

Solana Code Analysis as a Service



Project Proposal

Rust Cleaners

Arsh Kabarwal

Joseph Carpman

Kipp Corman

Olivia Ornelas

Department of Computer Science

Texas A&M University

Feb 14, 2022

Table of Contents

1	Executive summary (1 page; 5 points)	3
2	Introduction (1 page; 20 points)	4
2.1	Needs statement (5 points)	5
2.2	Goal and objectives (10 points)	5
2.3	Design constraints and feasibility (5 points)	5
3	Literature and technical survey (1-2 pages; 10 points)	6-7
4	Proposed work (35 pts)	8
4.1	Evaluation of alternative solutions (1 page, 10 points)	8
4.2	Design specifications (3-4 pages, 20 points)	9-10
4.3	Approach for design validation (1 page, 5 points)	11
5	Engineering standards (25 points)	12
5.1	Project management (1 page, 10 pts)	12
5.2	Schedule of tasks, Pert and Gantt charts (1 page, 5 pts)	13
5.3	Economic analysis (1/2 page; 3 pts)	14
5.4	Societal, safety and environmental analysis (1/2 page; 2 points)	14
5.5	Itemized budget (1 page; 5 pts)	15
6	References (4 points)	15
7	Appendices (1 point)	16
7.2	Bios and CVs	16

1 Executive summary

As blockchain has gained popularity, more people have become interested in developing blockchain smart contracts. Unfortunately, there is a limited amount of tools available for reviewing blockchain smart contracts. Reviewing smart contract code before it gets submitted is important because it is very difficult to modify or update smart contracts once they're deployed onto the blockchain. Currently there are code review tools specifically designed for blockchain analysis, but most of them require the user to pay for the service or download the tool. Most of the online tools require the user to send the code to the developers before receiving a cost estimate, which slows down the development process. Downloadable code review tools must be configured and the syntax must be learned before use, increasing the difficulty of blockchain deployment.

Because of this, the project aims to create a Software as a Service code review tool tailor made for blockchain developers. This tool will be unique from the current solutions by reviewing a user's code with multiple open source tools. This will let users see what percent of code analysis tools pick up on each vulnerability in their code, and ensure that nothing is missed. For code submission the project will allow users to submit a Github repo or upload files. If they choose to submit through Github, there will also be an option to automatically re-run the code analysis each time changes are made to the repo. Once analysis is done, the application will report the results of the code review in an easy to understand format. This means visually showing which lines of code caused each vulnerability to trigger and how many different tools picked it up. There would also be a summary report to allow developers to easily see how many issues there are in larger codebases. Objectives for this project include reducing the cost of using a blockchain code review tool, reviewing the code with multiple open source tools, and preserving the history of the code being reviewed within the tool. The design for this project will be a web application with a dashboard that gives the user the option to login, upload code, or submit a github repo URL. Once code is uploaded to the dashboard, the tool will run multiple code review tools and scanners on that code and output the results. If the user decides to login, the user will have access to the history of what code the user has previously uploaded and reviewed on the dashboard. Since this tool is meant to be used for blockchain development, a majority of the tools are meant for the Rust programming language. This implementation is the most logical because it will allow the user to easily see results from the code review in an easy to access way.

The expected results from this project include creating a tool that is useful and available for the public to use. Another expectation is that the project will create a tool that is easy to use and is accurate as compared to other code review tools currently available. The expected benefits of this project would be lowering the barrier-to-entry of blockchain development and helping developers confidently create secure decentralized applications on the blockchain. Lowering the barrier-to-entry would be achieved by bringing a product to market that was cheaper, simpler, and faster than current solutions. On-demand cloud code review allows the project to only pay for the computational resources that it requires. Automatic code review and a simple payment plan removes the need for quotes and back and forths between ourselves and the user.

2 Introduction

This project will be a software as a service that will host open source static code analysis tools for code being deployed onto the blockchain. A blockchain is a decentralized distributed ledger that makes the recorded information very difficult to change. Blockchain technology helps with the verification, security, and transparency of transactions and purchases on a business or community network. Since code that is submitted to the blockchain can not be amended or changed, it is important that the code is secure so malicious attackers can't take advantage of the flaws in the unsecure code. The code submitted to the blockchain is implemented in smart contracts which are self executing computer programs.

With the emergence of smart contracts and programs on the blockchain, there is a learning curve for creating secure contracts and programs. It can be difficult to understand and comprehend all the security measures that need to be present in the programs. Some of the pitfalls in smart contracts include missing ownership checks, incorrect exception handling, invalid permission checks, and account validation confusions. Our service will help educate developers with many of these pitfalls by evaluating the numerous pitfalls for the user of the service.

The Solana blockchain was specifically chosen for this project due to its faster transaction speeds and high level of scalability. Solana is a rapidly growing blockchain that supports builders creating decentralized applications and non-fungible tokens. A tradeoff for these advantages of Solana is an abundance of security issues. Solana Code Analysis as a Service will help offset this disadvantage by allowing users to minimize potential security risks using static analysis tools. This service will utilize Rust static analysis tools because Solana primarily uses Rust to create programs on its blockchains. The creators of Solana chose to use Rust in order to allow experienced developers to easily create code that aids with memory errors and concurrent programming. Solana also uses C++ and C for the creation of smart contracts. Static analysis tools for these programming languages will also be added if there is extra time available.

