Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics O'Reilly Institute,
Trinity College, Dublin 2, Ireland

Decentralized Search Engine: Enhancing Privacy, Security, and Transparency in Web Search

Saumya Bahuguna

Supervisor: Dr. Hitesh Tewari

April 17, 2023

A Final Year Project submitted in partial fulfilment
of the requirements for the degree of
BA(MOD) Computer Science

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Signed: __________ Saumya Bahuguna __________ Date: April 17, 2023

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Signed: ______________ Saumya Bahuguna ______________ Date: April 17, 2023

Abstract

Centralized search engines raise privacy, security, and neutrality concerns due to their data collection and potential to manipulate search results. This Final Year Project (FYP) proposes a decentralized search engine to address these issues and offer a more secure, private, and transparent search experience.

The FYP employs blockchain technology, smart contracts, and a peer-to-peer (P2P) network architecture to develop the decentralized search engine. The custom blockchain is created using the Geth client, while the frontend and backend are designed with modern web technologies and Node.js, respectively. A private blockchain is utilized to maintain user anonymity, with users paying a nominal fee for queries to ensure system sustainability.

The proposed decentralized search engine is compared to popular centralized search engines and other decentralized alternatives to evaluate its performance and effectiveness. The comparison encompasses aspects like privacy, security, transparency, response time, throughput, and scalability. The results demonstrate the potential of the decentralized search engine in addressing centralized search engines' concerns and highlight areas for future research and improvement.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Hitesh Tewari, for his unwavering guidance, valuable insights, and encouragement throughout this project. His expertise and creativity in the field of blockchain have been instrumental in helping me delve deeper into this fascinating technology.

I am incredibly grateful to my family for providing me with countless opportunities and for their constant support and encouragement throughout my academic journey. Their belief in my abilities has motivated me to push my boundaries and achieve my goals.

Additionally, I am thankful to all my peers and the esteemed faculty members of Trinity College for their intellectual challenges, constructive feedback, and invaluable lessons. Their collective wisdom and guidance have played a pivotal role in shaping my knowledge and skills.

Finally, I would like to thank everyone who has directly or indirectly contributed to the successful completion of this project. Your support and encouragement have been invaluable in helping me reach this milestone.

Contents

1 Introduction 1
1.1 Problem Statement . . . 1
1.2 The Decentralized System . . . 1
1.3 Objective and Scope of the Project . . . 2
1.4 Overview of the System . . . 3

2 State of Art 5
2.1 Centralized Search Engines: Mechanisms, Privacy and Security Concerns . . 5
2.1.1 Introduction and How They Work . . . 5
2.1.2 Privacy and Security Threats Due to Data Collection . . . 5
2.1.3 Use of Data and User Information for Target Marketing and Advertising 6
2.2 Decentralized Search Engines: Mechanisms, Benefits, and Challenges . . . 7
2.2.1 Introduction . . . 8
2.2.2 How Decentralized Search Engines Work . . . 8
2.2.3 Benefits of Decentralized Search Engines . . . 8
2.2.4 Challenges Faced by Decentralized Search Engines . . . 9
2.2.5 Comparison of Decentralized Search Engines . . . 10
2.2.6 Presearch . . . 12
2.3 Technology Used . . . 13
2.3.1 Blockchain . . . 13
2.4 Ethereum . . . 17
2.4.1 Decentralized Applications (dApps) . . . 17
2.4.2 Geth . . . 17
2.5 Smart Contracts . . . 18
2.5.1 Components of Smart Contracts . . . 18
2.5.2 Execution of Smart Contracts . . . 19
2.5.3 Advantages of Smart Contracts . . . 19
2.5.4 Events in Smart Contracts . . . 19
2.6 Web3.js . . . 20

2.7 GunDB 21
2.8 Node.js 23

3 System Architecture and Design 24
3.1 Front-end Design 24
3.2 Back-end Design 25
3.3 Event Listener and Real-time Data Processing 25
3.4 Search Script 25
3.5 Data Storage and Synchronization 26
3.6 Justification for the Chosen Design 26
3.6.1 Decentralization 27
3.6.2 Privacy and Security 27
3.6.3 Scalability and Performance 27
3.6.4 Flexibility and Extensibility 27
3.6.5 User Experience 27

4 Implementation 29
4.1 Private Blockchain 29
4.2 Smart Contract Deployment and Interaction 31
4.3 Setting up a Node.js Server with Express 33
4.4 Integrating GunDB for Data Storage 35
4.5 Event Listener Implementation 35
4.6 Search Script and Multi-threading 36

5 Testing and Evaluation 38
5.1 Unit Testing 38
5.2 Scalability Testing 39
5.3 Performance Testing 39
5.4 Privacy and Security Evaluation 40
5.5 Size and Memory 40
5.6 Conclusion 40

6 Conclusion 41
6.1 Summary of the Project 41
6.2 Achievements and Challenges 41
6.3 Future Work and Improvements 41

List of Figures

1.1 A high level system architecture 2
1.2 A screenshot of a search on the system 4

2.1 A high level diagram of how google works 6
2.2 P2P Network 9
2.3 Working of a Centralized vs decentralized search engine 12

3.1 The image shows the System componenets 24

4.1 Communication between components 37

5.1 A graph showing responsetime vs Throughput 39

1 Introduction

1.1 Problem Statement

In today's digitally connected world, search engines play a pivotal role in providing relevant information to users. However, traditional centralized search engines pose significant concerns related to privacy and security, as they collect massive amounts of user data. This data collection facilitates targeted marketing and advertising, leading to potential misuse of user information. Additionally, centralized search engines can manipulate search results, impacting the neutrality of the information presented to users. The objective of this project is to develop a decentralized search engine that addresses these concerns and provides a more secure, private, and transparent search experience for users.

1.2 The Decentralized System

The growing concerns regarding security and privacy in centralized systems have highlighted the need for a more robust and user-centric approach to information retrieval. As part of this Final Year Project (FYP), a comprehensive solution has been developed to address these challenges by implementing a decentralized system that prioritizes user anonymity and data protection.

Leveraging the power of private blockchain technology and smart contracts, this Final year project (FYP) aims to create an environment where users can confidently search for information while maintaining their security and privacy. By employing a private blockchain, the system ensures that user data is securely stored and protected from unauthorized access, while also providing a reliable and transparent audit trail of transactions.

Smart contracts play a pivotal role in this solution by facilitating trustless interactions between users and the search engine. These self-executing agreements enable users to pay a minimal fee, as low as a hundredth of a cent per query, in exchange for enhanced security and privacy. This nominal cost underscores the value of safeguarding user information and promotes a more responsible approach to data handling.

The proposed decentralized system for this FYP signifies a paradigm shift in the way search engines are designed and operated. By prioritizing user anonymity and data protection, this solution offers a compelling alternative to traditional centralized systems that often compromise user privacy for the sake of convenience and profit. Ultimately, the FYP aims to foster a more secure and privacy-conscious search experience, paving the way for a new era of information retrieval that respects user rights and upholds data protection principles. The figure 1.1 Shows a high level design of proposed solution.

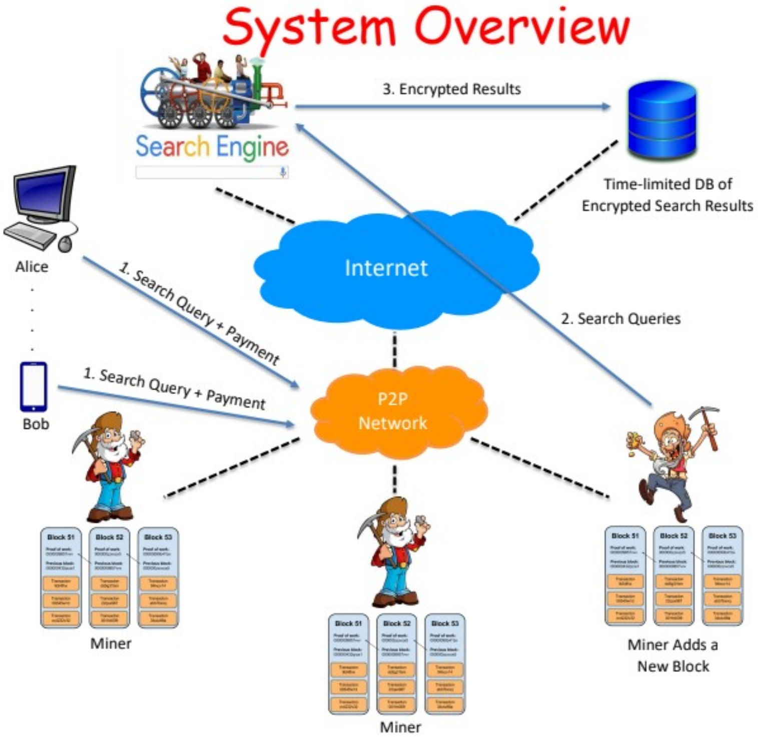


Figure 1.1: A high level system architecture

1.3 Objective and Scope of the Project

The primary objective of this project is to design and implement a decentralized search engine that offers enhanced privacy, security, and transparency compared to existing centralized search engines. The scope of the project includes:

- Analyzing the limitations and drawbacks of centralized search engines concerning data collection, privacy, and security.
- Studying the advantages and disadvantages of decentralized search engines and their potential to overcome the issues associated with centralized search engines.
- Identifying and utilizing appropriate technologies such as blockchain, Ethereum, smart contracts, Web3.js, and GunDB to develop the decentralized search engine.

- Designing and implementing a robust system architecture that incorporates front-end, back-end, event listeners, search scripts, and data storage components.
- Testing and evaluating the developed system in terms of performance, scalability, and security.

