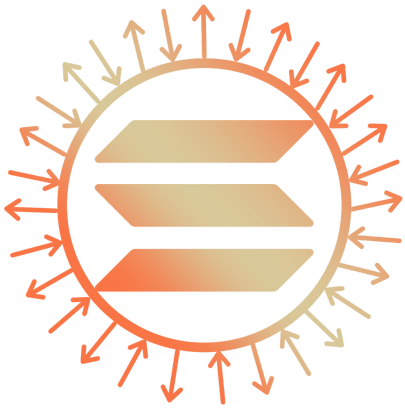
# Arbitrage on the Solana Blockchain



*Project Proposal*

Resol

Robert Elizondo

Anu Khatri

Grant Tugwell

Veronica Wisor

Department of Computer Science

Texas A&M University

February 7th, 2022

# Table of Contents

## 1 Executive summary

For this project, we will be creating an arbitrage system that runs on the Solana blockchain. This arbitrage system aims to create an effective and user-friendly product that will take advantage of the subtle changes in exchanges between the automated market makers on the Solana network and attempt to make a profit. This will include creating a system that will monitor exchanges between amm's on the Solana network, make profitable transactions, and have a user interface that allows the user to see everything that happens.

To create a successful arbitrage system, we designed four subsystems that will interact with each other: the automated market maker (amm) monitoring system, the transaction system, the logging system, and the user interface. The monitoring system will monitor and compare current exchanges on different amms. When a profitable comparison is found, the transaction system will be notified. The transaction system is responsible for making profitable transactions on the Solana blockchain. It will receive information about a potential transaction, attempt to make a transaction, and if the transaction would produce a profit, finish the transaction. This transaction system will always check to see if the transaction will produce a profit before finishing the transaction, ensuring that no money is lost. Any transactions that are made will be logged as part of the logging system. This system will allow for proper testing of the product. Finally, to make the product user friendly, all logs that are created will be displayed graphically in the user interface. This user interface will allow the user to interact with the system and commit actions such as starting or stopping the program.

Based on our research of other systems on the market similar to our proposed design, we expect our design to outperform the current options. This is due to our choice of using the Solana blockchain, and our decision to provide a user friendly interface. The Solana blockchain requires very little gas when making transactions, which means our system will be able to focus more on making a profit than worrying about deficits due to gas. This will allow our system to be more effective at making profits. Additionally, in comparison to other arbitrage systems, ours will provide a user interface that users can view their transaction data on, and interact with. Others on the market only offer command line interfaces which are not as accessible to all users.

With our well designed logging system and user interface, we expect that our project will be properly tested and be ready for launch without any major bugs. Additionally, with our in-depth list of tasks and schedule, we expect to deliver a fully functional arbitrage system by the end of April, eleven weeks from now. We have broken down the needs for the project into well defined tasks and a timeline for these tasks, which should keep us on schedule.

Overall, we have designed an effective arbitrage system that meets the needs of our users. In the following sections, we go into more detail about the research we have done, how our system is designed, and how we plan to follow through with our design.

## 2 Introduction

### 2.1 Needs statement

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.

### 2.2 Goal and objectives

The goal of our project is to create a profitable arbitrage program that will give the user useful and initiative insight into the performance of the program. If the user puts money into the program, we want them to be able to see how their money is moving around these DEXs and show all the relevant information regarding the transactions. The user would also be able to start and/or stop the trading at any time in an easy way.

One objective would be to create a price monitoring software system that will keep track of the prices of cryptocurrencies from different Decentralized Exchanges in order to find where profits can be made. In order to have accurate prices, the computer's connection to the internet will need to have low latency.

Another objective would be to create a function that will invoke the transaction on the Solana blockchain network that will perform a buy and a sell transaction simultaneously in order to receive a profit from the difference of the transactions. In order to successfully create a profit, the transactions must occur promptly and also have checks to make sure the transaction is still viable.

The last objective would be to log all the failed and successful transactions and to display the analytics onto a front end so that a user of the application does not have to use a command line and is able to view everything all at once.

A major operating condition that we have is that if the program were to be profitable, we would want it to be running 24/7. This means that the program would either have to be hosted on a server or have a dedicated local computer that will have the arbitrage always running, such as a Raspberry Pi.

### 2.3 Design constraints and feasibility

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.

## 3 Literature and technical survey

Our first tool that we reviewed was something called Bitsgap [3], which is a crypto currency trading bot that offers many different forms of automated trading. There are multiple pricing tiers that allow you to have different access to the trading bots, as well as the strategies available and the volume which you can trade. It has a user-friendly interface, with documentation about what the different strategies are that the bots use, as well as the different exchanges it can trade cryptocurrencies on. The main downside to this is that it is expensive, from the online reviews we have analyzed, many users ended up switching to different trading bot platforms because unless you are in the top tier, which costs $150 per month, you are extremely limited with what you can do. Additionally, these bots are not running on chain, they are making transactions through different crypto currency exchanges like Binance and Coinbase for example.

The second product we reviewed was called BlackbirdBot [1], which is a crypto currency arbitrage bot that only works with bitcoin. It is also an open source project that can be downloaded and run from github. It says that it is aimed at beginners; however, it does not have a ton of documentation. A major issue with BlackbirdBot is that its builds are reporting as failed.

