PROJECT REPORT

On

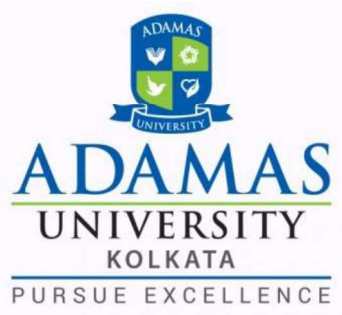
"ChainHire: A Privacy-Preserving and Transparent Job Search Portal Using an Enterprise-Level Permissioned Blockchain Framework"

Submitted in partial fulfilment of the requirements for the award of

Bachelor of Computer Applications (BCA)

In the department of

Computer Science and Engineering



Submitted by:
Satyajit Ghosh (UG/02/BCA/2020/003)

Rakibul Islam (UG/02/BCABFSI/2020/003)

Aditya Jaman (UG/02/BCA/2020/023)

Aratrika Bose (UG/02/BCA/2020/019)

Under the Guidance of

Dr. Abhishek Roy
(Professor, Dept. of Computer Science and Engineering)

School of Engineering & Technology
ADAMAS University, Kolkata, West Bengal
January - June 2023

CERTIFICATE

This is to certify that the project report entitled **"ChainHire: A Privacy-Preserving and Transparent Job Search Portal Using an Enterprise-Level Permissioned Blockchain Framework"**,
submitted to the School of Engineering and Technology (SOET), **ADAMAS UNIVERSITY**, **KOLKATA** in partial fulfilment for the completion of Semester – 6th of the degree of **Bachelor of Computer Applications** in the department of **Computer Science and Engineering**, is a record of bonafide work carried out by

Satyajit Ghosh (UG/02/BCA/2020/003),
Rakibul Islam (UG/02/BCABFSI/2020/003),
Aditya Jaman (UG/02/BCA/2020/023),
Aratrika Bose (UG/02/BCA/2020/019) under our guidance.

All help received by us from various sources has been duly acknowledged.
No part of this report has been submitted elsewhere for the award of any other degree.

Dr. Abhishek Roy
(Professor, Dept. of CSE)

Ms. Roneeta Purkayastha
(Assistant Professor, Dept. of CSE)

Dr. Sajal Saha
(HOD CSE)

Acknowledgement

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without mentioning the people whose constant guidance and encouragement made it possible. We take pleasure in presenting before you, our project, which is the result of a studied blend of both research and knowledge.

We express our earnest gratitude to **Dr. Abhishek Roy (Professor), Department of CSE**, for his constant support, encouragement, and guidance. We are grateful for his cooperation and valuable suggestions.

Finally, we express our gratitude to all other members who are involved either directly or indirectly in the completion of this project.

Declaration

We, the undersigned, declare that the project entitled **"ChainHire: A Privacy-Preserving and Transparent Job Search Portal Using an Enterprise-Level Permissioned Blockchain Framework"**, being submitted in partial fulfillment for the award of Bachelor of Computer Applications Degree in Computer Science & Engineering, affiliated to ADAMAS University, is the work carried out by us.

Satyajit Ghosh
UG/02/BCA/2020/003

Aditya Jaman
UG/02/BCA/2020/023

Rakibul Islam
UG/02/BCABFSI/2020/003

Aratrika Bose
UG/02/BCA/2020/019

Abstract

Finding a suitable employment position is a crucial and difficult task. Nowadays, Job seekers may instantly apply for multiple job openings using job search platforms. Personal information has to be exchanged as part of every job application. Sharing personal information across unsafe channels, however, increases the risk of data theft. Apart from that, attempts to tamper with hiring data are unfortunately common. Previously, traditional databases were only used for job search platforms, which do not provide sufficient transparency or protection against tampering. The incorporation of blockchain technology in job search portals can address these issues and provide a more transparent and secure process. Our proposed solution uses Hyperledger Fabric (HLF), an open-source blockchain framework, to create a secure and transparent job search platform. In this platform, both recruiters and job seekers can participate in a permissioned network, where smart contracts are used to ensure a transparent and privacy-friendly hiring process. To demonstrate the feasibility of this solution, we have implemented and deployed a prototype using Amazon Managed Blockchain Service. To understand the optimal configuration for our system, we tested the performance of our network using the Hyperledger Caliper tool. Although further research is necessary to fully understand the capabilities and limitations of using blockchain technology in job search portals.

Table of Contents

List of Figures

List of Algorithms

List of Abbreviations

Serial No.	Acronym	Definition
1	HLF	Hyperledger Fabric
2	AWS	Amazon Web Services
3	EC2	Elastic Compute Cloud
4	SSL	Secure Sockets Layer
5	HTTPS	Hypertext Transfer Protocol Secure
6	OTP	One-Time Password
7	CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
8	DNS	Domain Name System
9	CLI	Command-Line Interface
10	SDK	Software Development Kit
11	TLS	Transport Layer Security
12	TPS	Transactions Per Second

List of Publications

1. S. Ghosh, R. Islam, A. Jaman, A. Bose and A. Roy, "ChainHire: A Privacy-Preserving and Transparent Job Search Portal Using an Enterprise-Level Permissioned Blockchain Framework," *2023 International Conference on Advances in Intelligent Computing and Applications (AICAPS)*, Kochi, India, 2023, pp. 1-6, doi: 10.1109/AICAPS57044.2023.10074582.
   (Published in IEEE Xplore)

2. S. Ghosh, "CLI API in Hyperledger," *GeeksforGeeks*, 03-Jan-2023. [Online]. Available: https://www.geeksforgeeks.org/cli-api-in-hyperledger/. [Accessed: 20-Feb-2023].
   (Peer-reviewed Technical Article)

3. S. Ghosh, A. Bose, A. Jaman, R. Islam, and A. Roy, "Hyperledger Fabric: A Game-Changer for Energy, Education, Healthcare, and Beyond."
   (Manuscript Submitted)

Chapter 1

Introduction

1.1 Background

A job is a crucial aspect of a person's life, providing not only a source of income but also a sense of satisfaction, productivity, and purpose. With high levels of unemployment being a significant issue in society, job search portals play a vital role in helping job seekers find employment opportunities. At the same time, these portals assist companies in streamlining the recruitment process. Traditional job search portals often require job seekers to provide sensitive personal information, such as their name, address, and work history. Recently with each company moving to these online job sites for hiring and online job portals getting a high number of users, fraudsters also using it to their advantage. Fraudsters join these platforms and register themselves as fake recruiters and try to collect job seekers' personal information [1]. Many times it is also noticed that the personal information of the candidates is shared with third parties without consent. Apart from that, hiring ineligible candidates after removing the names of the eligible candidate's name from the record, is also come up many times [2].

The main operating principle of job search portals is as follows:

- On the platform, both recruiters and job seekers register.
- Recruiters post the job openings.
- All available positions are visible to job seekers.
- Job seekers can submit applications for one or more jobs.
- Details of the applicant were made available to recruiters.
- If a job applicant is hired, the recruiter updates the position as closed.

1.2 Objective

The decision to use blockchain technology instead of a traditional database system was made in order to safeguard the privacy of users and to enhance transparency in hiring records. In terms of Transparency, the smart contracts are designed to store job records on the blockchain ledger, and every new application and information about a new hire is recorded there. Once a candidate is hired, a private identifier(SHA256)[Secure Hash Algorithm 256-bit] of the successful candidate is shared with other registered candidates, so they can verify the hire. Job seekers can also check the history of a job record in a timeline manner as it is on the blockchain, and multiple parties need to endorse a transaction before it propagates in the network, which removes scams involved in tampering with hiring data to hire ineligible candidates after removing the names of eligible candidates, which is not possible with a traditional database-based system. In terms of Privacy, Smart contracts are utilized to store the personal data of job seekers on the blockchain ledger, which can protect their personal information from being shared with third parties by the company without consent. With the use of blockchain technology and smart contracts, job seekers can rest assured that only the recruiter to whom they have applied can only view their profile. Not even the portal admin can access their personal data, except for basic information, which is also not possible with a traditional database-based system.

