to see and do on their end
  - Design each of the pages on Figma
- Add / Use Solana's Wallet Adapter (Week of 3/28)
  - Utilize their React UI to add to our application
  - Use a test wallet to see if it can connect, as well as send a transaction
  - Make sure it works with our transaction module
- Add Visualization of Price Monitoring (Week of 4/4)
  - Find appropriate charting, such as chart.js
  - Use data from price monitoring and convert to appropriate format if needed
  - Use something like chart.js to display prices
- Add Performance Metrics Visualization (Week of 4/11)
  - Choose the appropriate charts based on the metrics being shown
  - Use data that Robert has produced to show on the dashboard
- User Testing (Week of 4/18)
  - Get feedback from peers and improve on the usage flow / design

Veronica:

- Begin to shift Transaction module on chain (Week of 3/21 and 3/28)
  - Pass information needed for transactions from price module to on chain function
  - Use Solana on chain functions to successfully send transactions
  - Return information on transaction to price module
- Introduce atomicity to on chain transaction module (Week of 4/4)
  - Send all transactions together as one transaction
  - Confirm or reject transaction based on profit/loss
- Develop better logic for price module (Week of 4/4 and 4/11)
  - Try to make price module perform quicker
  - Improve logic on what to trade and how much to trade
- Introduce error checking (Week of 4/11)
  - Add error checking throughout pricing and transaction modules
- Merge modules with UI (Week 4/11 and 4/18)
  - Merge modules with UI to be able to control starting/stopping trading through UI
  - Start moving away from command line

Grant:

- Continue working with Transaction module (Week of 3/21)
  - Figure out what other exchanges we can potentially work with
  - Add the ability to trade other non-stable coins
  - Determine if Jupiter will allow us to perform swaps between multiple AMM's
- Help transition Transaction module on chain (Week of 3/28)
  - Assist Veronica with her work she has been doing to move transaction module on chain

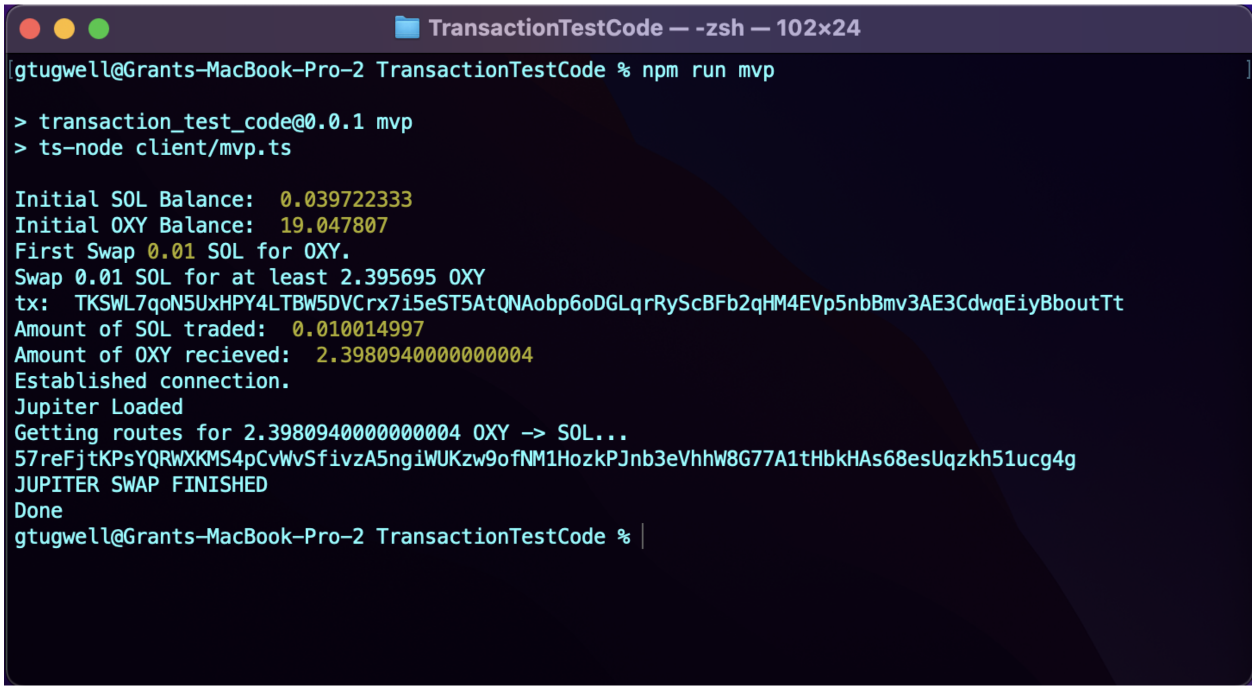
    - Look into best ways to connect our on chain transaction module with our price monitoring system
- Begin integration of on chain Transaction module and Pricing Module (Week of 4/4)
    - Verify on chain program works properly with our pricing module
    - Make sure information is current
- Explore atomicity and how to use it for Transactions module (Week of 4/4)
    - Work with Veronica to help get our onchain program working atomically
- Verify project is working as intended(Week of 4/11)
    - Continue working to make sure our program is behaving properly
    - Make sure transaction module is working and has necessary funds to stay on chain
- Working on integration with front end(Week of 4/18)
    - Make sure elements of the UI and elements of the code are working together properly
    - Work with Anu making sure our UI is usable