The third product we looked at was called Cryptohopper (https://www.cryptohopper.com/), which is a cloud based crypto currency trading website that allows for many different trading options. They have things like Arbitrage and Automatic Trading bots, as well as strategies that you can "buy" which give you access to various resources that someone else set up for you, say for example bots with specific parameters. This solution does not seem as well flushed out as the Bitsgap product. They also offer different tiers a user could purchase, with the cheapest simply only allowing for manual trading. It seems like unless you buy the top tier for $99 per month, you would be extremely limited in your trading features. Their software works by interacting with different exchanges through an API, meaning there is room for improvement in terms of speed that prices become available as well as transaction speed.

Our next project that we explored was another open source project on github called Bitcoin arbitrage [10]. This is a project that gets arbitrage data from many popular exchanges, such as Gemini and Binance, however the automated trading is only supported between exchanges Bitstamp and Paymium. This project only allows for Bitcoin arbitrage with currency pairs of USD, EUR, and CNY.

The final product that we looked at was called Pionex [6] which is a cryptocurrency trading platform that offers 'Free' built in trading bots with only a 0.05% fee. The benefit to this product is that you can simply deposit USD into the Pionex account, and then the bot can begin making trades. There is no need to download any other software or have any hardware wallets.

When comparing these existing products to our proposed solution, there are a few key differences we plan to take advantage of. First of all, our product will not require that we have to buy any kind of tier to actually use our software, we will be able to write and run all of the software ourselves, so besides any of the costs associated with hosting and transaction cost, we should not have to pay for very much. Additionally, we are planning to have our transaction and monitoring software be running on chain, which provides advantages in terms of transaction speed and transaction cost. Utilizing Solana's extremely fast transaction speed, we should have clear advantages over services such as Cryptohopper which use APIs to interact with crypto exchanges. Additionally, while we have not narrowed down exactly which currency

pairs we plan to trade, we do plan to offer arbitrage abilities on more than just one currency pair, unlike some of the github projects we reviewed.

# 4 Proposed work

## 4.1 Evaluation of alternative solutions

When examining the alternative solutions, across different specific products, we noticed some categories these could be grouped into. First, there is software that simply provides a trade signal, and lets the user do all the actual trading. The benefit to a solution like this is that the user has all of the control, which allows them to ignore certain signals they believe to be too risky or potentially not worth the effort. The downsides of this method is that it is extremely slow, especially compared to alternative solutions. While a human may be able to make a trade in just a few clicks, this at a minimum takes a few seconds, which is extremely slow when you're competing against bots that are able to find an opportunity and complete the transaction seemingly instantly compared to user input. This slow speed also makes this method more risky, because once the user buys the crypto to sell, the slow speed could mean the opportunity to make a profit or to even break even is gone.

Another solution are trading bots, which not only find trade signals, but also make the trades. The benefit to something like this is that they are relatively fast when compared to a human who makes the trade like in the first solution. The downside of this is that the user has little to no control over which trades are made. Due to the speed that the opportunity is found, and acted upon, there is not enough time for the user to be able to override a trade. Additionally, many of these bots may be running locally or interfacing with exchanges through API's, or their own API's, which could be slow depending on the connection. Another downside is that these types of bots often cost money, or take some sort of percentage of profit when making transactions, which can cut down on the users profits.

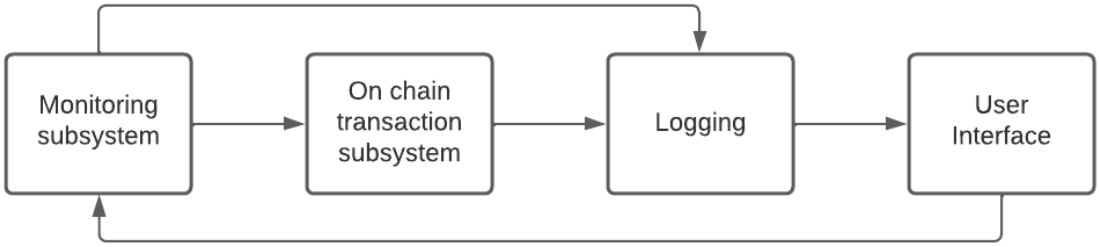
In addition to performing arbitrage on Solana's blockchain, there are other solutions that use other blockchains to arbitrage. In this case, the benefits are unknown, because it depends on which specific block chain is being used, however, some potential benefits would include things like network speed, which potentially could be faster. Some downsides would be that transaction costs would likely be higher on a different blockchain, because while there are blockchains with lower fees, a majority of the other blockchains have a higher transaction cost. Another potential downside to using a different blockchain, is environmental impact. Depending on which type of validation is used to confirm transactions, other blockchains that use systems like Proof of Work which are computationally much more expensive than Solana's Proof of Stake, use more energy. Another potential downside is transaction speed. Solana currently is one of the fastest blockchains when validating a transaction, and other general purpose blockchains like Etherium for example, are much slower to validate a transaction.

As far as trading goes, another solution we came across when examining how to make money, were different trading strategies. Some of the ones we came across were methods that simply said to buy and hold a certain crypto for a long period of time. The benefit to a strategy like this is that due to the long time horizon you are willing to hold that particular cryptocurrency for, you are willing to let the crypto price fluctuate, meaning you could get a higher return on your initial investment than if you were simply concerned about arbitrage. The downside to a strategy like this is that you could also potentially get a much lower return on your initial investment. Additionally, while having a long time horizon can be good for things like more stable assets such as stocks or real estate, crypto has been shown to be much more volatile, meaning that the owner of certain crypto assets is exposed to more risk.