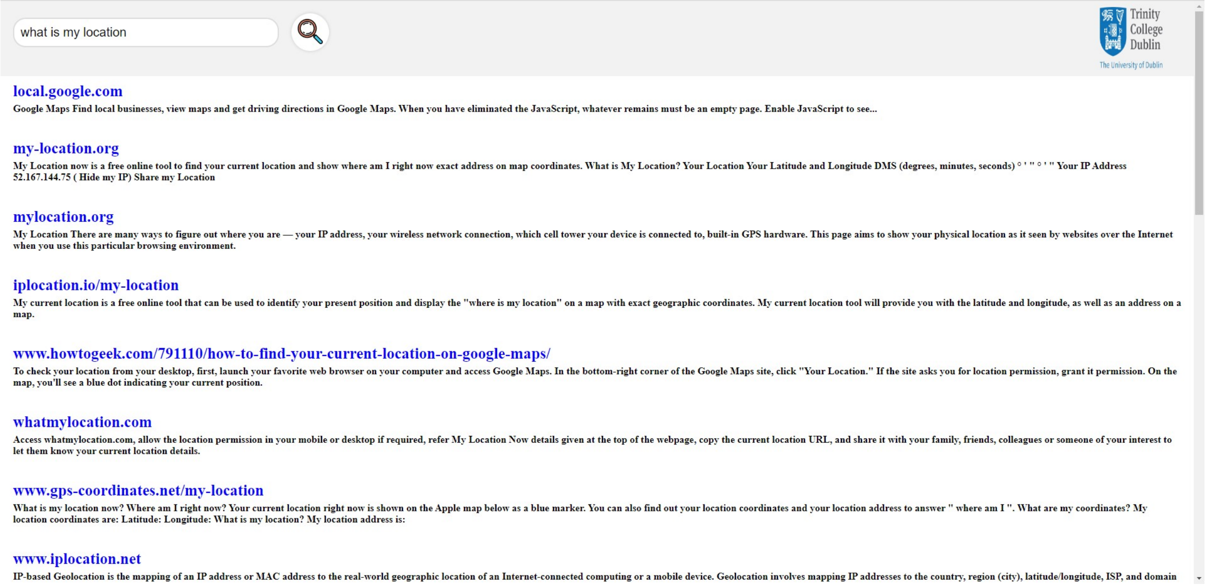
1.4 Overview of the System

The proposed decentralized search engine system comprises the following components:

- Front-end design: The user interface of the search engine is developed using Node.js and Express, providing an intuitive and user-friendly experience.
- Back-end design: The system leverages Ethereum smart contracts and Web3.js to manage search queries, handle transactions, and maintain data integrity.
- Event listener and real-time data processing: An event listener monitors the smart contract events to identify new search queries and process them in real time.
- Search script: A Python web scraping script retrieves search results from DuckDuckGo, a privacy-focused search engine, ensuring that the data collection is minimal and secure.
- Data storage and synchronization: GunDB, a decentralized database, is used for storing and synchronizing search results across multiple nodes, enhancing the system's resilience and scalability.

The project's primary focus is to provide users with a decentralized search engine that offers a more secure, private, and transparent search experience. By addressing the drawbacks of centralized search engines, the decentralized search engine aims to minimize the potential misuse of user information, ensuring that search results remain neutral and unbiased. Additionally, the report will present testing and evaluation results, highlighting the performance, scalability, and security aspects of the system. Finally, the report will conclude with a summary of the project, its achievements and challenges, and potential future work and improvements.

With a thorough understanding of the objectives, scope, and system overview, we can now proceed with the report's subsequent chapters, providing a comprehensive account of the project's development and outcomes.



what is my location

local.google.com

Google Maps Find local businesses, view maps and get driving directions in Google Maps. When you have eliminated the JavaScript, whatever remains must be an empty page. Enable JavaScript to see...

my-location.org

My Location now is a free online tool to find your current location and show where am I right now exact address on map coordinates. What is My Location? Your Location Your Latitude and Longitude DMS (degrees, minutes, seconds) ° ' " ° ' " Your IP Address 52.167.144.75 (Hide my IP) Share my Location

mylocation.org

My Location There are many ways to figure out where you are — your IP address, your wireless network connection, which cell tower your device is connected to, built-in GPS hardware. This page aims to show your physical location as it seen by websites over the Internet when you use this particular browsing environment.

iplocation.io/my-location

My current location is a free online tool that can be used to identify your present position and display the "where is my location" on a map with exact geographic coordinates. My current location tool will provide you with the latitude and longitude, as well as an address on a map.

www.howtogeek.com/791110/how-to-find-your-current-location-on-google-maps/

To check your location from your desktop, first, launch your favorite web browser on your computer and access Google Maps. In the bottom-right corner of the Google Maps site, click "Your Location." If the site asks you for location permission, grant it permission. On the map, you'll see a blue dot indicating your current position.

whatmylocation.com

Access whatmylocation.com, allow the location permission in your mobile or desktop if required, refer My Location Now details given at the top of the webpage, copy the current location URL, and share it with your family, friends, colleagues or someone of your interest to let them know your current location details.

www.gps-coordinates.net/my-location

What is my location now? Where am I right now? Your current location right now is shown on the Apple map below as a blue marker. You can also find out your location coordinates and your location address to answer " where am I ". What are my coordinates? My location coordinates are: Latitude: Longitude: What is my location? My location address is:

www.iplocation.net

IP-based Geolocation is the mapping of an IP address or MAC address to the real-world geographic location of an Internet-connected computing or a mobile device. Geolocation involves mapping IP addresses to the country, region (city), latitude/longitude, ISP, and domain

Figure 1.2: The figure shows a screenshot of a search done on the system where the location of the user is being asked and as we can see there is no response in regards to the location

2 State of Art

2.1 Centralized Search Engines: Mechanisms, Privacy and Security Concerns

2.1.1 Introduction and How They Work

Centralized search engines have become an indispensable part of our daily lives, serving as the primary source of information retrieval on the internet. These search engines, such as Google, Bing, and Yahoo, employ vast networks of data centres and advanced algorithms to crawl, index, and rank web pages based on their relevance to user queries. [1]

At the core of centralized search engines lies a centralized server infrastructure responsible for storing, processing, and serving search results to users. This infrastructure is owned and controlled by a single organization, which can monitor, manipulate, and monetize user data without any external oversight.

When a user submits a query to a centralized search engine, the request is sent to the search engine's data centres, where it is processed by the search algorithm. The algorithm then scans its index of web pages, evaluating and ranking them according to various factors, such as keyword relevance, page authority, and user behaviour. The search engine subsequently returns a list of search results, ordered by their perceived relevance to the user's query. Figure 2.1 is a sequence diagram showing how Google works and collects users' data

2.1.2 Privacy and Security Threats Due to Data Collection

While centralized search engines provide users with quick and convenient access to information, their centralized nature raises significant privacy and security concerns. [1] These search engines collect vast amounts of user data, including search queries, IP addresses, device information, location data, and browsing history. This data collection enables search engines to build detailed profiles of individual users, which can be used for various purposes, often without the users' knowledge or consent.

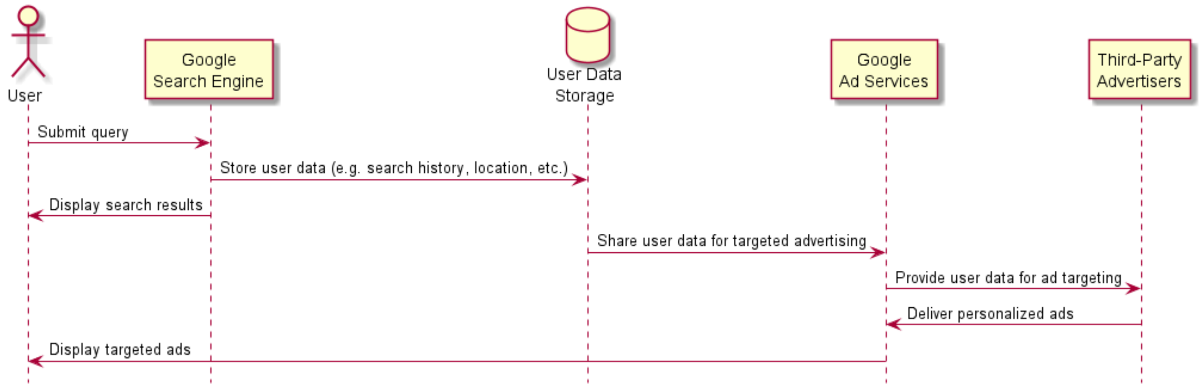


Figure 2.1: A high level diagram of how google works

One of the primary security threats posed by centralized search engines is the potential for data breaches. Since these search engines store user data in centralized databases, they become attractive targets for cybercriminals seeking to gain unauthorized access to sensitive information. A successful data breach can result in the exposure of millions of users' data, putting their privacy and security at risk.

Moreover, centralized search engines' extensive data collection practices can lead to unwarranted surveillance and the erosion of user privacy. By monitoring and analyzing users' search queries and browsing patterns, search engines can infer users' interests, preferences, and even their beliefs, effectively undermining the users' right to privacy and anonymity online.

2.1.3 Use of Data and User Information for Target Marketing and Advertising

The data collected by centralized search engines is a valuable commodity, particularly for targeted marketing and advertising. These search engines often monetize user data by using it to deliver personalized ads and promotional content, tailored to users' interests and browsing habits. This practice, known as targeted advertising, enables advertisers to reach specific audiences with a higher likelihood of engagement and conversion.

While targeted advertising can improve the relevance of the ads users see, it also raises ethical and privacy concerns. By leveraging user data to create targeted ads, centralized search engines essentially commoditize users' personal information, profiting from their online activities without providing them with adequate control over their data.

Furthermore, the use of user data for targeted advertising can lead to unintended consequences, such as the reinforcement of filter bubbles and echo chambers. By serving users content that aligns with their existing interests and beliefs, centralized search engines may inadvertently contribute to the polarization and fragmentation of online discourse, undermining the open and diverse nature of the internet. [2]

In conclusion, centralized search engines, while undoubtedly valuable in providing access to information, present considerable privacy and security challenges due to their data collection practices and centralized infrastructure. The use of user data for targeted advertising further exacerbates these concerns, highlighting the need for alternative search solutions that prioritize user privacy and data protection. Decentralized search engines, such as the one proposed in this FYP, offer a promising path forward, empowering users with greater control over their online experience while mitigating the risks associated with centralized data collection and storage.

By developing and implementing a decentralized search engine as part of this FYP, we aim to address the privacy and security concerns inherent to centralized search engines. By leveraging blockchain technology, smart contracts, and a privacy-focused infrastructure, our proposed solution will provide users with a secure, transparent, and private alternative for their online search needs.