The service developed during this project will allow users to submit their code and then observe vulnerabilities, security issues, and performance issues in their code. The analysis results will be displayed onto a dashboard that enables the user to easily distinguish the performance of the code. The service will also create a formal report which will establish the legitimacy of the submitted code. This will allow for developers to be confident that their smart contract is stable and secure. The service will inform the user of the various static analysis tools that were used to judge the performance and security of the code. If multiple tools pick up on the same bug, this means that this bug has a high precedence. The user will also be able to compare their code with other versions of their code to see how Solana Code Analysis as a Service has improved the quality of their code.

2.1 Needs statement

The current barrier to entry of blockchain smart contract development is too high. The difficulty of modifying or updating deployed smart contracts demands robust code review tools to catch programming errors before deployment. Currently, code review tools that analyze code meant for blockchain requires the user to download the tool. Additionally, most tools require the user to send portions of their code directly to the company in order to get a quote for how much the tool will cost. Our project aims to create a low cost tool that doesn't require the user to download anything in the form of Software-as-a-Service.

2.2 Goal and objectives

The goal of this project is to implement a Software-as-a-Service tool which allows users to quickly and clearly review their code before it is deployed to the blockchain. This tool should be as user friendly as possible to lower the very high barrier of entry to smart-contract deployment. The minimum viable product would allow users to submit and run their code, and then visually show what lines have potential security/performance issues. In addition to this, the product will review a user's code using multiple tools and present the results in an easy to understand way. Some additional features we plan to add are creating a dashboard to summarize their results, user accounts and an area to view history and progress over time.

Objectives for this project include the ability to reduce the cost of using a blockchain code review tool by 20% and identify at least as many vulnerabilities in their code as the existing products do. Additional objective include:

- The product will be able to review code using at least 3 code review tools and produce results from each on the same dashboard.
- The product will be able to accept code from at least file upload and github url.
- The product will preserve up to 10 prior submissions for comparison.
- The product will preserve code review report data for historical modeling.

2.3 Design constraints and feasibility

One constraint in our project is that we are limited to using open source static analysis tools that already exist. We do not have the time or expertise to create our own, which means we are relying on others. For our objective of showing as many problems in the submitted code as existing tools, this entirely depends on how effective the tools we end up finding are. Another constraint will be a time constraint. This project has to be completed within about 3.5 months which can cause a lot of features and functionalities to be left out or uncompleted. Finally, we are potentially financially constrained. If this product ends up gathering a lot of users over time, there would be additional costs to start up more servers so that the users don't have extremely slow responses. All of their past entries would still need to be stored, and over time with people possibly submitting entire codebases this could be quite large too.

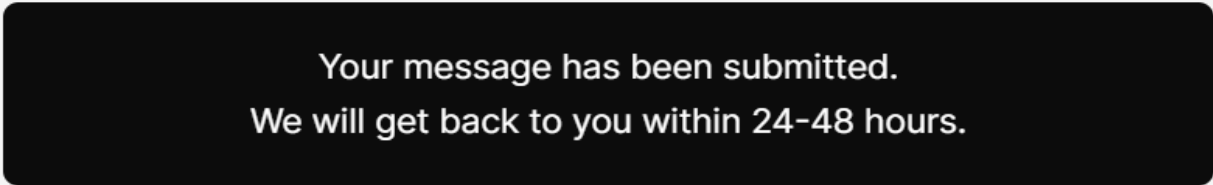
3 Literature and technical survey

- Blockchain Science Code Review

The Blockchain Science Code Review[1] is an automated tool for the static analysis of the source code which provides a detailed security code analysis. This tool supports many coding languages except Rust. The Solana Code Analysis as a Service will focus primarily on dealing with Rust. In addition, the Blockchain Science Code Review requires a developer to contact the owners of the tool in order to utilize the tool. The Solana Code Analysis as a Service code review will allow a user to submit their code and immediately receive results, making the process faster and smoother. The Blockchain Science Code Review is geared towards businesses rather than to regular developers or coders which Solana Code Analysis as a Service will target.

- Soteria

Soteria[2] is a security services and audit tool specifically for Solana Programs. Soteria can be used to detect vulnerabilities and tell the user why a specific command was flagged or where the code is vulnerable. Solana Code Analysis as a Service will be faster than Soteria. Whenever source code is submitted to the Soteria application, the earliest time range the results will take is 24-48 hours even for the simplest code. Solana Code Analysis as a Service will have a GitHub auto-update functionality that Soteria doesn't. This will allow the user to easier see how the tools are affecting their code.



Your message has been submitted.
We will get back to you within 24-48 hours.