Finally, we have our solution that we have decided to run with. Our idea is to have our monitoring and our trading performed on chain. The benefit to a solution like this is that because our monitoring code and our transactional code are on chain, we should be able to monitor and process our transactions as fast as possible. Additionally, because we are doing this on the Solana blockchain, our transactional fees will be relatively very low. The downside to our solution is that the user will have no control over the trading because it will be too fast for a human to override a transaction. Additionally, because all of the monitoring and trading will be performed on chain, the interface that the user sees, which will be a program that is either running locally or online, will be a slight delay of what is actually happening. However, this is an acceptable trade off because the user would not be doing anything besides monitoring what is happening.

### 4.2 Design specifications

Our arbitrage system will consist of four different modules: the monitoring module, transaction module, logging module, and the user interface module. The modules will all interact with each other in order to glue the system together.




**Monitoring system**

This system will monitor exchange rates of automated market makers on the Solana blockchain. Our current plan is to have this monitoring system monitor Raydium and Mango Markets for the SOL/USDC exchange rate. We would like to expand this to other amms and exchange rates, however due to the constraint on time this is a stretch goal.

The current design involves a loop that will loop until a user interface button is clicked to indicate a stop on the program.. During the loop, the monitoring system will check both amm's exchange rates for SOL/USDC and compare them. For each comparison made, this system will log in log files the comparisons being made. If a potential profit is found, this system will notify the transaction module to attempt to make a transaction. In addition to notifying the transaction system, the system will also log that it notifies the transaction system.

Our design for this system also incorporates the use of multithreading. During the loop, there will be at least two threads running and checking the exchange rates concurrently. The thread count will increase if we hit our stretch goal of expanding to other exchange rates or amms. Due to the quick changes that occur in exchange rates, this concurrency is necessary to allow for more accurate comparisons of exchange rates.

Finally, in our current design plan, we plan to have this monitoring system on the Solana blockchain along with the transaction system. This will allow us to easily notify the transaction system without needing to connect locally to an on chain application. We hope that this will create a faster application that can flag

profits and make successful transactions. However, there is also potential to make this module local and have it connect with on chain applications to get amm exchange rates and to communicate with the on chain transaction system.

**Transaction system**

This module will make transactions between two amms on the Solana blockchain. This module will take information from the monitoring module to attempt to make a transaction. Similar to the monitoring system, this system will be on chain and will make on chain transactions between two automated market makers. The transactions made in this module will only occur with SOL/USDC between Raydium and Mango Markets unless time permits to expand this.

The current design for this system is to receive information from the monitoring system. With this information, the system will make a transaction that will buy from the amm with the lower exchange rate of SOL/USDC and sell from the amm with the higher exchange rate. Then, before finishing the transaction, the system will check if a profit was made. If the buying and selling produced a profit, the system will complete the transaction and log the results. Otherwise, the system will cancel the transaction and log the failure. This check accounts for the possibility of the exchange rates having changed since the monitoring system noted them, and will ensure that only profits are made with transactions.

**Logging**

The logging system is the module that helps connect all other modules to the user interface. Once the on chain function is invoked and either completes or rejects the transaction, it will return a summary of the attempted transaction. This includes the metadata about the transaction, an example of what the on chain function would return is:

{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}

There could be more data points that we find will be useful while developing the project, but for now if the on chain function returns something similar to what is listed above, it will allow us to closely monitor the actions of the arbitrage. These summaries of completed transactions and failed transactions will allow the team to tune the arbitrage and try different strategies to find out which is best or if the most optimal performance comes from a combination of strategies. Logging this information will make testing and validation with the paper wallet possible and it is a pivotal step before adding real money to the arbitrage.

**User Interface**

Like the logging system, the user interface will allow the user to see the and analyze the performance of the arbitrage. Instead of the basic responses that will be returned from the on chain function, the user interface will be a web application that will be much more detailed and easier to interpret than a command line interface. The user interface will have useful graphs and charts that show important performance metrics from the arbitrage. This will also be very useful during testing because it will allow us to quickly see and visualize problems or inefficiencies within the program. 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. We anticipate the dashboard to look something like the following image:


## 4.3 Approach for design validation

To ensure proper design validation for our arbitrage system, each subsystem will log information as the system is running. In the final product, the user will be able to use the user interface to have a user-friendly way to view the logs. Additionally, in order to validate the system before launching the product, we will be using a paper wallet.

A paper wallet is a wallet that holds fake money. Having a paper wallet allows us to make transactions on chain without the risk of losing real money. It is necessary to have to provide a safe way to test our system, in the case of having a faulty monitoring or transaction system. This testing phase will be the most important, as it will allow us to try different financial strategies. Once we can prove that the arbitrage can be profitable over an extended period of time with a paper wallet, we will then add real money into the system and ensure that the system remains profitable.

The monitoring system will log metadata for all the comparisons that it makes between Raydium's and Mango Market's exchange rates. These logs will show us if the monitoring system is retrieving correct data from the on chain amm's. If the logged exchange rates are very different from the posted exchange rates on each amm's site, then we will know that the retrieval of information from the amms is not working properly. In addition, the monitoring system will log if it finds a potential area for profit within these comparisons. This will tell us if the comparison portion of the monitoring system is correctly identifying profits.

The transaction system will also log metadata for all transactions that it attempts to make. These logs will show us how many transactions are successful, and whether the transactions are making a profit. If the logs show that no transactions are being successfully made when there is a profit or that transactions are being made when there is a loss, then we know that there are errors within the system.

Finally, all the metadata that is logged in the monitoring system and transaction system will be displayed in the user interface. This user interface will take logs in real time and update graphical displays and information displayed. This real time updating will allow testing to continue to happen while the product is running, which will provide a comprehensive way to test the system under the conditions that it must perform.