This decentralized search engine will maintain user anonymity and limit data collection by utilizing a private blockchain and smart contracts to process search queries. Users will be charged a minimal fee for each query, ensuring that the costs associated with maintaining privacy and security are kept low. By eliminating the need for extensive data collection and centralized storage, our proposed search engine will minimize the risks associated with data breaches and unwarranted surveillance.

Furthermore, by adopting a decentralized architecture, we can reduce the potential for search result manipulation and promote a more neutral and unbiased presentation of information. This will help counteract the issues of filter bubbles and echo chambers often associated with targeted advertising in centralized search engines.

In summary, the decentralized search engine proposed in this FYP represents a promising solution to the privacy and security challenges posed by traditional centralized search engines. By emphasizing user privacy, data protection, and neutrality in search results, our project aims to empower users with a secure and transparent alternative for their online search needs, ultimately contributing to a more open and diverse internet landscape.

2.2 Decentralized Search Engines: Mechanisms, Benefits, and Challenges

[3]

2.2.1 Introduction

The concept of decentralized search engines has gained significant attention in recent years due to the increasing concerns over privacy, data centralization, and censorship on the internet. Decentralized search engines provide a way to search for information while ensuring data privacy, fairness, and network resilience. This paper explores the working mechanism of decentralized search engines, their benefits, and the challenges they face.

2.2.2 How Decentralized Search Engines Work

Decentralized search engines employ a distributed network of nodes or computers to store, index, and retrieve data, as opposed to traditional centralized search engines, which rely on a single entity or a small group of entities to manage and control the entire search process.

Peer-to-Peer Networks

The underlying infrastructure of decentralized search engines is based on peer-to-peer (P2P) networks, where each node in the network acts as both a client and a server, sharing resources and information with other nodes. [4] The process of searching in a P2P network is executed using a distributed hash table (DHT), which is a data structure that efficiently maps keys to values across the nodes in the network.

Content Indexing and Retrieval

In decentralized search engines, content indexing is performed locally on each participating node. Each node is responsible for indexing its own content and then sharing the index with other nodes in the network. This ensures that no central authority controls the indexing process, and all nodes contribute to building the search index. The content retrieval process involves querying the DHT to identify the nodes holding the desired information.

2.2.3 Benefits of Decentralized Search Engines

Enhanced Privacy

One of the most significant advantages of decentralized search engines is the enhanced privacy they provide to users. In traditional centralized search engines, user data is collected and stored, which can lead to privacy breaches and surveillance concerns. Decentralized search engines, on the other hand, do not rely on a single entity for data storage or retrieval, which prevents the centralized collection of user data.

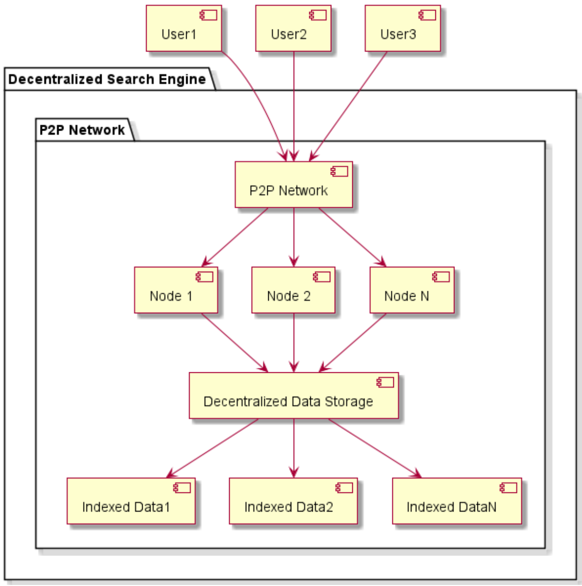


Figure 2.2: This component diagram illustrates the P2P network and decentralized data storage in a decentralized search engine. Users interact with the P2P network, which is a cluster of different nodes, which are connected to decentralized data storage. The data storage contains indexed data distributed across the network, ensuring that no single entity controls the information or search results

Resistance to Censorship

Decentralized search engines are inherently resistant to censorship due to their distributed nature. With no central authority controlling the indexing or retrieval of data, it becomes challenging for governments or other entities to enforce censorship on specific content or websites. This ensures that information remains accessible to users without any external interference.

Network Resilience

The distributed nature of decentralized search engines also ensures network resilience, as there is no single point of failure in the system. If a node goes offline or is compromised, other nodes can continue to operate and serve search queries. This ensures that the search engine remains operational even in the face of targeted attacks or network failures.

2.2.4 Challenges Faced by Decentralized Search Engines

Scalability

One of the main challenges faced by decentralized search engines is scalability. The distributed nature of these systems makes it difficult to handle large volumes of data and search queries, especially as the number of participating nodes increases. Moreover, the reliance on P2P networks and DHTs can result in increased latency and reduced query performance compared to centralized search engines.

Incentives for Participation

Another challenge faced by decentralized search engines is providing adequate incentives for users to participate in the network. Since these systems rely on user participation for content indexing and retrieval, it is crucial to ensure that users are motivated to contribute their resources and computing power. This can be achieved through various mechanisms, such as token-based incentives or reputation systems, which reward users for their contributions to the network.

Quality and Relevance of Search Results

Decentralized search engines also face the challenge of ensuring the quality and relevance of search results. In the absence of a central authority, it can be difficult to curate and rank search results effectively. Developing efficient algorithms and methods for ranking search results in a decentralized environment remains an area of active research.

Security and Trust

Lastly, security and trust are significant concerns in decentralized search engines. Since these systems operate without a central authority, it becomes essential to establish trust between nodes and protect the network from malicious actors. Techniques such as cryptography and consensus algorithms can be employed to ensure data integrity and maintain trust within the network.

2.2.5 Comparison of Decentralized Search Engines

Comparing decentralized search engines to their centralized counterparts reveals key differences in how they approach data storage, retrieval, and processing. Centralized search engines often prioritize performance and relevance, leveraging extensive data collection and advanced algorithms to deliver accurate search results. However, this approach can come at the expense of user privacy and security, as centralized search engines are more susceptible to data breaches and misuse of personal information.

Decentralized search engines offer several advantages over centralized search engines in terms of privacy, security, and user control. This section will discuss how decentralized search engines like Presearch, YaCy, Bitclave, and the FYP system can be better than some centralized ones, such as Google and Bing.

Privacy: One of the primary concerns with centralized search engines is their data collection and storage practices . Centralized search engines often track user behavior, search queries, and browsing history, which can be used for targeted advertising and other purposes . In contrast, decentralized search engines prioritize user privacy by minimizing data collection and storage. For example, the FYP system uses blockchain technology and a peer-to-peer database (GunDB) to ensure that user information remains private and secure. Similarly, Presearch, YaCy, and Bitclave focus on protecting user privacy by reducing the amount of data collected and stored.

Security: Centralized search engines can be vulnerable to hacking, data breaches, and censorship . Decentralized search engines, on the other hand, offer enhanced security due to their distributed nature. In a decentralized search engine, data is stored across multiple nodes, making it more difficult for hackers to compromise the system or for governments to censor search results . The FYP system, for example, leverages the security features of blockchain technology and smart contracts to protect user data and maintain the integrity of search results.

User Control and Incentives: Decentralized search engines often provide users with greater control over their data and search experience. For instance, Presearch rewards users with PRE tokens for their search activities, giving them an incentive to participate in the ecosystem . The FYP system also allows users to have more control over their search experience, as they can customize their search results by choosing which nodes to trust and which data sources to include. By providing users with control and incentives, decentralized search engines encourage active participation and help to maintain a vibrant and diverse ecosystem. [5]

Censorship Resistance: Decentralized search engines are more resistant to censorship than centralized ones . In centralized search engines, search results can be manipulated or censored by the search engine provider, governments, or other entities . Decentralized search engines, however, distribute search results across multiple nodes, making it more difficult for any single entity to control or censor content. This feature ensures that users have access to unbiased and uncensored information, promoting freedom of expression and access to knowledge.

Competition and Innovation: Decentralized search engines can foster competition and innovation in the search engine market . Centralized search engines, like Google, have a dominant market share, which can stifle innovation and lead to monopolistic practices . Decentralized search engines, by offering an alternative, can encourage competition and drive innovation in the search engine space, ultimately benefiting users with better search experiences and greater choice.

Despite these advantages, decentralized search engines also face some challenges. For instance, they may not yet provide the same level of search result accuracy and comprehensiveness as centralized search engines . Additionally, decentralized search engines may struggle with issues related to scalability, performance, and user adoption. However, as the technology behind decentralized search engines continues to evolve and mature, it is likely that these challenges will be addressed and overcome.

In conclusion, decentralized search engines offer a promising alternative to traditional centralized search engines, addressing many of the privacy and security concerns associated with the latter. While they may face challenges in performance and adoption, their unique advantages make them an increasingly attractive option for users seeking a more secure and transparent search experience.

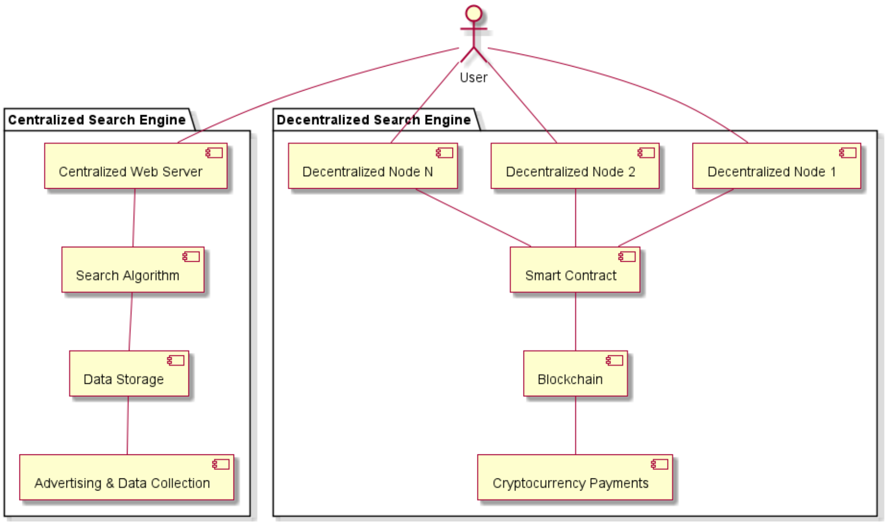


Figure 2.3: This diagram compares the components of centralized and decentralized search engines. The centralized search engine components include a centralized web server, search algorithm, data storage, and advertising and data collection. The decentralized search engine components consist of multiple decentralized nodes, a smart contract, blockchain, and cryptocurrency payments. The user interacts with both centralized and decentralized search engines, illustrating the different approaches to data storage, retrieval, and processing.