Image of Soteria's message after code is submitted

- Anchain.AI

Anchain.AI[3] is an AI-Powered Blockchain Intelligence Platform for security, risk, and compliance. This service requires a user to request a demo in order to use the services. This service would be useful if needed for a large scale project that needs an immense amount of inspection and auditing of the safety of the smart contracts. Solana Code Analysis as a Service is more feasible for the average developer in getting quick and reliable analysis. The Solana Service will skip hurdles that large scale development projects can't.

- SonarQube

SonarQube[4] is a tool for inspecting code security and code quality with the use of code reviews. SonarQube utilizes clear visuals and dashboards that display vulnerabilities and bugs. Rust is not one of the 29 languages that this service evaluates. Solana Code Analysis as a Service will use similar visual elements to SonarQube, but it will be different because it is specifically used for Rust and will be for code that is deployed to the blockchain. SonarQube has a Community version that is free, but this feature can be difficult to users with novice experience. Solana Code Analysis as a Service will be very user-friendly and super easy to use.



Image of SonarQube's Code Analysis Dashboard

- DeepSource-
 DeepSource[5] is a fast and reliable static analysis platform that allows for self-hosted development. DeepSource analyzes every pull-request made in the user's provided repository. It generates solutions for the bug fixes automatically. The service works with all of the languages used for Solana. It is also free for personal accounts and small teams, but it costs money for any teams larger than three. Solana Code Analysis as a Service allows for easier flexibility on how users want to use the static analysis tools. Since Solana Code Analysis as a Service uses many static analysis tools, there is a larger chance that an error is detected than just one tool which DeepSource uses. The Solana Code Analysis as a Service also allows code to be submitted directly without a needed repository on GitHub, GitLab, or Bitbucket.

4 Proposed work

4.1 Evaluation of alternative solutions

- Alternative solution 1: Create a dashboard that allows users to login, upload code, or submit a github repo URL to be reviewed. On this platform, the tool will use different types of open source code review tools (primarily for Rust) to analyze the code and show where vulnerabilities or bugs are within the code and suggestions on how to fix them. In addition to this, the tool would also be able to keep track of that file or repo's review history since the user logs in. A downside of this solution is that it requires the most user data to be stored within the project's data servers. As a result, the tool will have higher security needs and an increased negative environmental impact. Despite this, it is the best solution because it would allow the user to easily access their previous versions of reviewed code, has easier access to Github repos, and aims to make the code review process more personalized than the other solutions.
- Alternative solution 2: Allow users to submit code in any language that is popular on the blockchain. This would cause our product to potentially gather a larger user base, but require more work before it would be up to the same standard. Since there are so many existing solutions that do static code analysis, it is important that our project is able to do a good enough job that someone would want to use it over those alternatives. Focusing on just 1 (and maybe 2) languages that are used in Solana lets our project focus on using many analysis tools to create a quality product.
- Alternative solution 3: Removing the login feature which wouldn't allow the web application to store the user's upload history and prevents the tool from connecting to GitHub efficiently. Doing this would be beneficial because users wouldn't save time not having to login and their data wouldn't have to be stored on the tool's servers. This can save money, complexity, and lower the environmental footprint of the tool. Conversely, removing the login feature would make connecting to Github more complicated, requiring users to constantly re-login, and wouldn't allow the tool to effectively store the user's previous history using the tool on different versions of the code submitted.
- Alternative solution 4: Create a new software code analysis tool instead of using open source code analysis tools online. For this project, using already created tools made more sense because of the time constraints and to help differentiate from other existing products that we analyzed above. All of their products implemented their own static code analysis rather than automatically running lots of existing open source tools. This solution also allows our service to show what percentage of open source tools found certain vulnerabilities to help determine how accurate it is.
- Alternative solution 5: Implement dynamic analysis. This will allow the user to see how their smart contract is performing while being executed. Dynamic analysis can provide significantly more feedback on the security and execution of the application on the blockchain. A con of this solution is that it can be difficult to trace vulnerabilities in a runtime solution. Many of these vulnerabilities will be very difficult to be changed because code on the blockchain is immutable. It would be more costly to fix the errors than just simply creating a new amended smart contract.

4.2 Design specifications

Figure 1 is the initial user flow diagram for the Solana Code Analysis as a Service project. The first component of the diagram is the user login. The user simply logs into our website in order to use our service. The interface will allow for the user to either upload a Github URL or upload a local file on the user's system. These options give the user easier accessibility to upload their code. With the authorization of credentials, the user can choose to have their code on the GitHub be tracked and kept progress with the static analysis tools. Whenever the user pushes code to the repository, the Solana Code Analysis as a Service will send a notification to the user with an updated dashboard and report. This alert system will be implemented with the Github Repository API. Once the code has been submitted, the user will wait for the code analysis report and dashboard which will be either immediate or within a couple of minutes. Then, the user has the capability to view their updated code analysis reports on their account. This means being able to view previous versions of the code that were uploaded and how it has changed over time, and also viewing other reports that the user has requested in the past.