These testing methods will help us ensure that the design fits the stated need because it will verify that the transactions made produce a profit, and that the user will have up to date information to view on the user interface.

# 5 Engineering standards

## 5.1 Project management

Veronica Wisor has experience with multi-threading, concurrency, C/C++, Python, Node.js, Django, and JavaScript. Robert Elizondo has experience with Python, cloud computing, and backend development. Grant Tugwell has experience with Python, C++, cloud computing, and API creation. Anu Khatri has experience with C++, Node.js, JavaScript, UI/UX design, and frontend development.

Anu will be the team leader due to her past experience in leadership and project management roles. She will create the weekly agendas and reports so that the professor and the TA are informed on a weekly basis. Additionally, she will coordinate the weekly meetings, as well as take minutes for each meeting so that the rest of the members can also look back at them in the future for reference. She will be doing technical reporting due to her experience through previous projects.

Anu will also be the lead in the creation of the user interface that customers will use to add money, track the markets, and start/stop the arbitrage. Veronica will also aid in the UI design due to her previous experience, as well.

The finance expert will be Grant since he is the most knowledgeable in cryptocurrency, due to previous experience in mining and trading cryptocurrency. He will be the point of contact if other team members may not be sure about a process related to cryptocurrency.

Robert will be the blockchain expert since he has the best grasp and understanding of the process, such as the interactions with Solana and setting up requests and transactions. He will head the on chain transaction module, which will execute the transactions once the prices have been flagged in the amm monitoring module.

The creation of the amm monitoring module will be led by Veronica since she has experience in multi-threading, which will be important when it comes to being able to track prices quickly and efficiently across different markets.

The testing will be done through Paper Wallet, which will be headed by Grant and Robert since they have worked with creating and using a paper wallet. This will also be testing if the goals have been met for the project.

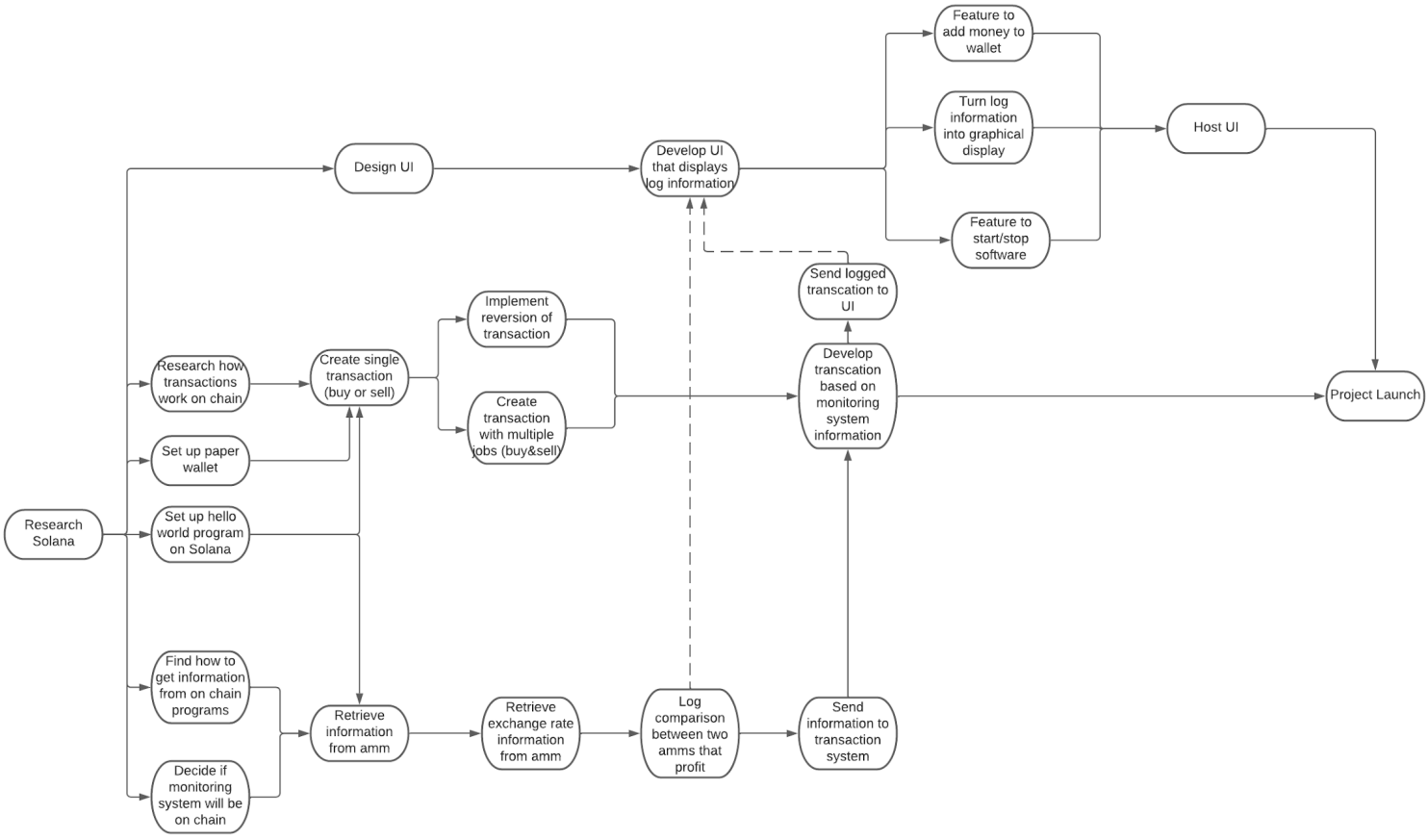
Hosting the application will be led by Robert due to experience with AWS. Grant and Robert will be in charge of creating an EC2 instance since they have experience in cloud computing from past classes.