1.3 Scope

Traditional job search portals often rely on centralized systems that can be vulnerable to cyber-attacks and data breaches. In contrast, blockchain technology provides a decentralized and secure platform that can protect job seekers' personal information and prevent fraud. This has led to the emergence of blockchain-based job search portals that can revolutionize the job market by providing a more secure and efficient hiring process. As more companies look for ways to protect their data and comply with privacy regulations, the market for blockchain-based job search portals is expected to grow rapidly.

Chapter 2

Literature Review

In this section, We have discussed the existing works on job search platforms and different areas where the HLF-based implementation is proposed earlier.

2.1 Existing Work on Job Search Platform

After conducting thorough research for previous work on job search platforms, we found that there is a very limited amount of research in this area present. Most of the available work is insufficient for comparison. Despite this, we attempted to compare our work with some of them. Table 2.1 compares our work with the existing ones. We have observed that the majority of the work is done using MySQL and SQL Server databases and along with that PHP is used as a backend. In our extensive search, we haven't found any work that uses any blockchain framework earlier for job search portals.

Table 2.1: A comparison of the proposed solution with the current available solutions.

Features	Our Solution	Mathisugan[3] Prodhan et. al.[4] Katariya et. al.[5]	Pinjari et. al.[6]
Blockchain Platform	Yes	No	No
Smart Contract	Yes	No	No
Decentralization	Yes	No	No
Data Privacy	Yes	No	No
Data tempering	No	Yes	Yes
Technology Used	Hyperledger Fabric	MySQL	SQL Server
Implemention	Yes	Yes	Yes
Deployment	Yes	No	No

2.2 Existing Work on Hyperledger Fabric

The year 2016 marked the debut of Hyperledger, which came with a suite of technical and organizational governance protocols and the support of 30 founding corporate members. During the incubation period, the Hyperledger Technical Steering Committee approved two blockchain framework codebases, which were IBM's OpenBlockchain and Blockstream's libconsensus. These codebases were subsequently merged by Digital Asset to form the HLF[1].

Among various blockchain platforms available today, HLF has emerged as a prominent choice for various reasons. It is an open-source, permissioned blockchain platform that provides a modular, scalable, and secure foundation for building decentralized applications suitable for diverse use cases. It enables greater trust, transparency, and tamper-resistance among the various stakeholders involved, ensuring data privacy and accountability. We analyzed existing research papers, technical reports, whitepapers, and case studies, focusing on HLF applications across the different sectors.

2.2.1 Energy Sector

Blockchain facilitates P2P energy trading platforms, allowing producers and consumers to directly trade energy without intermediaries. This empowers small-scale renewable energy generators, such as rooftop solar panel owners, to sell excess energy to their neighbours, fostering a decentralized energy market. Blockchain's immutable and transparent nature allows for accurate tracking and verification of energy generation, consumption, and transactions. This helps in managing certificates of origin, renewable energy credits, and carbon credits, ensuring compliance with regulations and promoting green energy adoption.

Karandikar et. al. introduces RenewLedger, a platform built on HLF that enables renewable energy transaction and storage management, along with incentivizing direct-to-consumer demand response for peak savings [7].

Moon et. al. suggested a peer-to-peer (P2P) platform that employs HLF, as well as cloud computing for low-capability devices [8]. On the other hand, Kim et al. proposed a P2P platform that utilizes HLF and incorporates an Automated Market Maker (AMM) system [9].

[1] `https://www.hyperledger.org/about`

2.2.2 Education Sector

Blockchain technology has the potential to revolutionize the education sector in various ways. One significant use case is the secure storage and sharing of academic credentials, such as degrees and certificates. This allows for the easy verification of an individual's qualifications by employers and institutions, reducing the risk of fraud. Wai et. al. proposed such a student record storage system based on HLF [10]. This paper also addresses the issue of transaction access time and indexing capability in the HLF blockchain. The proposed storage structure in the paper enhances transaction access time by selecting an appropriate block size based on the configuring parameter and transaction arrival rate, reducing transaction storage time latency. To improve the indexing capability, the paper recommends replicating the metadata of the SR block on an off-chain database, which improves overall performance. By enhancing access to metadata, this technique helps to overcome the limitation of indexing capability.

2.2.3 Healthcare

Blockchain technology can be utilized in the healthcare sector in various ways, including medical records management for secure storage, access, and sharing of electronic health records while maintaining data integrity and patient privacy. It can also be used in supply chain management to track and authenticate drugs, medical devices, and equipment, preventing counterfeit products and ensuring transparency. In clinical trials and research, blockchain can improve data integrity, security, and collaboration. The application of HLF in the healthcare sector can be broadly categorized into two categories - Health data and Supply chain management.

In [11, 12, 13] the researchers used HLF to build a privacy-friendly patient data management system. Wadud et. al. proposed Remote Patient Monitoring for which to secure the data sharing between patients, healthcare providers, and medical devices, the authors used HLF [14].

Amin et. al. proposed a HLF-based Biomedical Supply Chain Management System [15]. The Biomedical Engineering Supply Chain (BESC) plays a crucial role in providing medical equipment, such as Covid-19 testing kits, Personal Protection Equipment (PPE), and medications, to the healthcare industry. It is imperative that these biomedical products can be tracked and their data securely maintained. Failure to do so could lead to the possibility of tampering with critical information, ultimately jeopardizing the safety of patients and the public. Le et. al. introduced a comparable supply chain system, called BloodChain, which is based on HLF technology, for managing

the blood donation network [16]. Tang et. al. also proposed a similar supply chain management system for Epidemic Material [17]. Apart from these, Li et. al. proposed a medicine intellectual property protection scheme based on HLF [18].

2.2.4 Diverse Industries

Diverse sectors are utilizing blockchain technology, specifically through the HLF framework, to enhance trust, transparency, and privacy. These sectors mainly comprise BFSI, E-Voting etc.

Banking, Financial Services and Insurance (BFSI)

In the BFSI sector, Ariffin et. al. created a Trade Finance Application that utilizes software connectors to explore the fundamental building blocks of software interaction for HLF. The application serves as a medium to investigate these components and their implications for the development of HLF-based solutions in the BFSI domain [19]. The study seeks to achieve a thorough comprehension of software connectors for HLF applications. However, evaluating non-functional system properties such as scalability and performance requires additional investigation. The authors note that one possible drawback of HLF is its dependence on Orderer Nodes for block creation, which eliminates decentralized consensus. Lee et. al. developed a Time Bank System that automates manual recording using smart contracts [20]. Aleksieva et. al. implemented smart contracts for insurance services and successfully tested them to create and query records [21]. The proposed system aims to provide faster and more transparent claim processing, but the authors think that the adoption of new technology and the complexity of configuring the blockchain system are potential challenges that need to be addressed.

E-Voting

Marańda et. al. developed a Decentralized Electronic Voting system that achieved auditability, transparency, and openness [22]. The study successfully implemented a complete application and demonstrated feasibility, resulting in tamper-proof voting records. The proposed system aims to increase the trust of voters. The study conducted by Vidwans et al. assesses the effect of an increasing number of transactions on blockchain performance. The authors place particular emphasis on evaluating the management of private keys and HLF's performance. [23]. According to their findings, the use of Hybrid Encryption in a chaincode led to an increase in CPU usage.