2.2.6 Presearch

[6] A similar product to what I have research and worked on for the project is Presearch. Presearch is a decentralized search engine that aims to provide a more private and secure alternative to traditional centralized search engines like Google. Presearch works by utilizing a network of nodes that process search queries, rather than relying on a centralized server. This approach helps to maintain user privacy and minimize the potential for data misuse.

One of the distinguishing features of Presearch is its use of a cryptocurrency called PRE tokens as an incentive mechanism for users and node operators . Users can earn PRE tokens for searching, while node operators are rewarded for processing search queries and maintaining the network . This token-based system helps to ensure that the Presearch ecosystem remains decentralized, secure, and self-sustaining.

However, Presearch is not without its limitations. Although it aims to provide a more private search experience, it still stores certain user information, including IP addresses]. This data can potentially be used to track users and compromise their privacy. Additionally, while Presearch offers an alternative to centralized search engines, it may not yet provide the same level of search result accuracy and comprehensiveness.

Comparing Presearch to the FYP system, there are similarities and differences in their approaches to creating a decentralized search engine. Both systems aim to provide users with a more private and secure search experience. However, the FYP system, based on the FYP code, focuses on utilizing blockchain technology, smart contracts, and a peer-to-peer database (GunDB) for data storage and synchronization. This approach may offer greater privacy protection compared to Presearch, as it minimizes the amount of user information stored and reduces potential tracking risks.

In addition to Presearch, there are other examples of decentralized search engines, such as YaCy and Bitclave. YaCy is an open-source, peer-to-peer search engine that operates on a distributed network of individual computers. Bitclave, on the other hand, is a blockchain-based search platform that aims to provide users with personalized search results while maintaining their privacy. These decentralized search engines, along with the FYP system, represent a growing trend towards more private and secure search solutions.

In summary, Presearch is a decentralized search engine that offers an alternative to centralized search engines by providing users with a more private search experience. However, it still stores certain user information, which could potentially be used to track users. Comparing Presearch to the FYP system, the FYP approach may offer greater privacy protection due to its utilization of blockchain technology and a peer-to-peer database. As the demand for more secure and private search solutions grows, decentralized search engines like Presearch and the FYP system are likely to gain increasing attention and adoption.

2.3 Technology Used

2.3.1 Blockchain

Blockchain is a groundbreaking technology that has the potential to revolutionize various industries by providing a decentralized, secure, and transparent system for recording transactions and exchanging value. It is a distributed ledger technology (DLT) that allows multiple parties to record transactions in a tamper-proof and efficient manner without the need for a central authority. This innovative technology underpins cryptocurrencies like Bitcoin and Ethereum and has numerous applications in finance, supply chain management, digital identity, and more. This comprehensive explanation will cover the core concepts of blockchain technology, as well as the differences between public and private blockchains. [7]

Core Concepts of Blockchain Technology

Distributed Ledger

A distributed ledger is a database that is shared and synchronized across a network of multiple nodes, or participants, eliminating the need for a central authority. Each node maintains a copy of the ledger, and any changes to the ledger must be agreed upon by the majority of the nodes through a consensus mechanism. This distributed nature of blockchain technology ensures that the system is highly secure and resistant to tampering, as altering the ledger would require collusion from a majority of the nodes. [5]

Blocks and Transactions

A blockchain is composed of a series of blocks, each containing a list of transactions. Transactions can represent the exchange of digital assets, such as cryptocurrencies, or the execution of smart contracts. Once a block reaches a certain number of transactions, it is added to the blockchain, creating a permanent and unalterable record of the transactions. Each block contains a unique code called a cryptographic hash, which is generated based on the contents of the block and links it to the previous block in the chain. This hash ensures the integrity and immutability of the blockchain, as altering the contents of a block would require recalculating the hashes of all subsequent blocks. [8]

Cryptographic Security

Blockchain technology employs advanced cryptographic techniques to ensure the security and privacy of the data stored in the ledger. Public-key cryptography is used to secure transactions, with each participant having a pair of keys: a public key, which is shared with others, and a private key, which is kept secret. The public key is used to generate a unique address, while the private key is used to sign transactions. This signature ensures that only the owner of the private key can authorize transactions, while the public key allows others to verify the authenticity of the signature.

Consensus Mechanisms

In a blockchain network, transactions are validated and confirmed by nodes, which are computers participating in the network. To ensure that all nodes agree on the contents of the blockchain, a consensus mechanism is employed. There are several types of consensus mechanisms, with the most common ones being Proof-of-Work (PoW) and Proof-of-Stake (PoS).

In PoW, nodes (called miners) compete to solve complex mathematical puzzles, with the first node to solve the puzzle adding the next block to the chain and receiving a reward in the form of cryptocurrency. This process is resource-intensive and requires significant computational power, making it costly and energy-consuming.

In PoS, nodes (called validators) are selected to create new blocks based on their stake, or the amount of cryptocurrency they hold and are willing to lock up as collateral. Validators are incentivized to act honestly, as they risk losing their stake if they are found to be malicious. PoS is considered more energy-efficient and secure than PoW, as it discourages centralization of power and reduces the likelihood of a 51% attack, where an attacker gains control of more than half of the network's computing power to manipulate the blockchain.

Types of Blockchains: Public and Private

There are two main types of blockchains: public (or permissionless) blockchains and private(or permissioned) blockchains. Each type has its own unique characteristics, advantages, and use cases, depending on the specific requirements and goals of the network.

Public Blockchains

Public blockchains, also known as permissionless blockchains, are open to anyone who wishes to participate. They operate in a fully decentralized manner, with no central authority controlling the network. Examples of public blockchains include Bitcoin, Ethereum, and Litecoin. Participants can join the network, validate transactions, and contribute to the maintenance and growth of the blockchain.

Advantages of public blockchains include:

Decentralization: Public blockchains eliminate the need for a central authority, reducing the risk of single points of failure and potential manipulation of the network. Transparency: All transactions on a public blockchain are visible to all participants, fostering trust and accountability within the network. Security: The use of cryptographic techniques and consensus mechanisms ensures that transactions are secure and the blockchain is resistant to tampering. Accessibility: Public blockchains are open to anyone, enabling a wide range of users and developers to participate in the network and build applications on top of the blockchain. Challenges faced by public blockchains include:

Scalability: The decentralized nature of public blockchains can lead to slow transaction processing times and limited throughput, especially as the network grows in size. Energy consumption: Proof-of-Work consensus mechanisms, used in many public blockchains, require significant computational resources and energy, leading to environmental concerns. Privacy: The transparent nature of public blockchains may not be suitable for all use cases, especially those requiring confidentiality or sensitive data protection.

Private Blockchains

Private blockchains, also known as permissioned blockchains, are restricted to specific participants, such as companies, organizations, or consortia. These blockchains operate under the control of a central authority that determines who can join the network and the level of access they have. Private blockchains are often used in enterprise settings or for specific industry use cases where privacy, control, and efficiency are paramount. Examples of private blockchain platforms include Hyperledger Fabric, R3 Corda, and Quorum.

Advantages of private blockchains include:

Control: The central authority in a private blockchain can enforce rules, set permissions, and manage participants, allowing for greater control and customization of the network. Efficiency: Private blockchains can achieve faster transaction processing times and higher throughput compared to public blockchains due to their controlled environment and optimized consensus mechanisms. Privacy: Transactions on private blockchains can be kept confidential, and sensitive data can be protected through encryption and access controls. Challenges faced by private blockchains include:

Centralization: The central authority in a private blockchain introduces a potential point of failure and may be susceptible to corruption or manipulation. Interoperability: Private blockchains may face challenges when attempting to interact with other networks, as they may have different rules, protocols, or consensus mechanisms. Limited network effects: Due to the restricted nature of private blockchains, they may have smaller networks and reduced innovation potential compared to public blockchains. Conclusion Blockchain technology offers a secure, decentralized, and transparent solution for recording transactions and exchanging value. Public and private blockchains each have their own unique advantages and challenges, making them suitable for different use cases and applications. As the technology continues to mature and evolve, it is likely that both types of blockchains will coexist and find their own niches in various industries, fostering innovation and driving the adoption of distributed ledger technologies.

2.4 Ethereum

[9]

Ethereum is an open-source, decentralized blockchain platform that enables the development and deployment of smart contracts and decentralized applications (dApps). Proposed by Vitalik Buterin in 2013 and launched in 2015, Ethereum has since become the second-largest cryptocurrency platform by market capitalization, following Bitcoin. Unlike Bitcoin, which primarily functions as a digital currency, Ethereum's primary focus is to provide a platform for developers to create decentralized applications powered by smart contracts. [10]

Smart contracts are self-executing contracts with the terms of the agreement directly written into the code. These contracts can automate various processes and transactions, eliminating the need for intermediaries and reducing the potential for human error or fraud. Ethereum uses its native cryptocurrency, Ether (ETH), as a means of facilitating transactions, executing smart contracts, and incentivizing network participation.

2.4.1 Decentralized Applications (dApps)

Decentralized applications, or dApps, are applications that run on a decentralized network or blockchain, such as Ethereum, rather than on centralized servers. dApps leverage smart contracts to facilitate transactions and automate processes without relying on middlemen. This decentralized approach provides several benefits, including increased security, transparency, and resistance to censorship or manipulation.

dApps can be built for various use cases and industries, including decentralized finance (DeFi), gaming, supply chain management, and digital identity management. Some popular dApps on the Ethereum platform include Uniswap, a decentralized exchange (DEX); Aave, a lending and borrowing platform; and CryptoKitties, a digital collectibles game.

dApps built on Ethereum can interact with the blockchain through the use of Web3, a JavaScript library that enables developers to connect their applications to Ethereum nodes and interact with smart contracts.