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

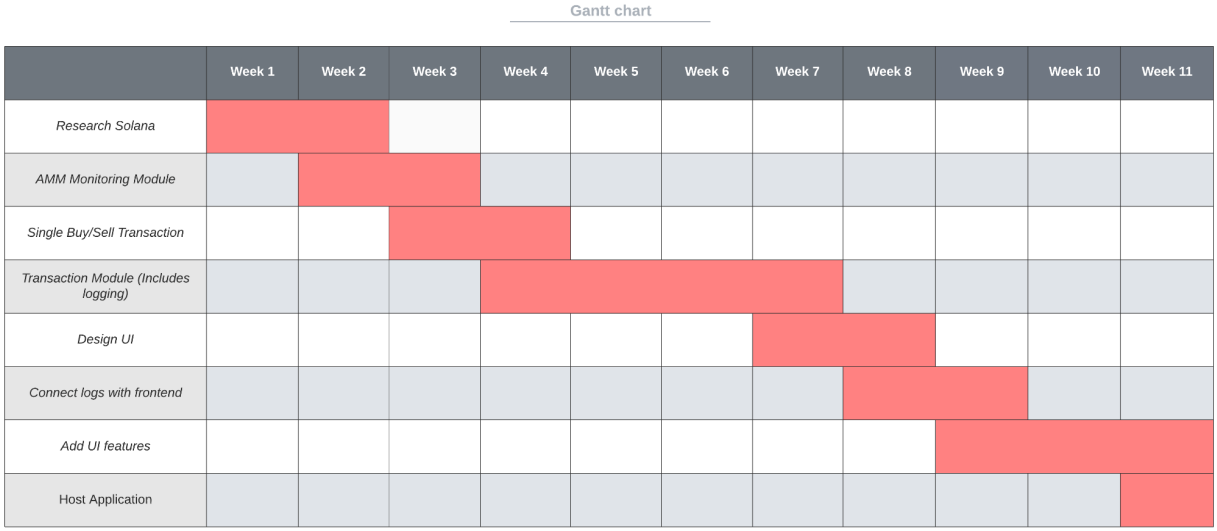
Some ways that the team and its progress will be managed will be through taking notes of all meetings, whether it be brainstorming sessions, standups, etc. These minutes will be added to the Git repository each time. Additionally, once development begins, a software development tool such as JIRA or Trello will be utilized to track the progress for each task, as well as create a backlog of tasks that will need to be completed. The tasks will also be broken down into more manageable tasks that can be completed in shorter lengths of time. The Gantt chart will be utilized, as well as updated throughout the project for a better understanding of the task completion and the overlap that may occur with other tasks. Each member will also have their own individual logs to track their work and progress for the course of the project. There will be progress checks each Monday and Wednesday.

## 5.2 Schedule of tasks, Pert and Gantt charts

**Pert Chart**



## Gantt Chart




### 5.3 Economic analysis

If this system were to become a commercial product, several things would need to be taken into account that would introduce economic issues. For this product to be used by multiple users, it would need to be hosted 24/7 on a service such as AWS. With more traffic due to it being a commercial product, a better server than currently needed would be required. Additionally, the logging system and UI that is included in our design is built to support a single user. To expand this product to the public would require a database to house all the data from the logging system. All user data that is displayed in the UI and is used by the monitoring and transaction systems would need to be kept on record. This could also bring in the need for security measures for such a database.

As with all software services, sustainability is very important. In order to ensure sustainability for a commercial product, we would need a team of software engineers that provide continuous tech support for users. Also, this team of software engineers would need to continue to update the service so that the service remains relevant and up to date with the current market.

### 5.4 Societal, safety and environmental analysis

For the early stages of the project, we do not plan on making the program public. The arbitrager will only be used by our team, with our own money. We want to do extensive research and testing before releasing it to the public, as it could have detrimental effects on someone's finances. As far as safety concerns go, we do not expect there to be any. Because our arbitrage is built on the Solana platform, the environmental footprint is kept very low. Solana uses proof of stake to validate new blocks and this process is much less energy intensive compared to proof of work. Other cryptocurrencies such as Ethereum use proof of work and this process of validation is much more computationally intensive and leaves a much larger environmental footprint compared to proof of stake.

### 5.5 Itemized budget

In order for the arbitrage program to be successful at its goal of turning a profit, it needs to have money to use. Once the program has been tested extensively, 10 dollars will be introduced to it. If we plan to host the arbitrage to be running 24/7, we would have to purchase an AWS server, such as an EC2 instance. EC2 has a free tier that uses the t2.micro server. This could be a good start to our project, but if we do need to go up from the free tier, we would also use the next step up, which would be the paid tier. The following image shows the pricing for some of the available instances on AWS:

| Instance name ▲ | On-Demand hourly rate ▽ | vCPU ▽ | Memory ▽ | Storage ▽ | Network performance ▽ |
|---|---|---|---|---|---|
| a1.medium | $0.0255 | 1 | 2 GiB | EBS Only | Up to 10 Gigabit |
| a1.large | $0.051 | 2 | 4 GiB | EBS Only | Up to 10 Gigabit |
| a1.xlarge | $0.102 | 4 | 8 GiB | EBS Only | Up to 10 Gigabit |
| a1.2xlarge | $0.204 | 8 | 16 GiB | EBS Only | Up to 10 Gigabit |
| a1.4xlarge | $0.408 | 16 | 32 GiB | EBS Only | Up to 10 Gigabit |
| a1.metal | $0.408 | 16 | 32 GiB | EBS Only | Up to 10 Gigabit |