Miscellaneous

In addition to its use in well-known sectors such as BFSI, and E-Voting, HLF is also employed in several other domains to enhance privacy, transparency, and decentralization. For instance, it is utilized in supply chain management for tea powder [24], ride-sharing applications [25], KYC processes [26], authentication of Mobile IoT devices [27], personal data vaults [28], and to improve telecom industry processes [29], as well as in enterprise information systems [30].

Chapter 3

Overview of Hyperledger Fabric

Hyperledger Fabric is a blockchain framework that provides businesses with a scalable and modular platform for developing blockchain applications. It is an open-source project created by the Linux Foundation's Hyperledger project and designed for various industries, including healthcare, finance, and supply chain management. The modular architecture of Hyperledger Fabric enables developers to customize the platform as per their application's specific needs. It supports several consensus algorithms, including Raft and Practical Byzantine Fault Tolerance (PBFT), that enable multiple parties to agree on the ledger's state. Smart contracts are supported by Hyperledger Fabric which automates business processes and uses Chaincode, a programming model that supports several programming languages, such as Java, JavaScript and Go. The privacy and confidentiality of enterprise applications are essential, and Hyperledger Fabric ensures this by using a permissioned network model that allows only authorized parties to access the ledger data. Hyperledger Fabric also supports private data collection that enables sensitive data to be shared only with authorized parties.

3.1 Hyperledger Fabric Architecture

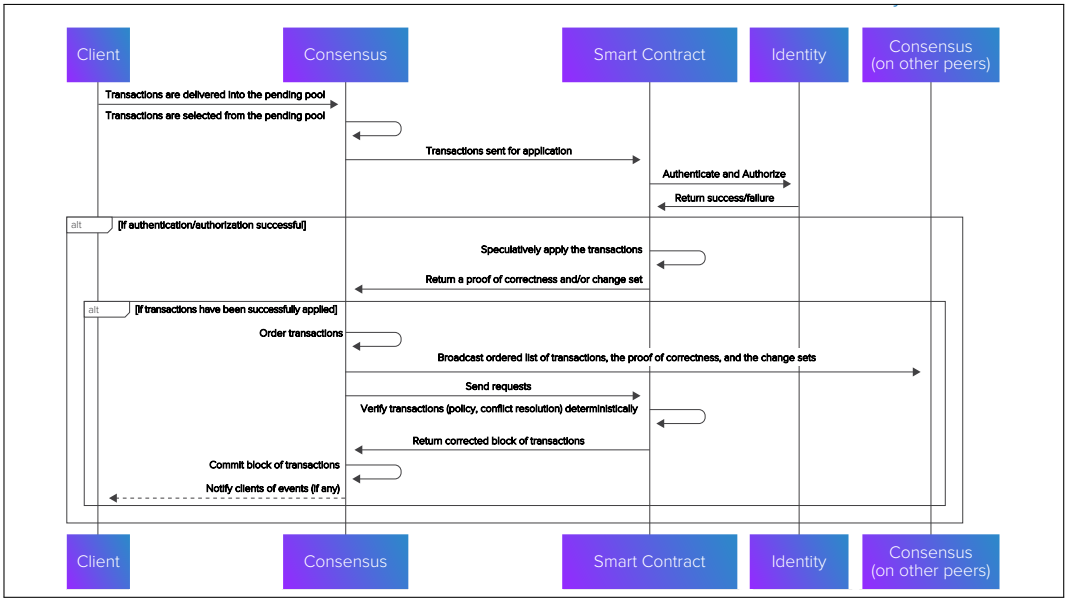


Figure 3.1: Hyperledger Smart Contracts and other layers

Hyperledger projects adhere to design principles that prioritize modularity, interoperability, security, and ease of use. These principles [31] are reflected in the architecture of Hyperledger projects, which are comprised of various distinct components. Hyperledger Fabric comprises three critical components, including the Consensus Layer, which maintains a consistent view of the ledger among network nodes and validates transactions as per the system rules. The Smart Contract Layer is another vital component that automates business processes and executes business logic using smart contracts. The Communication Layer is responsible for securely transporting peer-to-peer messages between nodes, enabling secure propagation of transactions across the network, and secure communication among nodes. Figure 3.1 [31] illustrates how Smart Contracts integrate with and impact other architectural layers within the framework. The Data Store Abstraction and Crypto Abstraction provide flexibility in the use of different types of databases and cryptographic algorithms, respectively, without requiring changes to other parts of the system. Identity Services provide authentication and authorization, ensuring that only authorized users and systems can access the blockchain, while Policy Services manage various policies specified in the system. APIs enable external systems to interface with blockchains in a

standardized manner, and Interoperation promotes interoperability between different blockchain networks.

3.2 Components of Hyperledger Fabric

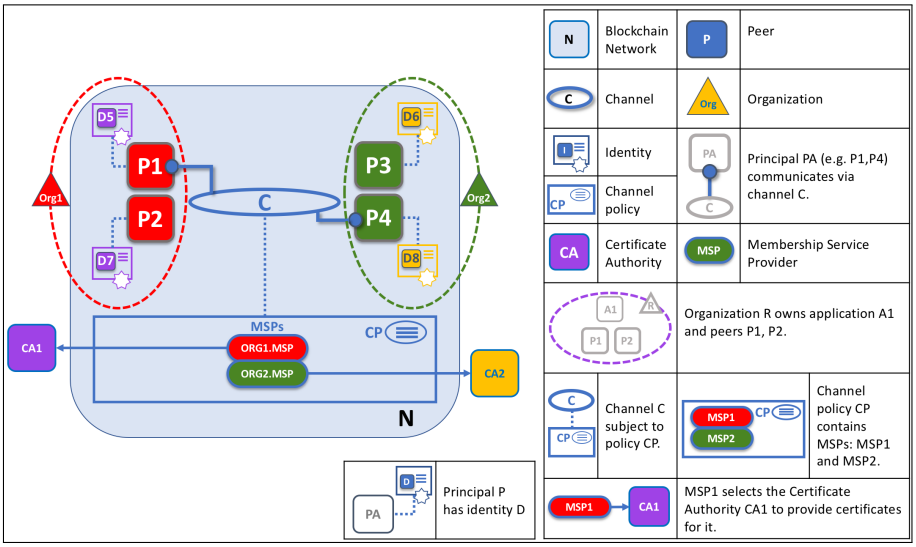


Figure 3.2: Generic Hyperledger Fabric network

One of the main components[1] of the HLF network is a **Channel**. It creates a sort of network subnet that isolates the state and smart contracts. The same data and smart contracts are accessible to all peers participating in a channel. It cannot be accessed from outside the channel. **Peers** host copies of ledgers and smart contracts, making them fundamental nodes in the network. A peer can host several ledgers and smart contracts. The peers can be classified into four categories. **Endorsing peers** receive transactions from clients, validate them, and then endorse them. **Committing peers** receive endorsed transactions from endorsing peers, execute the transactions, and then write the results to the ledger., **Validating peers** participate in the consensus process to ensure that the ledger is valid and consistent and **Anchor peers** act as a reference point for other peers in the network and help them discover and connect to other peers. An HLF component called **Membership Service Provider (MSP)** provides an abstraction of membership processes. An MSP specifically abstracts away the cryptographic protocols and methods involved in certificate issuance, certificate validation, and user

[1]https://developer.ibm.com/articles/blockchain-basics-hyperledger-fabric/

authentication. **Certificate Authorities (CA)** generate public and private key-pair to prove the identity of different users in the network. As a private key cannot be shared, MSP comes in there to prove the identity. The blockchain maintainer setup the blockchain network and provides others with the necessary permissions to interact with the network. In Fabric, the smart contracts are called **Chaincode**. It is a piece of code that is deployed on a blockchain network and executes business logic. They are used to define the rules and logic for executing transactions on the network. A **Ledger** in HLF consists of **World State** and **Blockchain**. The World State of a blockchain network is stored in a state database, also known as a world state database or key-value store. The state database contains the current state of all assets on the network. There are two types of state databases in HLF. LevelDB is the default state database for HLF and is a fast and lightweight database that stores data in key-value pairs. CouchDB is an alternative state database that is more feature-rich than LevelDB and supports more advanced querying and indexing capabilities. In our setup, we have used CouchDB for its advanced nature. The generic architecture of the Hyperledger Fabric network is depicted in Figure 3.2 [32].