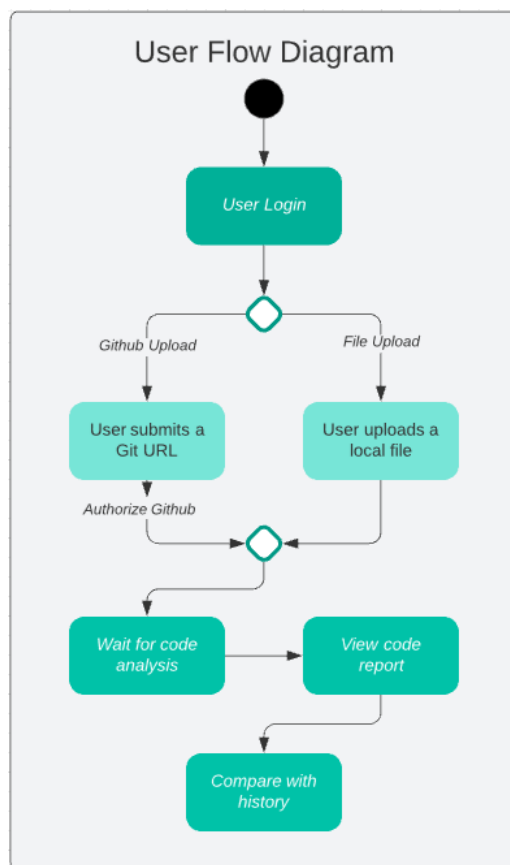


Figure 1 - User Flow

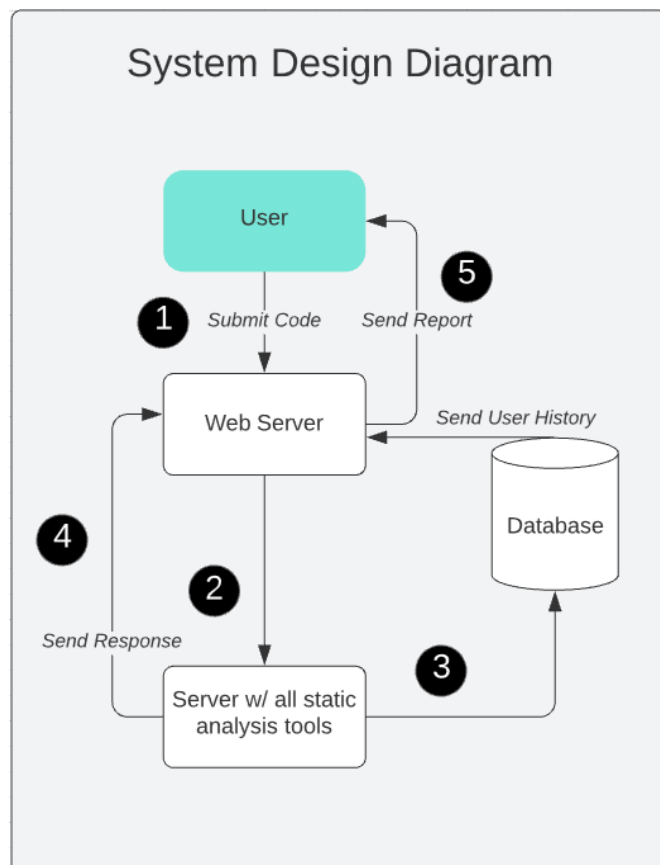


Figure 2 - System Design

As shown in figure 2, the main user interface that includes logic for accounts and sending/viewing code analysis results will be hosted on Amazon Web Services. A local server will hold all of our static analysis tools. This local server will be manually. The local server will communicate with the Amazon Web servers to provide quick feedback to the users. All of our results from the static analysis tools will be stored into a database. Amazon DynamoDB will be used as the database for this project. This database will store user history. Whenever the user wants to compare the results of their current running code to their previous code, the database will be used to pull the information from the previous code report.

For the front-end development, Flask will be used for the project. Flask is a micro web framework written in Python. Flask will allow the software-as-a-service to be built easily on a server. For the back-end development, we will use Python. Using Python for the whole project will allow the back-end and front-end communicate effectively. Python is great for storing data and pulling results from the different static analysis tools.

This project will be using multiple Amazon Cloud Services. For the servers, Amazon Web servers will be used and for the database, Amazon DynamoDB will be used. Amazon Cloud Services was chosen because they provide developers to use reliable and inexpensive cloud services. Both services used in this project are free to use. Once there is a need for more storage or faster services, the project can easily be moved to other Amazon Cloud Services. Amazon Cloud Services allows for easy and flexible storage. The service has built in security measures that will ensure that a user's code isn't leaked or deleted without their permission. This will allow Solana Code Analysis as a Service to have high data performance. The AWS Free Tier provides free trials for different products. Amazon EC2 instances can be used from the free tier. The EC2 instances will allow the project to run cloud applications with high performance.

Approach for design validation

Most of the project's goals are quantifiable and easy to test, but comparisons to competing solutions will have to be made to validate the product and goals.

Using at least three code analysis tools can be verified through inspection of the source code and through attaching each tool's result to the report for testing purposes. If the report shows three different tools creating results, this test will pass. On the live build, the tools should find a consensus between these results before passing the report to the user.