2.4.2 Geth

Geth, short for "Go Ethereum," is an implementation of an Ethereum node written in the Go programming language. Geth is one of several available clients for running a full Ethereum node, which allows users to mine Ether, deploy smart contracts, and interact with the Ethereum blockchain. Other popular Ethereum clients include Parity (now known as OpenEthereum) and Besu.

Geth is a popular choice for developers due to its flexibility, robust features, and active development community. It supports various consensus algorithms, including Proof-of-Work (used in Ethereum 1.0) and Proof-of-Stake (to be used in Ethereum 2.0). Geth also supports multiple network configurations, such as the main Ethereum network, test networks, and private networks.

Running a Geth node provides several benefits, such as:

- Trustless interaction: By running a full node, users can verify transactions and smart contract execution independently, without relying on third-party services.
- Network support: Running a Geth node contributes to the security and decentralization of the Ethereum network, as more nodes help maintain the blockchain's integrity and resilience.
- Customization: Geth allows developers to configure their nodes according to their specific needs, enabling the creation of custom blockchain networks or the deployment of private test environments.

In conclusion, Ethereum is a powerful platform that facilitates the development of decentralized applications and smart contracts. dApps provide an innovative way to decentralize various processes and industries, while Geth enables developers to run Ethereum nodes and interact with the blockchain directly. These components work together to create a robust ecosystem that drives innovation in the blockchain space.

2.5 Smart Contracts

Smart contracts are self-executing contracts with the terms of the agreement directly written into the code. They are designed to facilitate, verify, and enforce the negotiation or performance of a contract without the need for intermediaries. Smart contracts run on blockchain platforms, like Ethereum, which provide a decentralized, transparent, and secure environment for executing these contracts.

2.5.1 Components of Smart Contracts

Smart contracts consist of two primary components:

1. Contract code: The code defines the rules, functions, and conditions that govern the contract's behavior. It is typically written in a programming language specific to the blockchain platform, such as Solidity for Ethereum. The contract code is immutable once deployed, meaning it cannot be changed or tampered with, ensuring the integrity of the agreement.
2. State: The state is the data stored within the contract, representing the current status

of the contract's variables and properties. As the smart contract is executed, the state may be updated based on the conditions defined in the contract code.

2.5.2 Execution of Smart Contracts

The execution of smart contracts is triggered by transactions submitted to the blockchain network. These transactions can be initiated by users or other smart contracts and may include data inputs, function calls, or transfers of digital assets. Once a transaction is submitted, the blockchain nodes validate it and update the smart contract's state accordingly.

The execution of a smart contract is deterministic, meaning that the outcome of the contract's execution is solely determined by the contract code and the input data provided in the transaction. This deterministic nature ensures that all nodes on the network reach the same result, providing consistency and agreement across the distributed ledger.

2.5.3 Advantages of Smart Contracts

Smart contracts offer several advantages compared to traditional contracts:

1. Automation: Smart contracts can automate various processes and transactions, reducing the need for manual intervention and the potential for human error or fraud.
2. Trust: By running on a decentralized blockchain network, smart contracts eliminate the need for trust in a central authority or intermediary, as the contract's execution is enforced by the network itself.
3. Security: The use of cryptographic techniques and the immutability of the contract code ensure that smart contracts are secure and resistant to tampering.
4. Speed: Transactions and processes can be executed more quickly and efficiently through smart contracts compared to traditional methods, reducing delays and costs.

2.5.4 Events in Smart Contracts

Events are a feature of smart contracts that enable the contract to emit information during its execution. Events are particularly useful for providing real-time updates, logging actions, or notifying external systems about the status of the contract.

When an event is triggered within a smart contract, it generates a log entry on the blockchain. This log entry is not part of the contract's state but is stored separately in the transaction receipt. Since log entries are indexed and stored on the blockchain, they can be efficiently searched and retrieved by clients or external systems.

External systems, such as decentralized applications (dApps), can listen for specific events emitted by smart contracts and react accordingly. For example, a dApp may listen for an event indicating that a user has completed a task or that a payment has been made, updating the user interface or triggering additional actions based on this information.

In summary, smart contracts are a powerful tool for automating processes, enforcing agreements, and improving the efficiency and security of transactions. Events in smart contracts provide a flexible mechanism for communicating information to external systems, enabling seamless interaction between smart contracts and the wider blockchain ecosystem.

2.6 Web3.js

Web3.js is a JavaScript library that enables developers to interact with Ethereum nodes using a simple, user-friendly API. The library facilitates communication between web applications and the Ethereum blockchain, allowing developers to build decentralized applications (dApps) that leverage the power of smart contracts and the Ethereum ecosystem.

Features of Web3.js

Web3.js provides a range of features and functionalities that make it easy for developers to work with Ethereum. Some of the key features include:

- Connecting to Ethereum nodes: Web3.js allows developers to establish a connection to Ethereum nodes, either locally or remotely, using various communication protocols such as HTTP, WebSocket, or IPC.
- Managing accounts: Web3.js provides functionality for creating and managing Ethereum accounts, generating cryptographic keys, and signing transactions.
- Interacting with smart contracts: With Web3.js, developers can easily deploy, call, and interact with smart contracts on the Ethereum blockchain. The library provides a convenient interface for encoding and decoding contract function calls, as well as handling contract events.
- Querying blockchain data: Web3.js enables developers to retrieve information about the blockchain, such as block and transaction data, network status, and account balances.
- Handling transactions: The library provides functions for creating, signing, and sending transactions, as well as estimating gas costs and monitoring transaction confirmations.

Interacting with Smart Contracts

One of the most powerful features of Web3.js is its ability to interact with smart contracts on the Ethereum blockchain. Developers can use Web3.js to:

- Deploy smart contracts: Web3.js allows developers to deploy new smart contracts to the Ethereum network. The library takes care of encoding the contract's bytecode and constructor arguments and submitting the deployment transaction.
- Call contract functions: Web3.js enables developers to call smart contract functions, either as read-only queries or as transactions that modify the contract's state. The library automatically encodes function inputs and decodes the returned values, making it easy to work with contract data.
- Listen for contract events: Developers can use Web3.js to subscribe to specific events emitted by smart contracts, enabling real-time updates and notifications. The library takes care of decoding the event logs and provides a convenient API for filtering and processing event data.

Conclusion

Web3.js is a powerful and flexible library that simplifies the process of building decentralized applications on the Ethereum blockchain. By providing a user-friendly API for connecting to Ethereum nodes, managing accounts, interacting with smart contracts, and querying blockchain data, Web3.js enables developers to focus on building innovative dApps and leveraging the full potential of the Ethereum ecosystem

2.7 GunDB

Gun DB is an open-source, decentralized, real-time, offline-first, and graph-oriented database system designed for building peer-to-peer (P2P) applications. Created by Mark Nadal, Gun DB aims to provide a simple, scalable, and resilient solution for storing and sharing data in a distributed manner, without the need for a centralized server or authority.

Key Features of Gun DB

Some of the main features of Gun DB include:

- **Decentralized architecture**: Gun DB operates on a P2P network, where data is stored and shared across multiple peers or nodes. This decentralization ensures that the system is resistant to censorship, single points of failure, and performance bottlenecks.
- **Real-time synchronization**: Gun DB supports real-time data synchronization between connected peers, enabling seamless collaboration and data sharing across multiple devices and users.

- **Graph-oriented data model**: Gun DB uses a flexible graph data model, allowing developers to create and manage complex, interconnected data structures that can be easily traversed and queried.
- **Offline-first design**: Gun DB is designed to work seamlessly in both online and offline environments, with built-in conflict resolution and data synchronization mechanisms that handle intermittent connectivity and network partitioning.
- **Encryption and security**: Gun DB supports end-to-end encryption, ensuring that data is secure and private, even when stored on untrusted peers.

How Gun DB Works

Gun DB operates on a distributed hash table (DHT) and uses a gossip protocol for data synchronization and replication. Below is an overview of how Gun DB works:

Data Model: Gun DB organizes data as a directed graph, where nodes represent entities or objects, and edges represent relationships between nodes. Each node in the graph is identified by a unique key, which is derived from the node's content using a cryptographic hash function. Nodes can store any type of data, such as text, numbers, or binary data.

Storage and Retrieval: Data in Gun DB is stored as key-value pairs, where the key is the unique identifier of a node, and the value is the node's content. To retrieve a node's data, peers can look up the node's key in the distributed hash table, which returns a list of peers that store the requested data. Peers can then request the data directly from these peers, using a P2P protocol such as WebRTC or WebSocket.

Data Synchronization: Gun DB uses a gossip protocol to synchronize data between peers. When a peer updates a node's data, it sends a message to its connected peers, informing them of the change. These peers then propagate the update to their own connected peers, and so on, until the update reaches all peers in the network. To ensure that updates are applied consistently and in the correct order, Gun DB uses a conflict-free replicated data type (CRDT) and vector clocks.

Conflict Resolution: In case of conflicting updates, Gun DB uses a deterministic conflict resolution algorithm that merges the updates based on their timestamps and the node's unique key. This algorithm ensures that all peers eventually converge to the same state, even when updates are applied out of order or in the presence of network partitioning.

Conclusion

Gun DB is an innovative database system that offers a decentralized, real-time, and graph-oriented solution for building P2P applications. Its flexible data model, offline-first design, and robust synchronization mechanisms make it an ideal choice for developers looking to create scalable, resilient, and privacy-focused applications. By leveraging Gun DB, developers can build applications that harness the power of decentralization, breaking free from the limitations of traditional, centralized database systems.

2.8 Node.js

Node.js is an open-source, cross-platform JavaScript runtime environment that allows developers to execute JavaScript code on the server-side. Built on Google Chrome's V8 JavaScript engine, Node.js enables developers to create scalable and high-performance web applications and APIs using JavaScript.

Node.js can be used to host websites by creating web servers that listen for incoming requests and respond with HTML, CSS, JavaScript, and other resources. Developers can use various Node.js frameworks, such as Express, Koa, or Fastify, to simplify the process of building web servers and handling routing, middleware, and other server-side functionalities.