Chapter 4

Proposed Model

In our proposed solution, all recruitment firms act as an "organization" within the HLF network. These organizations run multiple peer nodes. These peer nodes can be run by different departments of the organization such as Management, Hiring and the Finance department. The Recruiter can create a transaction proposal and their proposals will need to be endorsed by their superiors or finance team and management of the company. After endorsement is done the transaction is committed to the ledger of the company. In Figure 4.1, Company A and Company B have their private channels, in which they have their internal client applications. A company as per its policy can configure the channel and endorsers. Each company creates a separate channel using anchor peer with the portal admin's peer in the HLF network. The data of job seekers are treated as an asset and stored in the ledger. L3 holds the data about job seekers and using Smart Contract S1, Company A and using S2, Company B intercommunicates with ledgers of different channels. The Orderer node group the transactions into blocks and order them in sequence to commit. A company can have its own orderer or rely on the portal ones. The smart contracts, enforce the rules and logic of the system. Our

Table 4.1: Smart Contract methods

HR Contract	**Admin Contract**	**Job seeker Contract**
createJobposting() hiredUpdate() queryMyJobPostings() readCandidates() deleteJobposting()	deleteJobseeker() deleteJobposting() queryAllJobposting() queryAllJobseeker()	createJobseeker() updateJobseeker() readJobseeker() deleteJobseeker() updateJobseekerPassword() getJobseekerPassword() applyForJob()

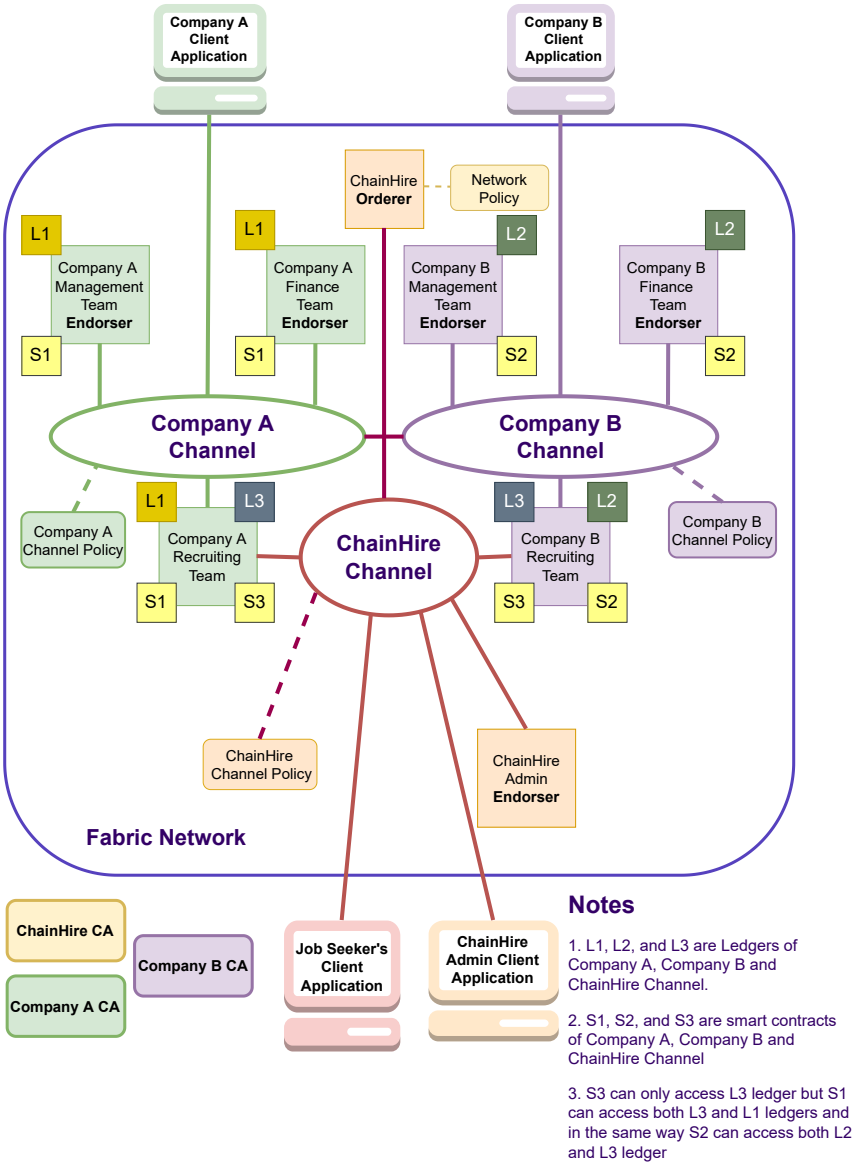


Figure 4.1: Architecture of Proposed Solution

Table 4.2: Use case of different actors

Admin
• **View** and **Moderate** job postings. • **View** and **Moderate** job seekers. (only basic details visible)
Recruiter
• **Create** and **Delete** job postings. • Provides **Updates** about the **Hired** candidates. • **View** job seeker details (only if he/she applied for a job posted by the Recruiter)
Job seeker
• **Create**, **View**, **Update** & **Delete** his profile. • **View** & **Apply** for the Job postings.

solution includes three types of smart contracts: one for administrators, one for recruiters, and one for job seekers. These contracts contain methods that allow different users to perform specific actions on the network. Tables 4.1 and 4.2 provide details on the methods and use cases of the different actors in the system.

The smart contract logic ensures that even the portal admin can access the basic details of the candidate and not full details, on the other hand, it also ensures that a recruiter can only able to see his profile if the job seeker makes any application for a job posted by that recruiter otherwise not. Thus the privacy of the job seeker is maintained. Once a candidate is selected, the recruiter update that in the system. After the update, a unique anonymous identifier of the hired candidate is then updated to that job posting which is also visible to other job seekers. Thus even without disclosing the hired candidate's details, we are able to become transparent with the other candidates.

Algorithm 1 represents the `createJobseeker()` smart contract method. This algorithm is used to add the information of a job seeker to the ledger when they register on the portal and provide their details. Algorithm 2 represents the `createJobposting()` method that is invoked by the Recruiter to add a new job advertisement in the ledger given its details. Algorithm 3 represents the `applyForJob()` method. This algorithm can be used by job seekers to submit an application for a job opportunity. Algorithm 4 allows a recruiter to access the information of a job seeker who has applied for a job role that the recruiter has posted. Before the job seeker's details are shared, the algorithm verifies that the job seeker has actually applied for the role in question. This algorithm represents the `readCandidates()`. Algorithm 5 allows a recruiter to mark a candidate as hired for a specific job role by updating their information in the ledger. It represents the `hiredUpdate()`

method. This algorithm serves to ensure the integrity and transparency of the hiring process by updating the information in the ledger once a candidate has been hired. The identifier of the hired candidate is also updated in the ledger, allowing other job seekers to see that. This feature prevents any tampering or changes to the hiring decision and promotes transparency within the system.

Algorithm 1 Job seeker registration

Input: $jobseekerId$, $firstName$, $lastName$, $password$, age, $phoneNumber$, $address$

Ensure: $password \neq empty$ and $jobseekerId$ not exists

1: Create record r of type JobseekerRecord
2: Set $r.jobseekerId = jobseekerId$
3: Set $r.firstName = firstName$
4: Set $r.lastName = lastName$
5: Set $r.password = password$
6: Set $r.age = age$
7: Set $r.phoneNumber = phoneNumber$
8: Set $r.address = address$
9: Submit r to add to the Blockchain.