The file upload and github upload tasks will be tested by submitting files of different lengths along with identical files hosted on github. This feature will be validated if the result of each file is identical to its github counterpart. The result must also be non-empty to test that the upload is working correctly. On the live build, we should not require that identically named files be the same, because a local version can be different than the repository version.

To test the code and report history feature, a new user account will be created and 11 versions of the same code will be submitted. The system will pass this test if the last 10 code versions are available for comparison and the report history is viewable in both text and graphic format.

To verify the effectiveness of the product's code review tools, the same code will be submitted to several other blockchain code analysis tools for comparison.

To ensure that the overall design is effective and does not have security vulnerabilities, the website will be put under heavy load with many users trying to access their past results and request new ones at once. Validation will also be done to verify that the user's code is safe.

5 Engineering standards

5.1 Project management

Each team member started this project with a basic understanding of blockchain technology. Every member has experience in hardware systems design and Olivia has some experience with developing web applications. Everyone will help design and develop this project, but each team member will have a primary responsibility. Because of experience in project development, Arsh will be the team leader. He will also work on systems design because he has experience from his Capital One internship and personal projects he's worked on. Joseph will focus on software and systems design while keeping minutes during weekly meetings with the professor and TA. This is to ensure important details from the meeting aren't missed and to remember what questions were asked and the answers to them. By Joseph keeping track of minutes, the team won't have to ask the same questions at the next meeting. Because Joseph has experience with database creation and building tools primarily in C he will also focus on creating user login functionality and storing information in databases within the responsibility of system design. Kipp has experience in distributed systems and networking communication between different services so he will help with systems and hardware. Olivia has experience in creating web applications for classes and on club project teams and will focus on frontend design. This will be helpful when integrating github and other code review tools into the project. Everyone on the team has experience working with APIs.

One tool that will be used to manage the team will be bi-weekly meetings that address the subjects on the weekly report template. These meetings will give everyone an opportunity to bring up any concerns they have with their current tasks or get input from other team members. These meetings will also be used to facilitate brainstorming sessions, especially during the first couple of meetings. If any additional meetings are needed so that a member can become unblocked, there is a team GroupMe where they can communicate outside of class. The team will also use a Gantt chart to track their progress and be updated as tasks are completed. If changes need to be made to the chart, it will be updated weekly to help keep the project on track and meet deadlines. In order to keep track of documents, a team google drive was created. This ensures everyone has access to documentation and can update files as needed. For code and version control, a Github organization was created for this project. This will allow team members access to each other's finished code and allow for easier integration of the project. Github can also be used to track the progress of the product at each software development lifecycle stage.

Any finance decisions and purchases will be decided as a team. This is to ensure every member is up to date on what is being purchased and how it will be used for the project. All financial decisions will be written down and discussed with the professor.

5.2 Schedule of tasks, Pert and Gantt charts

The first phase of the project is focussed on getting each part of the project working locally and individually before combining them. Specifically, drafting the code upload and account management on the web server, and running the code analysis tools locally to test and examine the output reports. Lastly, the team will begin hosting the web server remotely.

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION (DAYS)	WEEK 1					WEEK 2					WEEK 3				
						M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
1	User Interface																			
1.1	Basic code upload		2/16/22	2/23/22	7															
1.2	Account login/logout/creation		2/23/22	2/25/22	2															
1.3	View code vulnerabilities		2/24/22	3/1/22	7															
1.4	View analysis report		2/28/22	3/2/22	2															
2	Pipeline																			
2.1	Host Website		2/28/22	3/3/22	3															
2.2	Set up server for code analysis		3/3/22	3/8/22	5															

The second of the phase of the project is focussed on setting up the web server to automatically accept code and return the reports. The team will start with no code analysis to keep the pipeline simple. Once the pipeline works, the blackbox code analysis tools will be added to the pipeline. Once this is complete, the team will spend the remaining time to polish the minimum viable product.

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION (DAYS)	WEEK 4					WEEK 5					WEEK 6				
						M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
2.2	Set up server for code analysis		3/3/22	3/8/22	5															
2.3	Send uploaded code to server		3/8/2022	3/10/2022	2															
2.4	Send test report back to the web server		3/10/2022	3/11/2022	1															
3	Code analysis																			
3.1	Run cargo-audit on code		3/11/22	3/16/22	5															
3.2	Run Soteria (free version) on code		3/14/22	3/17/22	3															
3.3	Run an additional code review tool		3/14/22	3/17/22	3															
3.4	Polish MVP		3/17/22	3/23/22	6															

The last phase of the project is focussed on adding the project's more unique features and working on stretch goals. The main goals of this phase will be the multi-tool report consensus, code history insights, and Github auto-report features. We are confident in our ability to implement the consensus system and Github auto-report, while we expect the code history to be a stretch goal.