All of these instances have 10 Gigabit network performance and since this program is network bound this could prove very useful. If this program were to be running 24/7 then the price ranges from $.612 per day for the a1.medium all the way up to $9.79 per day for the a1.metal. The instance that we choose when hosting the arbitrage will depend greatly on the amount of money it produces and the amount of performance that is required. Because pricing from market to market can change so rapidly, paying for the 10 Gigabit network performance could prove to be beneficial.

We could also purchase a dedicated computer such as a Raspberry Pi for about 40 dollars. The Raspberry Pi could be left to be running 24/7 and it would be a more simple solution to AWS. If we used the cheapest AWS instance, the a1.medium, for the remaining time we are in the class (11 weeks), it would cost about $47.00 (.0255*24(hours)*7(days)*11(weeks)). Another key part of the budget is the cost per transaction. Solana is very cost effective in this regard and each transaction has fees that are very small and it takes out a lot of the risk of doing many repeated transactions. The post per transaction on average today is $.00107. This means that 100 transactions would cost about 1 penny. This removes the need to be cautious of the cost of transaction fees being greater than the profit, like with other cryptocurrencies. There will be no parts needed for the project, nor fabrication. The development process will not cost the team anything. It is also possible to lose money from the arbitrage program or gain money to be profitable. Our team will be receiving $600 from the University for the development and completion of the project. We will try to keep other costs down so that the majority of the money goes into the arbitrage itself.

## 6 References


[1] A. Millar, “Blackbird: An open source arbitrage bot,” *Medium*, 17-Mar-2017. [Online]. Available: https://medium.com/@TheAlexGalaxy/blackbird-an-open-source-arbitrage-bot-6e457f7ac63c. [Accessed: 13-Feb-2022].

[2] B. G. Malkiel, *A random walk down wall street: The time-tested strategy for successful investing*. New York, New York: W.W. Norton and Company, 2016.

[3] Bitsgap, “Bitsgap,” *Crypto Trading Bot - Fully Automated & Free Testing Available*. [Online]. Available: https://bitsgap.com/crypto-trading-bot/. [Accessed: 13-Feb-2022].

[4] Bitsgap, “Cryptocurrency arbitrage explained,” *Bitsgap blog*, 06-Jul-2020. [Online]. Available: https://bitsgap.com/blog/crypto-arbitrage-explained-tutorial/. [Accessed: 13-Feb-2022].

[5] “Blockchain Consensus Encyclopedia,” *README - consensus*. [Online]. Available: https://tokens-economy.gitbook.io/consensus. [Accessed: 13-Feb-2022].

[6] “Crypto trading robot: Free crypto trading bot,” *Pionex*. [Online]. Available: https://www.pionex.com/en-US/. [Accessed: 13-Feb-2022].

[7] “Developing with C: Solana docs,” *Solana Docs Blog RSS*. [Online]. Available: https://docs.solana.com/developing/on-chain-programs/developing-c. [Accessed: 13-Feb-2022].

[8] E. P. Chan, *Algorithmic trading: Winning strategies and their rationale*. Hoboken, NJ: Wiley, 2013.

[9] Maxme, “Maxme/bitcoin-arbitrage: Bitcoin arbitrage - opportunity detector,” *GitHub*. [Online]. Available: https://github.com/maxme/bitcoin-arbitrage. [Accessed: 13-Feb-2022].

[10] “The most powerful crypto trading bot,” *Cryptohopper*. [Online]. Available: https://www.cryptohopper.com/. [Accessed: 13-Feb-2022].

[11] Torsten Hartmann January 5, “What are the best bitcoin arbitrage bots in 2022?,” *CaptainAltcoin*, 05-Jan-2022. [Online]. Available: https://captainaltcoin.com/cryptocurrency-arbitrage-bots/. [Accessed: 13-Feb-2022].

[12] “🦾 Hummingbot review – an open source approach to trading bots,” *BitCourier*. [Online]. Available: https://bitcourier.co.uk/blog/hummingbot#:~:text=Hummingbot%20is%20an%20open%2Dsource,strategies%20independently%20from%20each%20other.d. [Accessed: 13-Feb-2022].

# 7 Appendices

## 7.1 Bios and CVs

Anu Khatri is a senior undergraduate student at Texas A&M pursuing a Bachelor of Science in Computer Science with a minor in Art. Although she has experience in various areas of computer science, she is most knowledgeable in front-end development and UI/UX design. She has completed courses, including Programming Studio, Software Engineering, Graphic Design I/II, and Designing for the Web. She has had internships at H-E-B Digital and J.P. Morgan Chase & Co., where she did UI work with Figma and front-end development with React and JavaScript.

Robert Elizondo is a senior undergraduate at Texas A&M University pursuing a BS in Computer Science. He is mainly interested in backend development and cloud computing. He used AWS during his internship at PACCAR in the summer of 2021 and has experience in the automotive industry. He is also interested in customer facing software development and consulting. After graduation, he will work for Salesforce as a Technical Consultant in Dallas, TX.