Algorithm 2 Creating a new job posting

Input: $jobpostingId$, $HRId$, $profilename$, $qualification$, $agelimit$, $salary$, $location$, $yearofexperience$, $skills$, $companyname$

Ensure: $jobpostingId$ not exits

1: Create record r of type JobPostingRecord
2: Set $r.jobpostingId$=$jobpostingId$
3: Set $r.HRId$=$HRId$
4: Set $r.profilename = profilename$
5: Set $r.qualification = qualification$
6: Set $r.agelimit = agelimit$
7: Set $r.salary = salary$
8: Set $r.location = location$
9: Set $r.yearofexperience = yearofexperience$
10: Set $r.skills = skills$
11: Set $r.companyname = companyname$
12: Submit r to add to the Blockchain.

Algorithm 3 Applying for a job

Input: $jobpostingId$, $jobseekerId$

/* r_{jp} is a object of type JobPostingRecord and r_{js} is a object of type JobseekerRecord */

$r_{\text{jp}} = readJobposting(jobpostingId)$
$r_{\text{js}} = readJobseeker(jobseekerId)$
if not r_{js} already applied for the job **then**
    $r_{\text{js}}.appliedTo.push(jobpostingId)$
    $r_{\text{jp}}.appliedCandidates.push(jobseekerId)$
    Submit r_{js} and r_{jp} to update the blockchain ledger.
end if

Algorithm 4 A Recruiter reviewing the details of a job seeker

Input: $jobseekerId$, $jobpostingId$, $HRId$

/* r_{jp} is a object of type JobpostingRecord and r_{js} is a object of type JobseekerRecord */

$r_{\text{jp}} = readJobposting(jobpostingId)$
$r_{\text{js}} = readJobseeker(jobseekerId)$
/* If the Jobseeker applied for the job posted by HR, then only that HR can see the Jobseeker's details */
if $r_{\text{js}}.appliedTo.includes(jobpostingId)$ and $r_{\text{jp}}.HRId = HRId$ **then**
    Create record r of type JobseekerRecord
    Set $r.jobseekerId = r_{\text{js}}.jobseekerId$
    Set $r.firstName = r_{\text{js}}.firstName$
    Set $r.lastName = r_{\text{js}}.lastName$
    Set $r.age = r_{\text{js}}.age$
    Set $r.phoneNumber = r_{\text{js}}.phoneNumber$
    Set $r.address = r_{\text{js}}.address$
    Set $r.qualification = r_{\text{js}}.qualification$
    Set $r.achievements = r_{\text{js}}.achievements$
    Set $r.experience = r_{\text{js}}.experience$
    Set $r.skills = r_{\text{js}}.skills$
    Set $r.certificates = r_{\text{js}}.certificates$
    return r
else
    State HR doesn't have permission to view.
end if

Algorithm 5 A Recruiter marking a candidate as hired in the system

Input: $jobseekerId$, $jobpostingId$

1: $r_{\text{jp}} = readJobposting(jobpostingId)$
2: $r_{\text{js}} = readJobseeker(jobseekerId)$
/*The HR can update someone as hired only if he applied for that posting.*/
3: **if** $r_{\text{js}}.appliedTo.includes(jobpostingId)$ **then**
4: $r_{\text{jp}}.hired = r_{\text{js}}.jobseekerId$
5: Submit r_{jp} to update the blockchain ledger.
6: **else**
7: State Jobseeker not applied in this posting.
8: **end if**

4.1 Project Requirements

4.1.1 Hardware

The system requires the following hardware:

- RAM: at least 8GB (each Node)
- Hard Disk: at least 20 GB (each Node)
- Processor: at least AMD Ryzen 3 or Intel i5 7th generation or above for the local development or Intel Xeon Family processor with 3.3 GHz clock speed (comes with t2.micro or above of EC2 at AWS).

4.1.2 Software

The system requires the following software:

- Blockchain: Hyperledger Fabric 2.2
- Operating System: any Debian-based Linux Distro 22.04 or above
- Web Server: NodeJS with ExpressJS and EJS
- Technologies: Javascript, HTML, CSS, Git, Bash, Docker

4.2 Project Modules

4.2.1 Registration

Job seekers can register with their contact and experience details, as well as their profile information.

4.2.2 Search

Job seekers can search for job opportunities based on their interests and apply for positions that they are interested in.

4.2.3 Job Post

Recruiters can post job openings for their organization, including information about the profile name, salary, working location, required educational qualification, and required experience.

4.2.4 Manage Account

Job seekers have the option to delete or update their accounts at any time. The admin is responsible for managing the details of both Job seekers and Job postings.

4.3 System Design

The use case for different actors of our system is depicted in the Use case diagram in Figure 4.2 and the following subsection provides its Text description.

4.3.1 Use Case Diagram

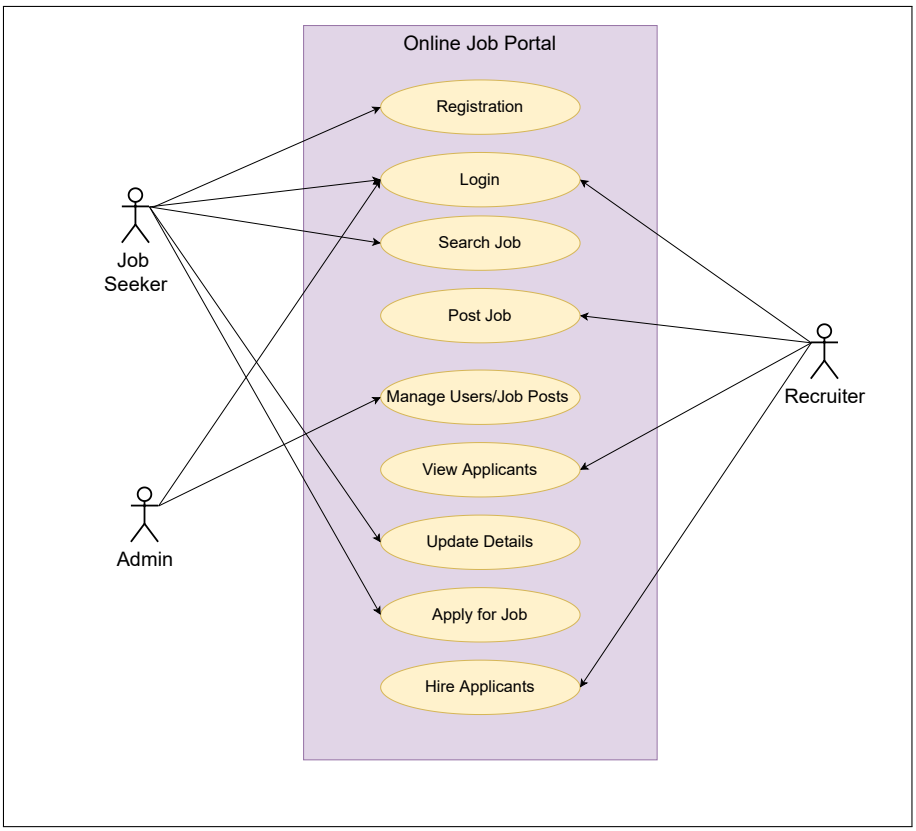


Figure 4.2: Use case diagram of the system

Text Description

U1: Registration: Using this use case, the Job seeker can register himself by providing the necessary details.

Scenario 1: Mainline sequence

1. Job seeker: Select the register option
2. System: Display prompt to enter a name, phone number, age, OTP etc.
3. Job seeker: Enter the necessary values.
4. System: Display message that the job seeker has successfully been registered.