Robert:

- Define a structure for output of the transaction function (Week of 3/21)
    - Talk to Grant and Veronica about what the transaction function will return and what the monitoring system will need from the transition function to take as input
- Finalize Storage System and Define Schema (Week of 3/21)
    - Find out which method would work best for the use case of the team
    - Begin to define how the data will be stored within the data structure
- Connect to storage system and interact with data (Week of 3/21)
    - Add sample data to the storage system and ensure that all the connections are correct and that the data is stored correctly and is able to be accessed.
- Begin performing analysis on the data (Week of 3/28)
    - Come up with methods to quantifiably determine and visualize the performance of the arbitrage program
- Learn about different trading strategies (Week of 3/28)
    - Research what are some of the different routes that the team can take and how will we determine which works best for us
- Integrate with frontend (Week of 4/4)
    - Produce data that can be used graphically
    - Implement an alert or message system to the frontend
- Devise a testing plan(Week of 4/11)
    - Plan out which strategies will be tested and how much money they will be assigned for that stage of testing
- Set up testing environment (Week of 4/11)
    - Set up the computer that will run the program and make sure that it has all the necessary dependencies installed
- Tune arbitrage parameters (Week of 4/18)
    - Learn from the tests and try to obtain information that could lead to an improvement in strategy

    - Create the new strategy within the transaction function and then retest and repeat for the best results
- Error handling and extended runs (Week of 4/25)
    - Look into runtime errors that may be occurring and find the root cause
    - Test longer runs to ensure that there are no errors over longer periods of time
- Work on stretch goals or any work that did not go according to plan and may have been behind schedule (Remainder of the Semester)

4 Preliminary results



TransactionTestCode — -zsh — 102×24

```
gtugwell@Grants-MacBook-Pro-2 TransactionTestCode % npm run mvp

> transaction_test_code@0.0.1 mvp
> ts-node client/mvp.ts

Initial SOL Balance:  0.039722333
Initial OXY Balance:  19.047807
First Swap 0.01 SOL for OXY.
Swap 0.01 SOL for at least 2.395695 OXY
tx:  TKSWL7qoN5UxHPY4LTBW5DVCrx7i5eST5AtQNAobp6oDGLqrRyScBFb2qHM4EVp5nbBmv3AE3CdwqEiyBboutTt
Amount of SOL traded:  0.010014997
Amount of OXY recieved:  2.3980940000000004
Established connection.
Jupiter Loaded
Getting routes for 2.3980940000000004 OXY -> SOL...
57reFjtKPsYQRWXKMS4pCvWvSfivzA5ngiWUKzw9ofNM1HozkPJnb3eVhhW8G77A1tHbkHAs68esUqzkh51ucg4g
JUPITER SWAP FINISHED
Done
gtugwell@Grants-MacBook-Pro-2 TransactionTestCode %
```

At this time, the program is currently on the command line. Currently, profits are not being made with every transaction; however, we are able to complete transactions at this point. Once the monitoring and execution of transactions are more refined and completed, we will move forward with creating and implementing the user interface for a better user experience. Our main focus when developing our MVP was getting our transactions to work. Now that we have working code that allows us to perform transactions between various tokens on different exchanges, we are excited to continue moving forward.

Here you can watch a youtube video we recorded that shows our project working: https://youtu.be/ZWq1n7FFtw0