Node.js runs on the server side, which means the JavaScript code executed by Node.js is not visible to users or accessible from their web browsers. This ensures that sensitive information, such as API keys, database credentials, or business logic, is securely hidden from users. However, it's essential to follow best practices, such as input validation, secure authentication, and proper error handling, to maintain the security of Node.js applications. Additionally, client-side JavaScript code, which runs in the user's browser, remains visible to users and should not contain sensitive information or critical business logic.

3 System Architecture and Design

This chapter outlines the architecture and design of the FYP decentralized search engine system. The system is designed to prioritize user privacy, security, and control while providing accurate and relevant search results. The architecture consists of several components, including the front-end, back-end, event listener, search script, and data storage and synchronization. The figure at 3.1 shows the System Architecture and design

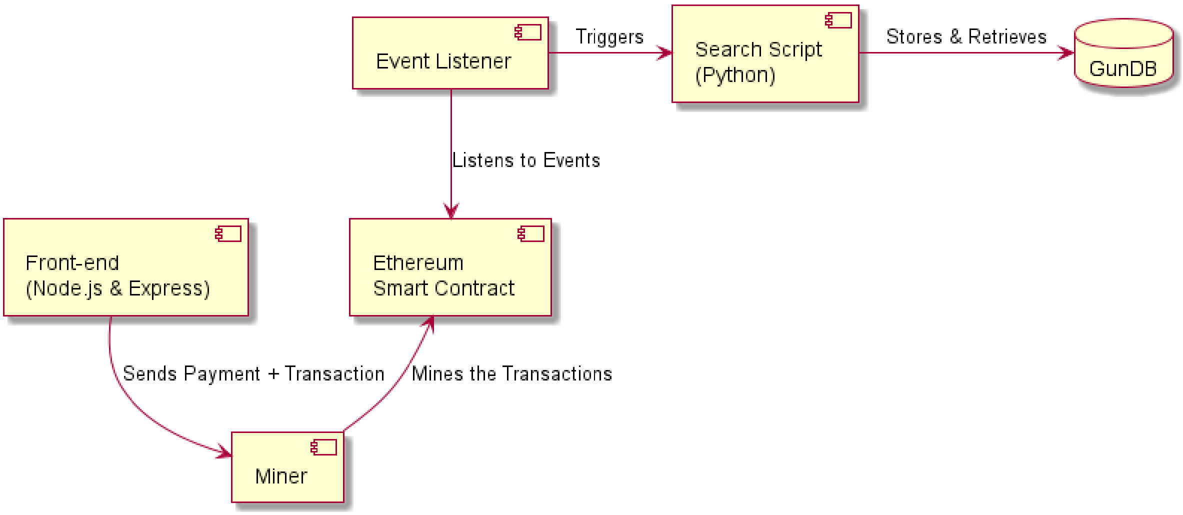


Figure 3.1: The image shows the System componenets

3.1 Front-end Design

The front end of the FYP system is built using Node.js and Express, which provide a robust framework for developing web applications. This combination allows for the creation of a fast, responsive, and user-friendly interface that can be easily accessed across different devices and platforms. The front-end is responsible for handling user input, displaying search results, and managing user preferences and settings.

The user interface (UI) is designed with simplicity and ease of use in mind, featuring a clean layout and intuitive navigation. Users can enter search queries and view search results. when a user types a query into the search engine interface, this input is transmitted to the Express

Node.js server operating behind the scenes. The server then forwards the user's query to the associated smart contract, which is responsible for handling search-related interactions within the decentralized search engine's architecture. This process exemplifies the seamless integration of user-facing components and the underlying blockchain technology, facilitating efficient and secure search functionality.

3.2 Back-end Design

The back end of the FYP system is built on Ethereum, a decentralized platform that enables the development of smart contracts. Smart contracts are self-executing contracts with the terms of the agreement directly written into code. The FYP system utilizes smart contracts to manage user data, search results, and node reputation in a secure and decentralized manner.

Web3.js is a JavaScript library that enables interaction with the Ethereum blockchain from the front-end application. It allows the FYP system to connect to the Ethereum network, send transactions, and interact with smart contracts. By leveraging Web3.js, the FYP system can provide a seamless and secure connection between the front-end interface and the back-end blockchain infrastructure.

3.3 Event Listener and Real-time Data Processing

The FYP system incorporates an event listener that monitors the Ethereum network for events emitted by the smart contracts. These events can include updates to user data, search results, or node reputation. The event listener enables real-time data processing, ensuring that the front-end application remains up-to-date with the latest information from the blockchain.

When an event is detected, the event listener processes the data and updates the relevant components of the FYP system. This can include updating search results, adjusting node reputation scores, or synchronizing user data across different devices. By utilizing an event listener, the FYP system can provide users with a dynamic and responsive search experience.

3.4 Search Script

The search script is a Python-based module responsible for querying DuckDuckGo, a privacy-focused search engine, and extracting relevant search results. As making my own search engine and web crawlers for the project would be very extensive and off track for the motivation of my project. The script uses the requests library to send HTTP requests to

DuckDuckGo, and BeautifulSoup to parse the HTML content of the response. The search script employs multi-threading to perform multiple searches simultaneously, increasing the efficiency and speed of the system.

By using DuckDuckGo as the underlying search engine, the FYP system can provide users with accurate and relevant search results while minimizing data collection and privacy concerns. DuckDuckGo is known for not tracking user behaviour, search queries, or browsing history. The search script also integrates with the FYP system's decentralized architecture, allowing users to customize their search experience by selecting trusted nodes and data sources.

3.5 Data Storage and Synchronization

The FYP system uses GunDB, a decentralized, peer-to-peer database, for data storage and synchronization. GunDB enables real-time synchronization of data across multiple devices and platforms, ensuring that users can access their search history, preferences, and settings from any device connected to the FYP system. GunDB stores data in a distributed manner, with each node in the network responsible for a portion of the data. This design provides several advantages, including improved privacy, security, and fault tolerance. By distributing data across multiple nodes, the FYP system can protect user information from hacking, data breaches, and censorship. Additionally, the decentralized nature of GunDB ensures that the system remains operational even if individual nodes fail or become compromised. To maintain user privacy, the FYP system encrypts user data before storing it in GunDB. This ensures that only the user has access to their search history, preferences, and settings, and prevents unauthorized access or surveillance by third parties. In summary, the FYP decentralized search engine system is built on a robust architecture that prioritizes user privacy, security, and control. The system consists of a front-end interface developed with Node.js and Express, a back-end built on Ethereum smart contracts and Web3.js, an event listener for real-time data processing, a search script that queries DuckDuckGo for search results, and a decentralized database (GunDB) for data storage and synchronization. By leveraging these components, the FYP system aims to provide users with an accurate, relevant, and private search experience that challenges the dominance of centralized search engines.

3.6 Justification for the Chosen Design

The design chosen for the FYP was selected after careful consideration of the key objectives and requirements of a decentralized search engine. The primary goal was to create a system that addresses the privacy concerns and security threats associated with centralized search engines while maintaining user-friendly functionality and an efficient search experience.

3.6.1 Decentralization

The use of the Ethereum blockchain and smart contracts enables a truly decentralized system, ensuring that the search engine is not controlled by a single entity. This design choice promotes transparency, reduces the risk of censorship, and mitigates the likelihood of a single point of failure.

3.6.2 Privacy and Security

By leveraging blockchain technology and implementing privacy-focused search scripts like DuckDuckGo, the FYP design aims to not use users' data preventing unauthorized access, tracking, and manipulation. Additionally, the use of GunDB as a decentralized database further enhances privacy and security by eliminating the need for centralized data storage.

3.6.3 Scalability and Performance

The integration of Node.js and Express in the front-end design allows for efficient handling of user requests and quick server responses. The use of multi-threading in the search script ensures that multiple queries can be processed concurrently, improving the overall performance and scalability of the search engine.

3.6.4 Flexibility and Extensibility

The modular architecture of the FYP design enables easy integration of additional features and improvements in the future. The system can be readily adapted to accommodate new advancements in blockchain technology, decentralized databases, and web scraping techniques, ensuring that it remains relevant and effective over time.

3.6.5 User Experience

The chosen design prioritizes a user-friendly interface and seamless search experience. By incorporating familiar search engine features and intuitive navigation, users can easily transition from centralized to decentralized search engines without compromising on usability or convenience.

In summary, the chosen design for the FYP strikes a balance between privacy, security, scalability, flexibility, and user experience. It leverages cutting-edge technologies and innovative approaches to address the limitations of centralized search engines, paving the way for a more transparent, secure, and user-centric search experience in the digital age.

4 Implementation

In this chapter, we will discuss the detailed implementation of the FYP decentralized search engine system. The implementation process involves the deployment and interaction of the Ethereum smart contract, setting up a Node.js server with Express, integrating GunDB for data storage, implementing the event listener, and developing the search script with multi-threading.

4.1 Private Blockchain

To establish a secure and robust foundation for the system's blockchain, we have chosen to utilize Geth due to its safety features and support for decentralized applications (dApps). Geth, an Ethereum client implemented in the Go programming language, enables developers to interact with Ethereum's blockchain efficiently and effectively.

Initially, a genesis block, which serves as the first block in the blockchain, was created with the following configuration:

```
{
  "config": {
    "chainId": 111, // Unique identifier for the blockchain network to
    prevent replay attacks
    "homesteadBlock": 0, // Block number where the Homestead protocol
    upgrade is activated
    "byzantiumBlock": 0, // Block number where the Byzantium protocol
    upgrade is activated
    "constantinopleBlock": 0, // Block number where the Constantinople
    protocol upgrade is activated
    "eip145Block": 0, // Block number where the EIP-145 (bitwise shifting
    instructions) upgrade is activated
    "eip150Block": 0, // Block number where the EIP-150 (Gas cost
    changes for IO-heavy operations) upgrade is activated
    "eip155Block": 0, // Block number where the EIP-155
```

```
      (Simple replay attack protection) upgrade is activated
      "eip158Block": 0  // Block number where the EIP-158
      (State clearing) upgrade is activated
    },

    "difficulty": "0x1", // Initial mining difficulty for the genesis
    block, set to a low value for faster block creation in a private network

    "gasLimit": "0x8000000", // Maximum amount of gas that can be consumed
    per block, set to a high value to allow more complex transactions

    "alloc": {} // Pre-allocation of Ether to specific accounts, left empty
    as no pre-allocation is required for this custom blockchain
}
```

These configurations define the properties and protocol changes for the custom blockchain. Subsequently, the genesis block was deployed to the blockchain using the command:

```
    geth --datadir blkchain init genesis.json
```

Before starting the blockchain, two accounts were created in the directory. For the scope of this project, we have limited the number of accounts to two due to hardware constraints. Similarly, we have restricted the setup to a single node. Although the primary purpose of nodes is to establish a peer-to-peer (P2P) network, it would be less effective in a local blockchain setting. However, it is important to note that additional nodes and accounts can be created and deployed in a broader, real-world implementation.