Scenario 2: At step 4 of mainline sequence

1. System: All the required information is not added.

U2: Login: Using this use case, the Job seeker, Recruiter or Admin can log in to the system by providing the necessary details.

Scenario 1: Mainline sequence

1. User: Select login option
2. System: Displays a prompt to enter Username and Password.
3. User: Enter the necessary values.
4. System: Displays log-in successfully and show dashboard.

Scenario 2: At step 4 of mainline sequence

1. System: Invalid credentials entered.

U3: Search Job: Using this use case, the Job seeker can search for a job.

Scenario 1: Mainline sequence

1. Job seeker: Selects the search job option with keywords.
2. System: Displays the jobs available for him.

Scenario 2: At step 2 of Mainline sequence

1. System: No suitable jobs available.

U4: Post Job: Using this use case, the Recruiter can register a job by providing the necessary details.

Scenario 1: Mainline sequence

1. Recruiter: Select the post job option.
2. System: Display prompt to enter a profile name, salary, location, eligibility etc.
3. Recruiter: Enter the necessary values.
4. System: Display message that the job post has successfully been added.

Scenario 2: At step 4 of mainline sequence

1. System: All the required information is not added.

U5: Manage Users/Job Posts: Using this use case, the Admin can manage users.

Scenario 1: Mainline sequence

1. Admin: Select to manage Job seekers or Job posts.
2. System: Display the either Job seeker or Job posts as per the selection at step 1.
3. Admin: Admin perform his moderation actions.
4. System: Display message that the operation is successful.

U6: View Applicants: Using this use case, the Recruiter can see candidates for a job post.

Scenario 1: Mainline sequence

1. Recruiter: Select the specific job post.
2. System: Displays the list of candidates and their details.

Scenario 2: At step 2 of mainline sequence

1. System: Displays no candidates applied.

U7: Update details: Using this use case, the job seeker can update his/her details

Scenario 1: Mainline sequence

1. Job seeker: Select the update option.
2. System: Displays the update form.
3. Job seeker: Enters all the updated details.
4. System: Displays the update is successfully done.

U8: Apply for Job: Using this use case, the job seeker can apply for a job

Scenario 1: Mainline sequence

1. Job seeker: Select the Search job option.
2. System: Displays the available jobs.
3. Job seeker: Selects a specific job and applies to it.

4. System: Displays the application is received.

U9: Hire Applicants: Using this use case, the recruiter can hire an applicant.

Scenario 1: Mainline sequence

1. Recruiter: Select the job post and check for applied candidates.
2. System: Displays the available candidates.
3. Recruiter: Selects a specific applicant and chooses the hire option.
4. System: Displays the applicant is hired.

4.3.2 Data Flow Diagram

Figure 4.3 illustrates the data flow at a high-level, or Level 0, in our system, and Figure 4.4 provides a more detailed, or Level 1, view of the data flow.

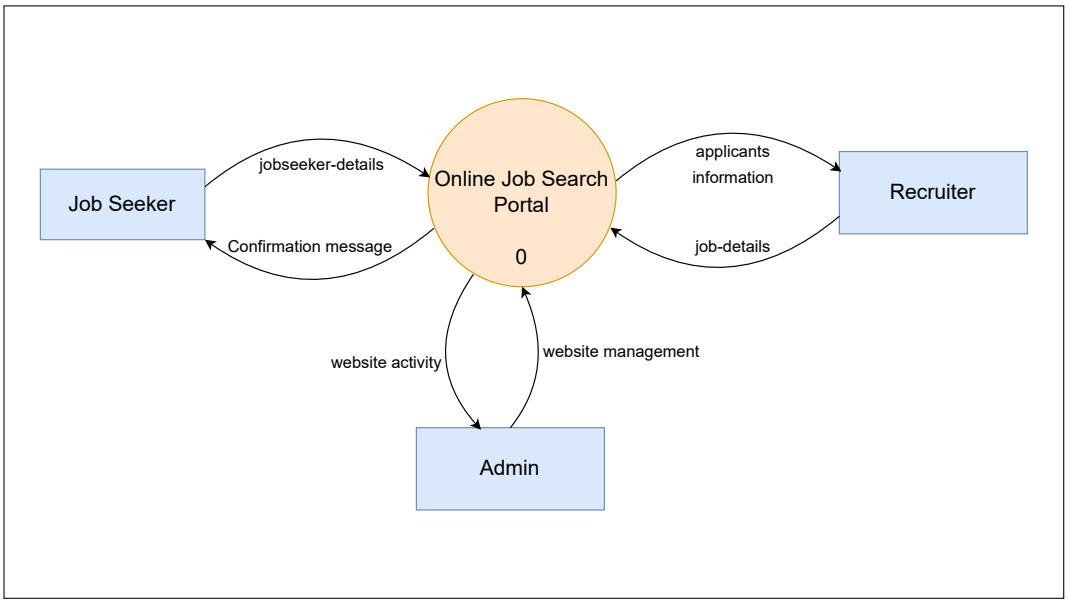


Figure 4.3: Data flow diagram (Level 0)