Grant Tugwell is a senior undergraduate at Texas A&M pursuing a BS in Computer Science. He mainly has experience creating, testing, and supporting API's, as well as working with Cloud Services such as AWS. He has interests in blockchain and machine learning, and hopes to one day be able to use his skills in the financial services industry or blockchain industry. After graduating he will be moving to Dallas and working as a Technical Consultant.

Veronica Wisor is a senior undergraduate student at Texas A&M pursuing a BS Computer Science with a minor in Statistics. She has experience in several different areas of computer science but specializes in low-level programming. She has completed many low-level courses at Texas A&M including Operating Systems, Parallel Computing, and Compiler Design. She also has experience with frontend UI design. In one of her courses, she won the best website design of the class. Finally, she had an internship with Advanced Micro Devices, where she worked as a CPU Post Silicon Verification Engineer. This involved writing diagnostics for Server CPUs and using C, C++, Assembly, Python, and Ruby.

# ANU KHATRI

Plano, TX | 972-890-4159 | anukhatri41@gmail.com


## EDUCATION

**Texas A&M University,** College Station, TX **May 2022**
Bachelor of Science - Computer Science, Minor in Art (New Media); GPR: 3.86
*Key Coursework:* Programming Studio; Graphic Design; Discrete Structures; Data Structures & Algorithms

## TECHNICAL SKILLS

**Frameworks/Libraries:** React.js; Node.js; Jest **Languages:** HTML; CSS; JavaScript; C++; Python; Java; GraphQL
**Tools:** Figma; Git; JIRA; Visual Studio Code; Adobe Creative Cloud (Illustrator, InDesign, XD, After Effects, Photoshop)

## WORK EXPERIENCE

**J.P. Morgan Chase & Co.,** Plano, TX **June 2021 – August 2021**
*Software Engineer Program Intern*

- Utilized Figma, React, & REST API to design and implement an updated UI and migrated a multi-page user experience into a single page application to align a legacy project with current company design standards
- Collaborated with interns, stakeholders, and end users to create a robust internal application

**H-E-B Digital,** Austin, TX **June 2020 – July 2020**
*Digital Customer Experience (CX) Intern*

- Updated UI for web timeslot booking modal & populated data through utilization of React, GraphQL, and Apollo
- Gained experience in working with a large codebase in a team software development cycle & mobile-first approach
- Engaged in seminars to foster deeper understanding of best practices for development and user security
- Utilized JIRA to track user stories and fix issues in the backlog

**Insubuy,** Plano, TX **May 2019 – July 2019**
*Software Engineer Intern*

- Championed efforts to migrate legacy code to WordPress formatting
- Converted UI/UX designs into code using HTML5 and CSS3 without use of a CSS framework
- Delivered comprehensive updates to more than 300 webpages to create clarity and optimized design
- Handled real-time production errors reported through Sentry

## ACADEMIC PROJECTS

**Happy Hikers Brand Book – Created for Graphic Design Course** **December 2021**

- Utilized InDesign, Illustrator, and Photoshop to design, develop, and create a brand book for Graphic Design class, including primary and secondary logos, typography, color palette, business cards, packaging, and box design

**Road Trip Planner – Application Created for Programming Studio Course** **December 2020**

- Spearheaded efforts to design, develop, and deploy a road trip planning application using React, Bootstrap, & APIs (Spotify, Google Maps, Yelp) while collaborating with 3 team members over 2 months using Agile methodology

**Plane Sleep - Website submitted for TAMUhack 2020** **January 2020**

- Designed and developed an interactive website created to comfort those who do not travel often via airplane
- Employed HTML and CSS and hosted on Firebase in less than 10 hours

**Website for Hindu Students Association – Texas A&M Chapter** **July 2019**

- Pioneered and deployed a website to promote the organization, which led to a 350% increase in membership
- Increased engagement, awareness, and participation in events through better communication

## LEADERSHIP EXPERIENCES

**Texas A&M University – Howdy Crew** **August 2020 – Present**
*Campus Tour Guide*

- Provide walking tours of campus to thousands of prospective students and their families while sharing my unique perspective on the university and the Aggie community

**Hindu Students Association – Texas A&M Chapter** **September 2018 – Present**
*President, Historian* *May 2021- Present, May 2019 – May 2021*

- Lead a team in order to plan and execute large-scale events for 300+ attendees, as well as meetings and socials for 135 general members, in order to preserve the knowledge of Hinduism & educate others in the community

# Grant R. Tugwell

gtugwell@tamu.edu | 832-495-9965 | www.gtugwell.com

## Education

Texas A&M University, College Station, TX — Expected Graduation Date: May 2022
Bachelor of Science in Computer Science
Cumulative GPA: 3.65

## Work Experience

**ProFrac, Willow Park, TX (Remote)** — **June 2021 – August 2021**
*Computer Science Intern*

- Continued developing an API which will be used to aid and coordinate employees in the field
  - Created functions to output JSON responses
  - Optimized API routing to cut down on maintenance
  - Helped debug issues found in production application
- Assisted in writing automated tests to be used in our development pipeline

**ProFrac, Willow Park, TX (Remote)** — **June 2020 – October 2020**
*Computer Science Intern*

- Helped develop an API which will be used to aid and coordinate employees in the field
  - Assisted in writing postgreSQL queries
  - Wrote code for numerous API routes/endpoints
  - Assisted in testing API routes/endpoints and implementing corrections
- Added features to working internal Power Apps