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION (DAYS)	WEEK 7					WEEK 8					WEEK 9				
						M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
4	Additional Features																			
4.1	Github upload		3/28/22	3/30/22	2															
4.2	Report consensus		3/30/22	4/4/22	4															
4.3	Auto-report and email		4/4/22	4/7/22	3															
4.4	Save report and code history		4/7/22	4/11/22	4															
4.5	Display historical data with diffs		4/8/22	4/14/22	6															

5.3 Economic analysis

Economic viability: The cost for the creation of the service will be minimal. The service will initially run on the AWS Free Tier. Once Solana Code Analysis as a Service has higher engagement, we will use Amazon EC2 instances. Prototyping costs will also be minimal since the backend and frontend of the service will be created by the team. Since open source static tools will be used, there will be no expenses associated with the tools themselves unless complex tools with prices are added later in the project timeline.

Sustainability: Solana Code Analysis as a Service will utilize static analysis tools from multiple vendors. It is important that the safety of these tools are monitored daily or weekly to ensure that the tools are still secure and haven't been breached. The Solana Code Analysis as a Service development team will stay in contact with the developers of the different tools, so they are alerted if there is a new bug or vulnerability in any of the tools.

Manufacturability: For this project, the worst case scenario would be a result of either our server being under heavy load or a user trying to run the tool on a very large codebase. For the first case, we would be able to easily add additional servers since it will be running on AWS. For the second case, the user would be made aware that the more code they are submitting the longer it would take. It would also mostly depend on the speed of the open source analysis tools that we are using, so not really under our control. Based on how slow many of the alternative products online currently are, neither of these should pose a problem. This tool should comply with all regulations, proven by the many alternatives out there.

5.4 Societal, safety and environmental analysis

The societal impact of this project will be beneficial because more people will have access to writing better code which allows them to get involved in blockchain development. By using this tool to test and enhance their code, users will save resources because they won't have to run their code on their hardware. Additionally, this project will reduce the time it takes for developers to debug by catching these errors through our tool.

In terms of environmental impact, Solana is much more environmentally friendly than other traditional crypto that involves mining. However, because we are using cloud servers for the static analysis tools and the website, there will be CO₂ emissions higher than using physical servers. The project aims to minimize the environmental impacts by only using hardware that is absolutely necessary.

A potential downside to this project is loss of privacy for our users. Since one of the intended features is to keep track of their previously submitted tasks, this would mean keeping the user's code in a database. To reduce the risk of a customer's code getting compromised, their code will be encrypted before getting stored in the tool's database. In addition to this, all the necessary precautions will be taken when implementing this project.

5.5 Itemized budget

Expected costs:

- AWS EC2 Servers: \$512 and \$42.34/mo
- Host the website on a URL with domain: \$5
- Try out existing paid products that accomplish a similar goal: \$25-\$100
- Additional database space if needed

6 References

Here you acknowledge all the documents that you used as references throughout the text. Please use the IEEE reference format (www.ieee.org/pubs/transactions/auinfo03.pdf)

- [1] “Blockchain code review: Blockchain Analyzer,” Bloqchain. [Online]. Available: <https://thebloqchain.com/audit/code-review/#:~:text=Code%20Review%20is%20the%20automated,analysis%20of%20the%20source%20code.&text=Blockchain%20code%20review%20and%20Blockchain,itsel%20in%20its%20own%20environment>. [Accessed: 10-Feb-2022].
- [2] “Security Services for solana programs,” Soteria. [Online]. Available: <https://www.soteria.dev/>. [Accessed: 12-Feb-2022].
- [3] “The guardians of web3 Digital assets,” AnChain.AI, 12-Feb-2022. [Online]. Available: <https://anchain.ai/>. [Accessed: 14-Feb-2022].
- [4] “Code quality and code security,” *SonarQube*. [Online]. Available: <https://www.sonarqube.org/>. [Accessed: 14-Feb-2022].
- [5] “Fast and reliable static analysis platform,” DeepSource. [Online]. Available: <https://deepsource.io/>. [Accessed: 14-Feb-2022].
- [6] <https://solana.com/solana-whitepaper.pdf>

7 Appendices (1 point)

7.1 Product datasheets

N/A

7.2 Bios and CVs

Olivia Ornelas:

Olivia Ornelas is a student at Texas A&M University graduating with a computer science major and a business and math minor. She is interested in distributed systems, and network security. Olivia interned with the United States Air Force communications division where she used a new tool to find anomalies within the Air Force's network. Recently, she interned with the same department and created a tool to virtualize new employee inprocessing paperwork and consulted on integrating a Hadoop based application into the Air Force's network.

Arsh Kabarwal:

Arsh Kabarwal is a student at Texas A&M University graduating with a computer science major and a business minor. He is interested in blockchain technology, data science, and machine learning. He has interned at Capital one where he worked on finding anomalies and irregularities on a data pipeline. He currently teaches Python and Data Science to middle schoolers.

Joseph Carpman:

Joseph Carpman is a computer science student at Texas A&M with a focus on networking and cybersecurity. He is intrigued by cryptography and the imperceptible innovations that make our digital world work. Joseph interned with Heart Ally books to remake their database so that it could be accessed remotely. He is currently employed as a 3D printing technician.