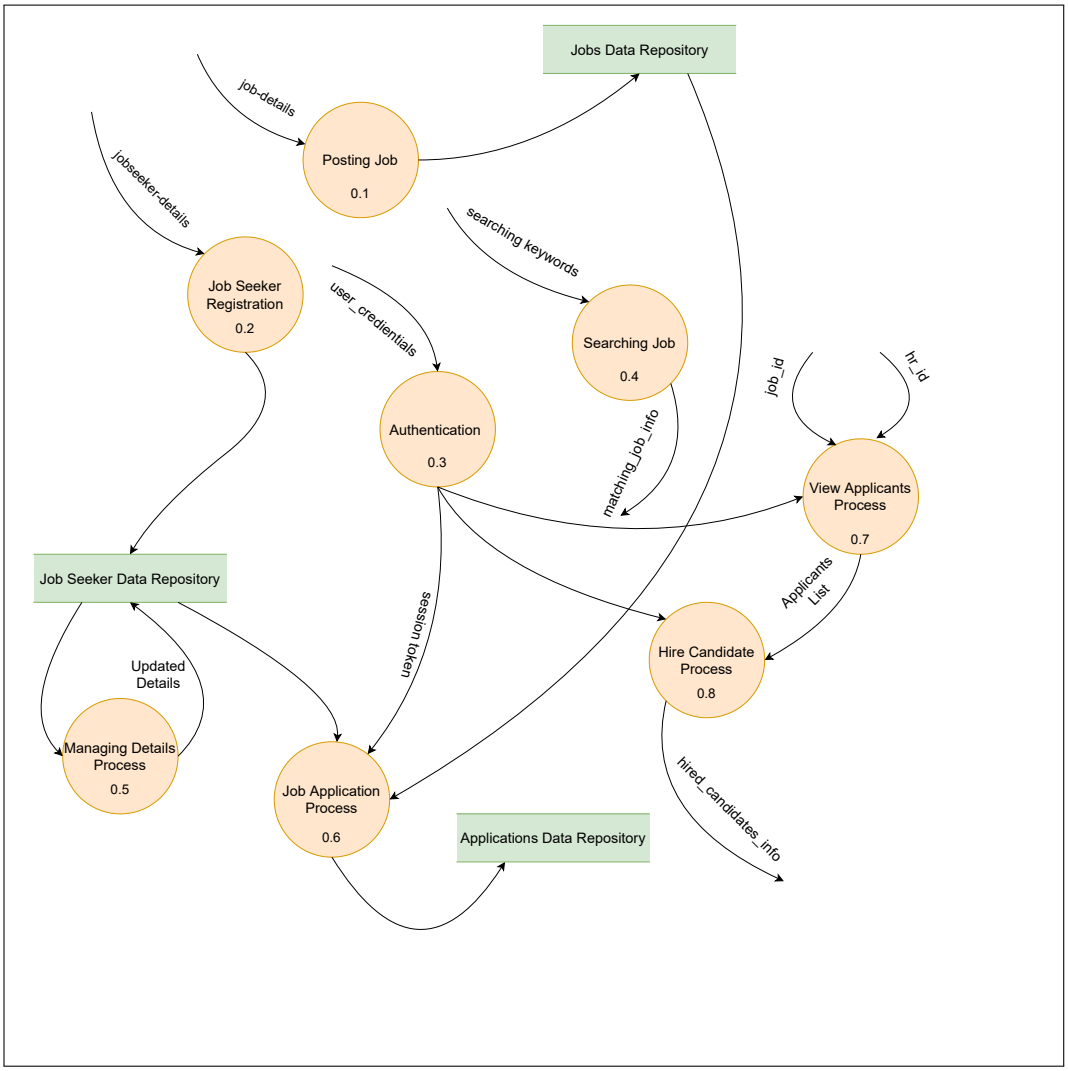


Figure 4.4: Data flow diagram (Level 1)

Data Dictonary

The section describes the data dictionary of Figure 4.3 and Figure 4.4.

job-details = job_id + hr_id + Age limit + Company name + Location + Profile Name + Qualification + Salary + Skills + Year of Experience

jobseeker-details = First Name + Last name + Age + Phone Number + Address + Password + (qualification) + (achievements) + (experience) + (skills) + (certificates)

user_credential = Phone Number + Password

matching_job_info = {job-details}*

hired_candidates_info = {jobseeker-details}^{1}

Chapter 5

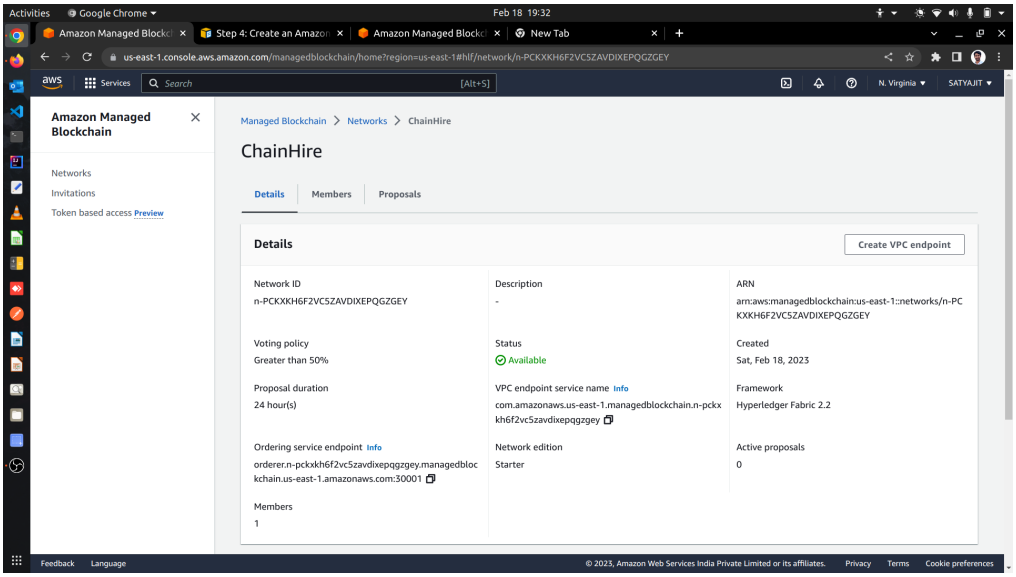
Implementation

To understand the practical feasibility of the system, we have developed a prototype of the job search portal and deployed it using the Amazon Managed Blockchain Service[1].

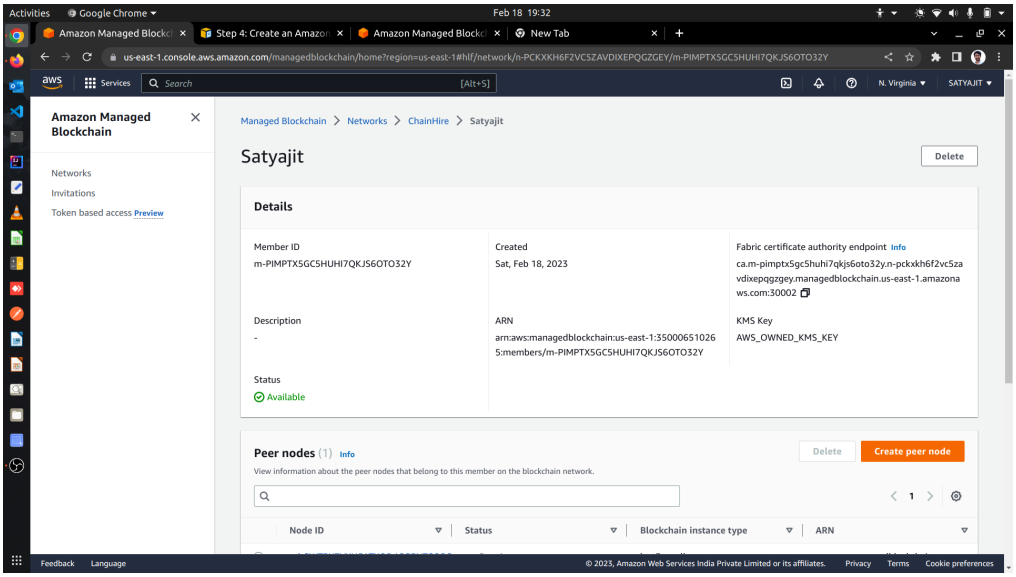
5.1 Amazon Managed Blockchain Service

Deploying a Hyperledger Fabric blockchain network using Amazon Managed Blockchain service involves several steps. Here are the high-level steps involved in the process:

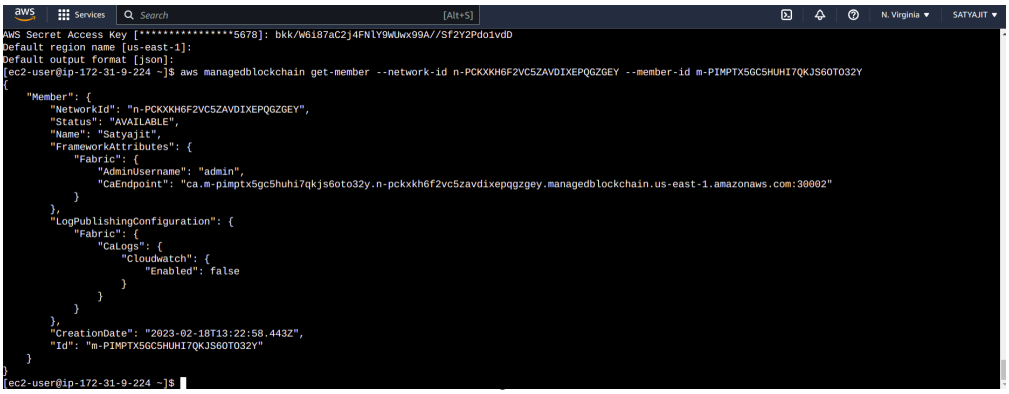
[1]https://aws.amazon.com/managed-blockchain/



(a)



(b)



(c)

Figure 5.1: ChainHire network on Amazon Managed Blockchain Service

- **Create an Amazon Managed Blockchain network:** First, you need to create an Amazon Managed Blockchain network. You can do this using the AWS Management Console or the AWS Command Line Interface (CLI). You will need to specify the network name, the blockchain framework (in this case, Hyperledger Fabric), the membership type, and other network configurations (depicted in Figure 5.1(a)).