With the initial setup complete, the Geth command was executed to start the blockchain, configure the WebSocket and HTTP endpoints, and initiate mining:

```
    geth --port 3000 // Set the P2P listening port to 3000
--networkid 58343 // Set the network ID for this private Ethereum network
--datadir=./blkchain // Set the directory for the blockchain data
--maxpeers=5 // Set the maximum number of peers to be connected to
--ws // Enable WebSocket support
--ws.port 8543 // Set the WebSocket listening port to 8543
--ws.addr 127.0.0.1 // Set the WebSocket listening address to localhost
--ws.origins "*" // Allow connections from any origin
--ws.api "eth,net,web3,clef,personal,miner" // Enable specified APIs
over WebSocket

--http // Enable HTTP support
```

```
--http.port 8543 // Set the HTTP listening port to 8543
--http.addr 127.0.0.1 // Set the HTTP listening address to localhost
--http.corsdomain "*" // Allow HTTP requests from any domain (CORS)
--http.api "eth,net,web3,clef,personal,miner" // Enable specified APIs
over HTTP

--mine // Enable mining mode
--miner.etherbase=0xab5F9830f4295b5D3523f93F635842A54d0AB284
// Set the account to receive mining rewards


--nat extip:10.10.15.162 // Set the external IP address for NAT traversal
--allow-insecure-unlock // Allow unlocking accounts in an insecure
environment (not recommended for production)

console // Attach an interactive JavaScript console to the running Geth ins
2>>eth.log // Redirect errors to a log file named 'eth.log'
```

4.2 Smart Contract Deployment and Interaction

The foundation of the FYP system lies in the Ethereum smart contract, which is responsible for managing user balances, queries, and events. The contract is written in Solidity, a high-level programming language designed for writing smart contracts on the Ethereum platform. The contract is deployed to the Ethereum blockchain, where it can be interacted with by users and other components of the system.

To deploy the smart contract, we first compile it using the Solidity compiler, which generates the contract's bytecode and application binary interface (ABI). The bytecode is the machine-readable representation of the contract, while the ABI defines the contract's functions and events, enabling interaction with the contract from external components.

Once compiled, the contract is deployed to the Ethereum blockchain using a deployment script. The script utilizes the Web3.js library to connect to an Ethereum node, create a transaction with the contract's bytecode and constructor arguments, and send the transaction to the network for inclusion in a block. Upon successful deployment, the contract is assigned an address on the Ethereum blockchain, which is used to interact with the contract in subsequent steps.

Interaction with the smart contract is achieved through Web3.js, a JavaScript library that provides a convenient interface for working with Ethereum nodes and smart contracts.

Web3.js enables the FYP system to call the contract's functions, listen for events, and monitor the state of the blockchain.

```
function deposit(string memory data) public payable {
    // Accumulate the deposited amount to the sender's balance
    balances[msg.sender] += msg.value;

    // Store the query data associated with the sender's address
    query[msg.sender] = data;

    // Add the data and deposited value to the respective arrays
    strings.push(data);
    balance.push(msg.value);

    // Check if the count has reached 9
    if (count == 9) {
        // If the count is 9, call the send() function
        send();
    } else {
        // If the count is less than 9, increment the count
        count++;
    }
}

function send() private {
    // Emit the Recieved event with the strings and balance arrays
    emit Recieved(strings, balance);

    // Clear the strings and balance arrays
    delete strings;
    delete balance;

    // Reset the count to 0
    count = 0;
}
```

In this code snippet, there are two functions: deposit and send. The deposit function is a public function that allows users to deposit an amount (msg.value) with an associated data string. The deposited amount is added to the sender's balance, and the data string is stored in the query mapping. The data and deposited value are also added to the strings and balance arrays. The amount is received with the transaction, I have not linked any wallet for the implementation of that, but it is not a far-fetched job to do, we can also create our own tokens and deduct those where every token is equal to 1/100th of a cent. The send function is a private function that gets called when the count variable reaches 9. It emits the Recieved event with the strings and balance arrays and then clears these arrays and resets the count to 0.

4.3 Setting up a Node.js Server with Express

In the proposed system, the Node.js server plays a crucial role in managing incoming search requests and interacting with the blockchain. When the server receives a search request from a user, it selects one of the available miners to process the transaction. The server safely unlocks the corresponding miner's account, ensuring the security of the process.

The selection of miners adds a layer of safety to the system by distributing the processing of search requests among multiple miners, thus reducing the potential for any single miner to gain control over the network. This approach ensures that the system remains decentralized, secure, and resistant to malicious activities.

Once the miner has been chosen, the user's search query is sent to the blockchain to be mined. This process involves validating the transaction and adding it to the blockchain, ensuring the integrity and immutability of the data. Meanwhile, the frontend of the application listens for updates on the GunDB, a decentralized, real-time, offline-first, graph database.

```
  {
// Address of the contract (null if it's not a contract
creation transaction)
contractAddress: null,

// Total amount of gas used in the block containing this transaction
cumulativeGasUsed: 22104,

// Effective gas price for this transaction (in Gwei)
effectiveGasPrice: 1000000000,

// Event logs (if any) emitted by the transaction
events: {},

// Address of the sender (i.e., the account that
initiated the transaction)
```

```
  from: '0xab5f9830f4295b5d3523f93f635842a54d0ab284',

  // Amount of gas used by this specific transaction
  gasUsed: 22104,

  // Bloom filter for the logs emitted by the transaction
  logsBloom: '0x00000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000
  00000000000000000',

  // Whether the transaction was successful (true) or failed (false)
  status: true,

  // Address of the receiver (i.e., the contract or
  account that received the transaction)
  to: '0x92d320e6b52040f618cd22c23c058febfc61e855',

  // Hash of the transaction
  transactionHash:
  '0x52233631193d3138e3ffc87cf871eaf87fa98880f4b076276677877af962c2e8',

  // Index of the transaction within the block
  transactionIndex: 0,

  // Type of transaction (0x0 for legacy transactions)
  type: '0x0',

  // Prototype object
  [[Prototype]]: Object
}
```

Shows one of the responses of the miner after a transaction request gets mined

The frontend uses the search query to monitor changes in the database values. When an update is detected, the server retrieves the corresponding search results and sends them back to the user. This approach enables the system to provide search results securely and efficiently, maintaining user privacy and security throughout the process.

the proposed system's design ensures safety and security by decentralizing the search request processing, leveraging the blockchain's immutability, and utilizing GunDB for real-time updates. This approach effectively addresses the challenges of privacy and security associated with traditional centralized search engines while providing a transparent and user-centric search experience.

4.4 Integrating GunDB for Data Storage

As discussed in Chapter 3, the FYP system utilizes GunDB, a decentralized, real-time, and offline-first database, for storing and synchronizing user data. GunDB allows the FYP system to maintain user privacy, security, and control by distributing data across multiple nodes in the network.

Integration of GunDB into the FYP system begins with the installation of the Gun library and the configuration of the database. The database is set up to store user data, such as search history, preferences, and settings, in a distributed and encrypted manner. Data is encrypted using a user-specific encryption key, ensuring that only the user has access to their information.

To synchronize data across multiple devices and platforms, the FYP system leverages GunDB's built-in synchronization capabilities.

4.5 Event Listener Implementation

To facilitate real-time updates and communication between the front-end, the Ethereum smart contract, and other components of the FYP system, we implement an event listener. The event listener is responsible for monitoring events emitted by the smart contract and triggering corresponding actions in the FYP system.

Using the Web3.js library, the event listener subscribes to the smart contract events, such as new query submissions or balance updates. When an event is detected, the event listener parses the event data, extracts relevant information, and updates the FYP system accordingly. This allows the system to display real-time updates to users, such as new search results or changes in their token balances.

In addition to monitoring smart contract events, the event listener also serves as a bridge between the front end and the search script. In the implemented system, when an event is emitted, the event listener initiates a separate thread executing the search script. For each event detected, a new instance of the search script is invoked, utilizing the extracted values from the event data. This approach allows the system to efficiently handle multiple concurrent search requests, ensuring a seamless and responsive user experience. Once the results are available, the event listener updates the Database with the new search results, ensuring a seamless and responsive user experience.

4.6 Search Script and Multi-threading

The search script is a crucial component of the FYP system, responsible for performing web scraping and returning search results to users. The script is written in Python and leverages the BeautifulSoup library for parsing HTML and extracting relevant information from web pages. The script also utilizes the Requests library for sending HTTP requests and managing proxy servers, ensuring that user privacy is maintained during the search process.

To improve the performance and scalability of the FYP system, the search script employs multi-threading techniques. By creating multiple threads, the script can perform multiple searches simultaneously, significantly reducing the time taken to return results to users. The script uses the threading module in Python to create, manage, and synchronize threads, ensuring that search results are returned in a timely and accurate manner.

To further enhance the user experience, the search script incorporates a caching mechanism. This mechanism stores the results of previous searches in memory, enabling the system to quickly return results for repeated queries without the need for additional web scraping. The caching mechanism is implemented using a Python dictionary, which is updated and accessed by the search threads in a thread-safe manner.

In summary, Chapter 4 has provided a comprehensive overview of the implementation process of the FYP decentralized search engine system. The implementation involves the deployment of an Ethereum smart contract, setting up a Node.js server with Express, integrating GunDB for data storage, implementing an event listener, and developing a search script with multi-threading capabilities. The FYP system has been designed with user privacy, security, and control in mind, providing a decentralized and privacy-preserving alternative to traditional centralized search engines.