Kipp Corman:

Kipp Corman is a student at Texas A&M University graduating with a computer science major and statistics minor. He is passionate about distributed systems and has taken multiple classes related to the field. Kipp interned at Confluent where he worked on containerized deployment with Kubernetes and networking protocols. Prior to that, he was at Nvidia doing full-stack development on a desktop application.

Olivia Ornelas

801 Marion Pugh, Apt 4101A
College Station, TX 77840
Github: <https://github.com/oliviaor254>

(210)882-9767
oliviaaornelas@gmail.com

OBJECTIVE

Seeking a full time offer in fall 2022

SKILLS

- Certified Scrum Master
- C++ (Intermediate)
- Python (Intermediate)
- Java (beginner)
- ReactJS (beginner)

EDUCATION

Texas A&M University, College Station, Texas

May 2022

Bachelor of Science in Computer Science, Minor in Cybersecurity, Minor in Mathematics

GPR: 3.48

Related coursework:

- Data Structures and Algorithms
- Computer Human Interaction (currently taking)
- Network and Distributed Systems in C

EXPERIENCE/PROJECTS

Air Force Civilian Service, San Antonio, Texas

May 2021- July 2021

Premier College Intern Program (Year 2)

- Designed a program that entered a unit's in-processing checklist into JIRA server based off the new person's username and Project key using Python and Jira's API.
- Conducted updates at daily scrum meetings with team.
- Mentored team members on programming as lead developer.
- Reviewed team member's code before merging into main branches on BitBucket.

Traceroute Program, Texas A&M University

April 2021

- Designed a program that imitated the ping function using ICMP protocol.
- Took an IP address or website name as input and print how many router jumps it took, listing the IP address at each one until it reaches the destination.
- Displays '*' if a timeout occurs after 3 pings are tried.

Air Force Civilian Service, San Antonio, Texas

June 2020-August 2020

Premier College Intern Program (Year 1)

- Researched how to detect cyber threats using a big data platform, and generated models with it.
- Conducted personal updates at weekly meetings with individual and team management.
- Gained better understanding of Leadership in the Air Force and how the military branch operates.

LEADERSHIP

MSC Hospitality, Texas A&M University

January 2021-Present

Member, January 2021-Present

- Promote aggie values through community service and leading tours of the campus, MSC, and bonfire memorial.

Society of Women Engineers (SWE), Texas A&M University

September 2018-Present

SWEInspire committee, January 2020-May 2020

- Visited local elementary schools in groups of 3 and preformed interactive activities to educate elementary students about different types of engineering on a bimonthly basis.

Aggie Coding club, Texas A&M University

September 2019-December 2020

Member, September 2019-December 2020

- Participated in 2 project teams: Study Buddy, an app aimed to digitize handwritten notes (stopped due to Summer); GrocerEZ: a tool to find cheapest prices given a grocery list (discontinued due to cost greater than feasibility)

WORK EXPERIENCE

Tech Intern

Jun. 2021 – Aug. 2021

Capital One

- Leveraged AWS' Machine Learning as a Service (MLaaS) algorithms to detect changepoints and anomalies within metadata for our data pipelines, which will contribute to the Capital One pipeline's health monitoring dashboards
- Performed trend and data analysis on data tables and presented analysis to stakeholders

Student Technician

Mar. 2021 – Present

Texas A&M Engineering IT Group

- Serving in a capacity of administrative or technical support for a department/division at Texas A&M Engineering
- Providing assistance to support staff for specialized projects via distinct programs and particular needs

Coding Instructor and Mentor

Jun. 2020 – Present

The Coding School - codeConnects

codeConnects is an initiative developed by The Coding School, dedicated to empowering the next generation through code.

- As Coding Instructor and Mentor, I taught one on one and group lessons. Curriculum I taught included Python Fundamentals, Python Turtle, and Intro to Web Development.

Digitization Student Coordinator

Jan. 2020 – Jan. 2021

Texas A&M Library – Digital Services

- Loading and managing photography software
- Sorting through the online library catalog
- Working with AutoCAD and Photoshop

Computer Science Intern

May. 2018 - Sept. 2018

Chili Peppers Computer Services

Computer Services based in Texas is a Premium IT Support Services company that focuses on servicing connected internet-enabled devices

- Identified customer needs and then sold computer parts to customers
- Worked in a team to provide repair services by identifying and pinpointing problems

EDUCATION

Texas A&M University

Graduation May 2022

Bachelor of Computer Science

Minor in Business

- GPA: 3.711
- Distinguished Student Award for Spring 2019 and Dean's Honor Award for Fall 2019
- Extra – Curricular: Aggie Coding Club, Engineering Mentorship Council, Brothers in Engineering and Science and Technology

PROJECTS

Market Dash!

Apr. 2020

- Game that allows the user to roam a market type place where the user attempts to collect as many groceries as possible, while attempting to keep a social distance. 4 levels where each level gets progressively harder

VizCal Visual Calendar

Nov. 2021

- Lightweight, user-friendly calendar interface that highlights task management and prioritization. Evaluations were carried out to test the functionality of the product

SKILLS & INTERESTS

- **Skills:** Proficient in C++ and Python; Moderate in Java, Solidity, Haskell, R-Studio, HTML, CSS, and JavaScript; Knowledge of Excel, PowerPoint, Word, Protocol, Photoshop, etc; Certified Secure Software Engineer Certification
- **Interests:** Working with a team, User-based applications, Projects that allow creativity, Learning and Using new skills

Joseph Carpman

Academic Goals

I aim to broaden my skill set and create an interesting project to attract exceptional employers.