## Personal Projects

**Women In Nuclear Engineering** — **January 2021 – May 2021**
*Developer – Software Engineering Course*

- Created a website to organize their members points, as well as display organization information
- Allows officers to post relevant information in forums we created, along with an updateable calendar
- Automatically formats user information so officers do not have to manually enter event attendance data

**Accident Info Visualization** — **September 2019**
*Independent Project – Hackathon Winner*

- Created a website during the 2020 TAMUhack that allowed for the user to input a city in Texas, and see map of auto accident locations from 2016-2019

## Skills

- C++, Python, Java, JavaScript, TypeScript, ruby, postgreSQL, Haskell, HTML/CSS, Microsoft Power Apps, Git
- Microsoft office suite programs, AutoCAD: Inventor, 3D Printing

## Awards & Involvement

- Member of Aggie Coding Club
- Member of Phi Eta Sigma Honor Society

## Interests

- Running – Coordinates a small workout group, responsible for leading exercises
- Singing – Practice music regularly with my roommates who can play various instruments

# Robert Elizondo

(214) 724-6850
robmelizondo@gmail.com
Dallas, TX
linkedin.com/in/robert-elizondo

## Education

**Texas A&M University**
GPA: 3.70
B.S. in Computer Science
Expected Graduation: *May 2022*

**Dallas College**
GPA: 3.71
A.S. of Science
Graduated: *Fall 2020*

## Work Experience

**PACCAR** *June 2021- August 2021*
*Software Engineering Intern; Seattle, WA*

-Architected an internal tool to increase productivity and maximize time spent in testing labs.
-Created a full stack web application in order to develop products faster and with more accuracy.

**Automotive Programmer** *June 2017- Present*
*Self Employed Programmer; Dallas, TX*

-Generate custom tune files to maximize performance of powertrain control modules.

**Chacho's Auto Electric** *May 2015- Present*
*Shop Manager and Technician; Dallas, TX*

-Diagnose vehicles using modern scan tool interfaces.
-Negotiate sales and repair rates.
-Repair and program electronic control modules.

**The Coding School** *July 2020- September 2020*
*Volunteer Programming Instructor; Remote*

-Tutored student one-on-one to strengthen programming knowledge with Python.

## Fluent Languages

**English**
-Reading, Writing, Speaking.
**Spanish**
-Reading, Writing, Speaking.

## Certification

**AWS Certified Cloud Practitioner**
Issued: August 2021

## Technical Skills

-AWS
-Python
-ReactJS
-C++
-HTML/CSS/JavaScript
-Swift
-Java
-Assembly

## Projects

**By The Cover** *Fall 2020*

-Web application that recommends movies to users based on visual and audio cues.
-Recommendations are made using the movie poster and the soundtrack from the Spotify API.
-The recommender system uses cosine similarity and K-nearest neighbors.

**PolitiStats** *Fall 2020*

-Web application that gives sentiment analysis score to the tweets and news about a politician.
-Developed using Django and a number of API's as a group, using the agile development method.

**Yelp DB Project** *Fall 2020*

-Stores and filters Yelp data in an AWS database.
-Allows users to search for specific items using a Java GUI that queries the table using PSQL.

**DLR** *Spring 2020*

-iOS betting application that periodically draws a set amount of money into a user shared pool.
-The amount, frequency, and probability of busting are customizable.
-Developed using Swift and JavaScript at Howdy Hack as a group of two.

**Personal Website**

-robelizondo.com

# VERONICA **WISOR**

945 Lake Island Dr, Canyon Lake, TX · Cell: (512) 970 - 8902
**vwisor@tamu.edu · www.linkedin.com/in/veronica-wisor-2022**

I am seeking a professional position as a software engineer utilizing my relevant knowledge, technical experience, and problem-solving skills. I am driven and eager to learn more and to continue developing my skills. I desire a position that involves a team-based environment with a lot of opportunities to learn.

## EXPERIENCE

**MAY 2021 – DECEMBER 2021**

### CPU POST SILICON VERIFICATION ENGINEER CO OP, AMD

- Developed two command line options for SLT diagnostic to add duty-cycle features
- Collected data on silicon with SLT diagnostics to continue to improve diagnostics for quicker times to failure
- Assisted team in targeting a quicker failure time on bug escapes
- Assisted team in N-1 Bring Up Testing
- Developed and generated randomized assembly test cases
- Added support to CPUID diagnostic for new project
- Developed communication skills through collaborating with coworkers and across teams
- Presented technical projects to AMD directors and other co ops

## EDUCATION

**AUGUST 2018 – MAY 2022**

### DEGREE: BS COMPUTER SCIENCE WITH STATISTICS MINOR, TEXAS A&M UNIVERSITY

- Current GPA: 3.589, senior classification and awarded Aggie Ring in Spring 2020
- Previous classes include:
  - Data Structures and Algorithms (C++), Computer Organization (HDL/Assembly), Intro to Computer Systems (C/C++/Linux), Design and Analysis of Algorithms, Compiler Design (C++/Lex/YACC), Operating Systems (C/C++/Assembly), Parallel Computing (C++/OpenMP/MPI/CUDA), and Database Systems (SQL)
- Major specific classes for Spring 2022 include:
  - Machine Learning and Distributed Systems

## SKILLS

- Experienced in team environments
- Experienced in the Waterfall Method and Agile for project management
- Programming Languages:
  - Strong Understanding: C++, Python, Java, Ruby, R Studio, Excel, SQL, Lex, YACC
  - Good Understanding: Django Web Framework, Scheme, HDL, x86 Assembly Language, OpenMP, MPI, CUDA, CSS, HTML