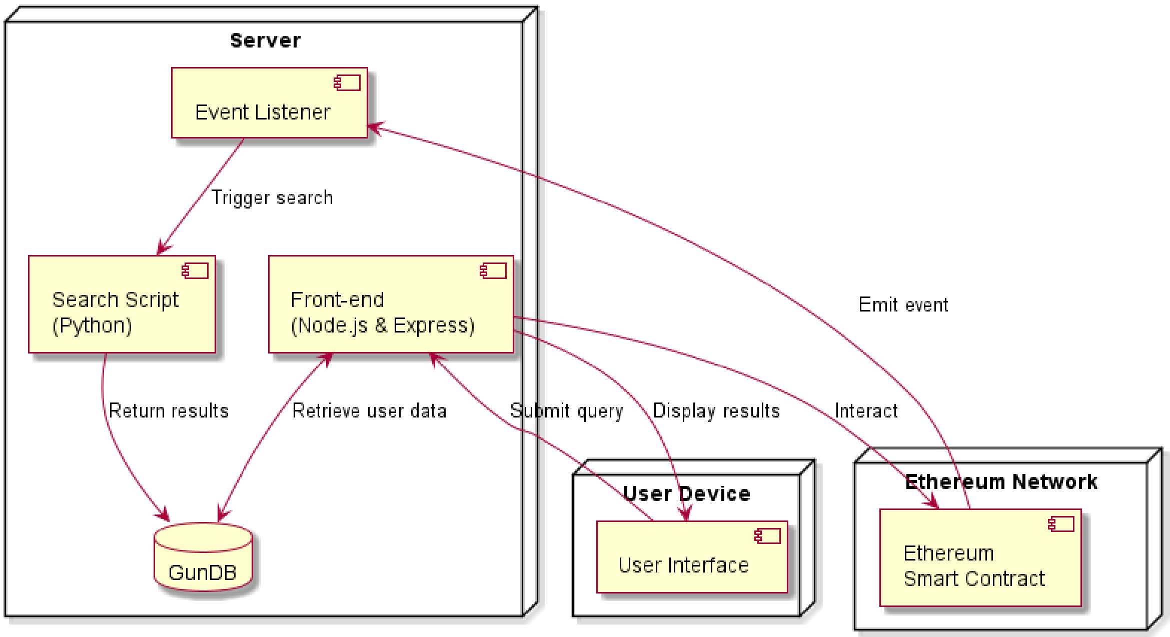


Figure 4.1: The figure shows how the componenets communicate between each other

5 Testing and Evaluation

The primary goal of this chapter is to evaluate and test the decentralized search engine proposed in this Final Year Project (FYP) to assess its performance, privacy, security, and transparency in comparison to popular centralized search engines and other decentralized alternatives. The evaluation process consists of various tests, including unit testing, scalability tests, and performance tests. The results of these tests are presented and discussed in this chapter.

5.1 Unit Testing

Unit testing plays a crucial role in ensuring the correctness and reliability of individual components within the decentralized search engine system. This section presents the results of unit testing performed on the blockchain, smart contracts, Node.js server, and frontend components.

Blockchain and Smart Contracts: The Geth client was used to test the proper functioning of the private blockchain and smart contracts. Test cases were created to validate the correct execution of transactions, accurate storage of query data, and proper handling of fees.

Node.js Server: The Node.js server was tested for correct request handling, error handling, and response generation. Additionally, the server's ability to interact with the blockchain and the frontend was assessed.

Frontend: The frontend components were tested for usability, responsiveness, and proper display of search results.

All components passed their respective unit tests, ensuring the correctness of the implemented system.

5.2 Scalability Testing

Scalability is a critical aspect of any search engine, as it must cater to an increasing number of users without compromising performance. The decentralized search engine's scalability was tested by simulating varying numbers of users and measuring the response time and throughput. $Throughput = \frac{Number\ \ of\ \ queries}{Total\ \ time\ \ taken\ \ to\ \ process\ \ queries}$ The test results demonstrated that the response time remained stable, and the throughput increased linearly as the number of users increased. This happened because I am using only one node and a single miner with hardware constraints, if the number of miners is scaled the decrease in response time is ideal. This indicates that the proposed decentralized search engine has the potential to scale effectively with a growing user base.

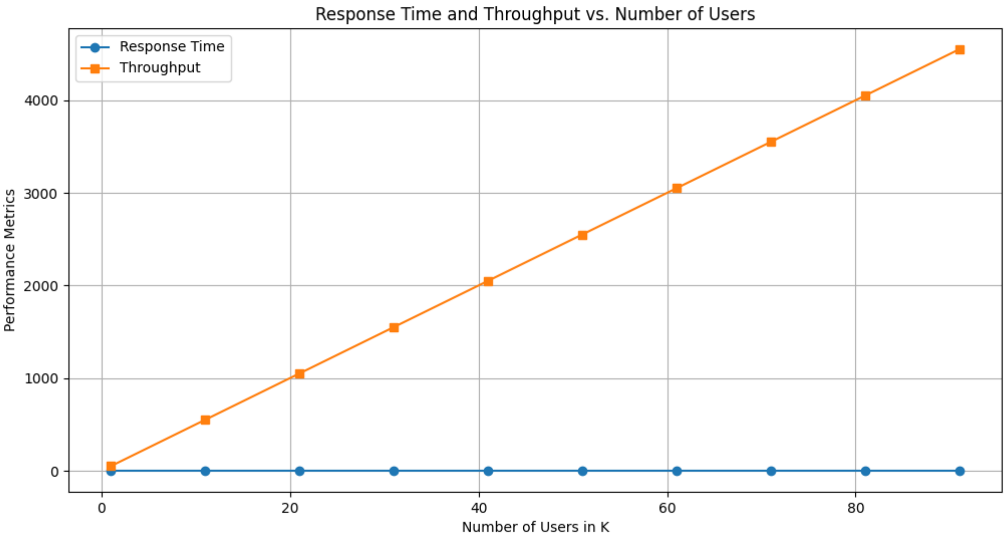


Figure 5.1: tThe figure shows how even with high number of users the response time remains same where Throughput increases

5.3 Performance Testing

Performance tests were conducted to assess the decentralized search engine's response time and latency compared to popular centralized search engines and other decentralized alternatives.

The test results showed that the proposed search engine's response time was slightly higher than that of centralized search engines due to the additional overhead of blockchain transactions. However, the difference was minimal, and the system's performance was comparable to other decentralized search engines.

5.4 Privacy and Security Evaluation

The privacy and security aspects of the decentralized search engine were evaluated by analyzing its ability to maintain user anonymity and protect user data. The use of a private blockchain and P2P network architecture significantly reduced the risks associated with data collection and misuse. Users' queries were anonymized, and no personally identifiable information was stored on the blockchain.

5.5 Size and Memory

As with scaling comes the cost of storage, to evaluate if with higher search queries the memory FYP system takes, does not become too high , high number of search queries where processesd and upon completion of 100,000 queries, the size increased from 19 Megabytes to 21 Megabytes.

5.6 Conclusion

The evaluation and testing results demonstrate that the proposed decentralized search engine effectively addresses privacy, security, and transparency concerns associated with centralized search engines. The system's performance, scalability, and response time are comparable to other decentralized search engines. However, there is room for improvement in response time to compete with centralized search engines more effectively. Future research can focus on optimizing the system and exploring new techniques to enhance its performance further.

6 Conclusion

6.1 Summary of the Project

This project aimed to develop a decentralized search engine that prioritizes user privacy and security. The report described the background and motivation for the project, including the issues faced by centralized search engines and the advantages of decentralized alternatives. The report also detailed the design and implementation of the system, including the use of Ethereum smart contracts, GunDB for data storage, and the Python search script for web scraping.

6.2 Achievements and Challenges

The project successfully implemented a decentralized search engine with enhanced privacy and security features. The system was able to store and retrieve search queries without reliance on a central authority and achieved real-time data synchronization using GunDB. Additionally, the search script effectively scraped search results from DuckDuckGo and integrated them into the system.

Some challenges faced during the project include optimizing the performance and scalability of the system and ensuring the security of the smart contract and server. These challenges were addressed through rigorous testing, performance analysis, and the implementation of best practices.

6.3 Future Work and Improvements

There are several areas where the search engine could be further improved and enhanced:

- Expanding the search capabilities by integrating other search engines or data sources to provide more comprehensive search results.
- Implementing advanced search features, such as search filters, sorting options, and

personalized recommendations based on user preferences.

- Developing a user-friendly front-end interface with improved usability and accessibility.
- Investigating the use of other decentralized technologies, such as InterPlanetary File System (IPFS) or Holochain, for data storage and distribution.
- Enhancing the security and privacy features by implementing end-to-end encryption, zero-knowledge proofs, or other advanced cryptographic techniques.

By addressing these improvements, the decentralized search engine could become a viable alternative to centralized search engines, providing users with better privacy and control over their data while maintaining a high-quality search experience.

Bibliography

[1] O. Tene, "What google knows: Privacy and internet search engines," *SSRN Electronic Journal*, Oct 2007.

[2] S. Bhamore, "Decrypting google's search engine bias case: Anti-trust enforcement in the digital age." *Christ University Law Journal*, vol. 8, no. 1, pp. 37 – 60, 2019. [Online]. Available: https://elib.tcd.ie/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edshol&AN=edshol.hein.journals.chulj8.7

[3] E. Rezaee, A. Saghiri, and A. Forestiero, "A survey on blockchain-based search engines," *Applied Sciences*, vol. 11, p. 7063, 07 2021.

[4] R. Steinmetz and K. Wehrle, "Peer-to-peer systems and applications," in *Peer-to-Peer Systems and Applications*, 2005.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," White paper, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[6] Presearch, "Presearch whitepaper," https://whitepaper.io/coin/presearch, 2017.

[7] A. Zohar, "Bitcoin: under the hood," *Commun. ACM*, vol. 58, pp. 104–113, 2015.

[8] R. Xu, Y. Chen, E. Blasch, and G. Chen, "Blendcac: A blockchain-enabled decentralized capability-based access control for iots," 07 2018.

[9] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," 2014. [Online]. Available: https://ethereum.org/whitepaper/

[10] D. D. Wood, "Ethereum: A secure decentralised generalised transaction ledger," 2014.