Get in touch!

Mobile:

832-712-6629

Email:

jcarpman100@tamu.edu

Website:

people.tamu.edu/~jcarpman100

Subjects of Interest

- > Computer Networking
- > Network Security
- > Cryptography
- > Cybersecurity
- > Software Development
- > Software Reverse Engineering
- > Database Development

Work Experience

Student Technician

Fischer Engineering Design Center | March 2019 - Present

- Used VBA scripts and Excel Data validation to streamline prototyping requests
- Cross trained fellow technicians to ensure that everyone was capable of fulfilling any request that a student could have.
- Worked closely with a small team to keep 3D prints and other long processes running around the clock

Database Intern

Heart Ally Books | November 2021 - January 2022

- Researched database automation for small businesses
 - Migrated Database structure and contents between MySQL and FMP databases
-

Academic History

Texas A&M University

Senior Student | Computer Science

Cybersecurity focused computer science program
Completed many rigorous engineering courses
Achieved Dean's Honor Roll

Prior Projects

- **Ram-Cache Simulator**: C program to simulate the paging and usage of Random Access Memory in a computer.
- **Seam Carving**: Content Aware image cropping program that utilized a seam carving function to determine what parts of the picture to remove.
- **Bouncer**: Python Script for procedurally generating ready-to-cut engineering drawings for use by a laser-cutter
- **Third Eye**: Accessibility device for users with poor sight. Used machine vision and an ultrasonic sensors to warn users of incoming hazards and identify their surroundings.
- **Trendify**: Web app that offered users statistics for their Spotify listening history. Included accessibility options such as text to speech and colorblind options.

Kipp Corman

☎ (201) 857-9772 | ✉ kcorman@tamu.edu | 📧 kcorman0 | 🌐 kipp

Education

Texas A&M University

BACHELOR OF SCIENCE IN COMPUTER SCIENCE, MINOR IN STATISTICS

- GPA: 3.96/4

College Station, TX

Aug 2018 - May 2022

Experience

Confluent

Remote

BACKEND SOFTWARE ENGINEER INTERN

Jun 2021 - Aug 2021

- On the private cloud team working on Confluent for Kubernetes.
- Designed a webhook server template using Go to validate and mutate Kubernetes commands before they are executed. Provides the user with a description of why their command was not accepted, and allows GitOps applications to be easily debugged.
- Set up certificate management to allow TLS communication between the webhook server and Kubernetes API server.
- Created admission controllers to stop Kafka topic parameters from being modified, and to prevent Kafka pods from being deleted when they have under replicated partitions.

NVIDIA

Redmond, WA

SOFTWARE ENGINEER INTERN

Jan 2020 - May 2020

- Leveraged C++ and TypeScript/Angular to improve telemetry and the user experience of the GeForce Now desktop client.
- Transitioned Mac auto-update system to extract files rather than mounting, reducing the amount of failed updates.
- Revamped the log collection method to prevent data from being lost during an upload, increasing the amount of logs collected from users.
- Automated Kibana logs to show summary information without having to attach a debugger, allowing them to be easily searched or filtered.
- Engineered and presented a proof of concept of the integration of GeForce Now into a video game distribution application.

Skills

Languages: C++, Python, Go, Java, SQL, JavaScript/CSS, HTML

Technologies: Git, Kubernetes, Node.js, Django, MongoDB, Postgres, SQLite

Concepts: Data Structures & Algorithms, Operating Systems, Concurrency, Computer Organization, Machine Learning, Agile

Projects

Wikipedia Search

Fall 2020

C++

- Programmed a multi-threaded application using a variation of the producer-consumer algorithm to quickly count the number of occurrences of each word in the entire Wikipedia.
- Designed an efficient concurrent hash table that involves merging different thread's hash tables together.

Elation

Fall 2020

PYTHON - DJANGO, SQLITE, AWS ELASTIC BEANSTALK

- Built a website that can determine a user's mood through text or speech input and gives them music recommendations. Users can view their past entries and visualize how their mood changes over time.

Sheet Music Reader

Summer 2019

PYTHON - FLASK, OPENCV

- Developed a web application to convert YouTube videos that scroll through sheet music into PDFs by using HSV color detection.

Urban vs. Nature Image Classification

Summer 2018

PYTHON - TENSORFLOW

- Web scraped a 24,000 image dataset and designed a convolutional neural network to classify them into above categories with 90% accuracy.

Involvement

Texas A&M RoboMaster Robotics

College Station, TX

COMPUTER VISION - MODELING LEAD

Aug 2019 - Present

- Constructed a low latency object detector that draws bounding boxes around robot's armor plates using Yolov4 and TensorRT.
- Led weekly scrum meetings and taught new members basic machine learning concepts.