- **Configure network settings:** Once the network is created, you can configure network settings such as the number of nodes, node instance types, and availability zones. You can also set up the network's access control policies and integrate it with other AWS services like AWS Key Management Service (KMS).

- **Create a Hyperledger Fabric channel:** After the network is created and configured, you need to create a Hyperledger Fabric channel. A channel is a private communication path between specific network members that allows them to share information and conduct transactions. You can create channels using the Hyperledger Fabric CLI or SDK.

- **Deploy chaincode:** Chaincode is a smart contract that runs on the Hyperledger Fabric network. You can deploy chaincode using the Hyperledger Fabric CLI or SDK. Once deployed, chaincode can be invoked by network members to execute transactions on the network.

- **Add members to the network:** You can add members to the network by inviting them to join the network. Members can be added using the AWS Management Console or the AWS CLI. Each member will need to create a Hyperledger Fabric identity and join the appropriate channel to participate in the network (depicted in Figure 5.1(b)).

- **Test and monitor the network:** Once the network is deployed and members are added, you can test and monitor the network to ensure it is functioning correctly. You can use tools like Hyperledger Explorer to view the blockchain and monitor network activity (depicted in Figure 5.1(c)).

Once the Hyperledger Fabric blockchain network is deployed using Amazon Managed Blockchain service, the next steps involve configuring an Elastic IP address for the EC2 node that hosts the client application and adding an address type record in the DNS panel of Cloudflare to point the domain towards the node (depicted in Figure 5.2).

Figure 5.2: DNS record configuration on Cloudflare

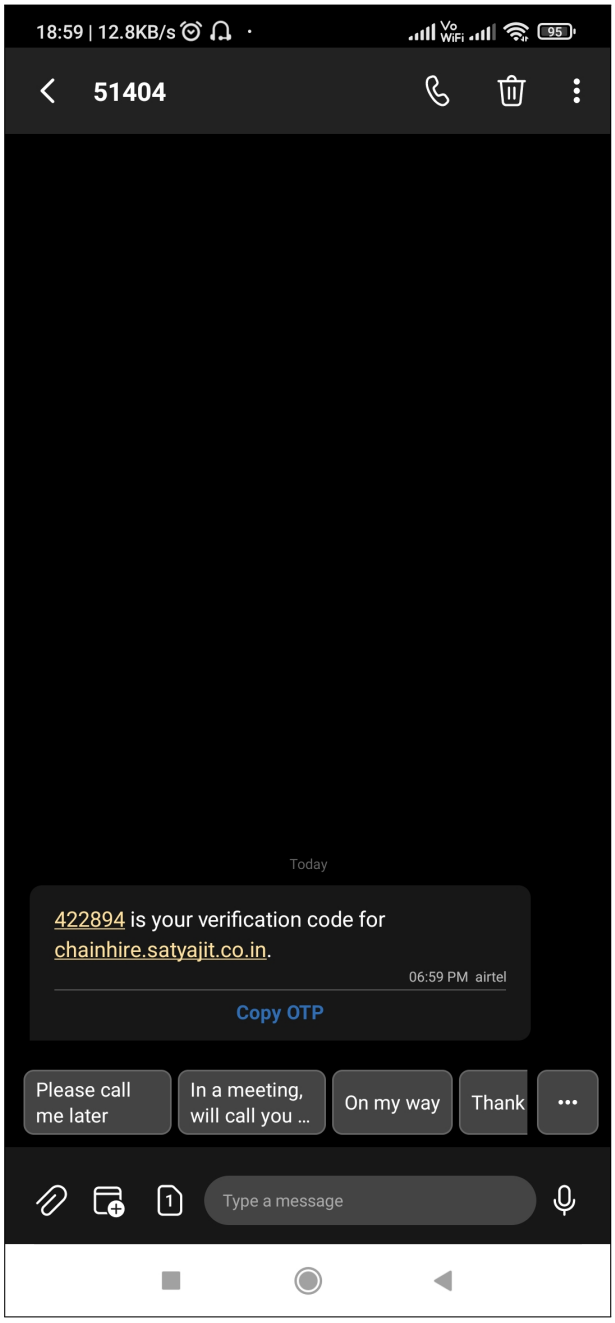


Figure 5.3: OTP verification using Google Firebase

To secure the communication between the client application and the blockchain network, it's important to generate an SSL certificate on the EC2 node for HTTPS. To automate this process, we can use Certbot, a popular tool for obtaining and installing SSL/TLS certificates from Let's Encrypt. By

implementing HTTPS with an SSL certificate, users can securely access the client application and interact with the blockchain network over the internet. To prevent bots from accessing the portal, we integrated Google Firebase to facilitate OTP verification and implement a reCAPTCHA system during job seeker registration (depicted in Figure 5.3).

5.1.1 Cost

The hourly cost for running this production network can be broken down as follows[2]:

- The cost for a single network member using the Standard Edition is $0.55 per hour.
- The cost for each peer node (there are two per member) using the m5.2xlarge instance type is $1.23 per hour.
- The cost for storage used by each peer node (with 500 GiB per node) is $0.139 per hour.
- The cost for data written to the network by each member is $0.01 per hour.

When you add all of these costs together, the total hourly cost for running the production network is $1.93.

5.2 User Interface (UI)

The user interface in Chainhire facilitates the tasks of job seekers, recruiters, and administrators. Figures 4.4 to 4.18 show the sequential user experience in Chainhire, which guides users through the platform and helps them perform their respective tasks.

[2]As per the example rates as of May 2023 given by AWS.

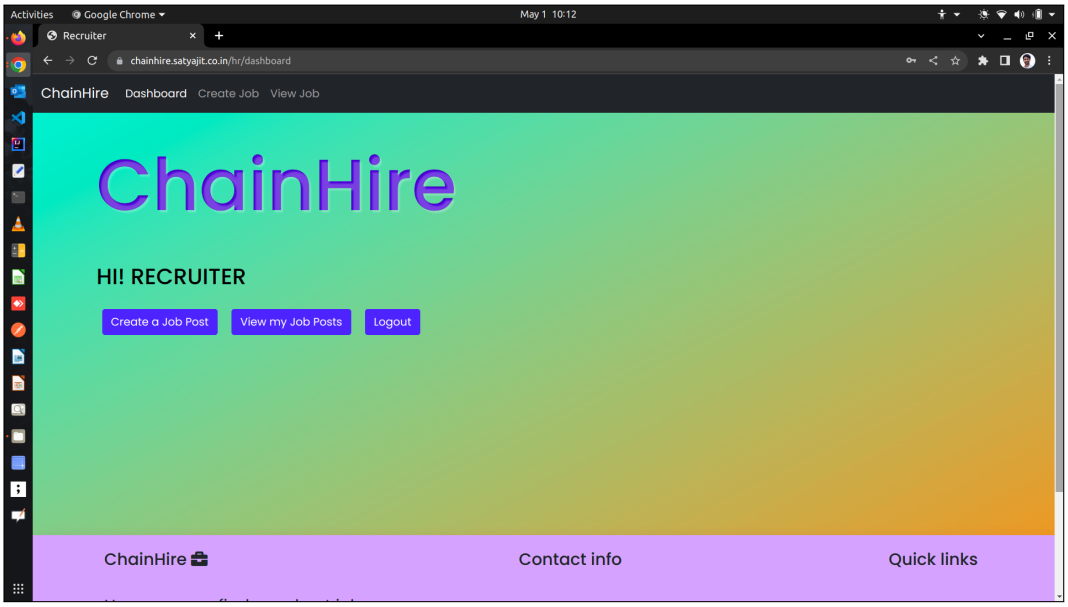


Figure 5.4: Recruiter opens dashboard

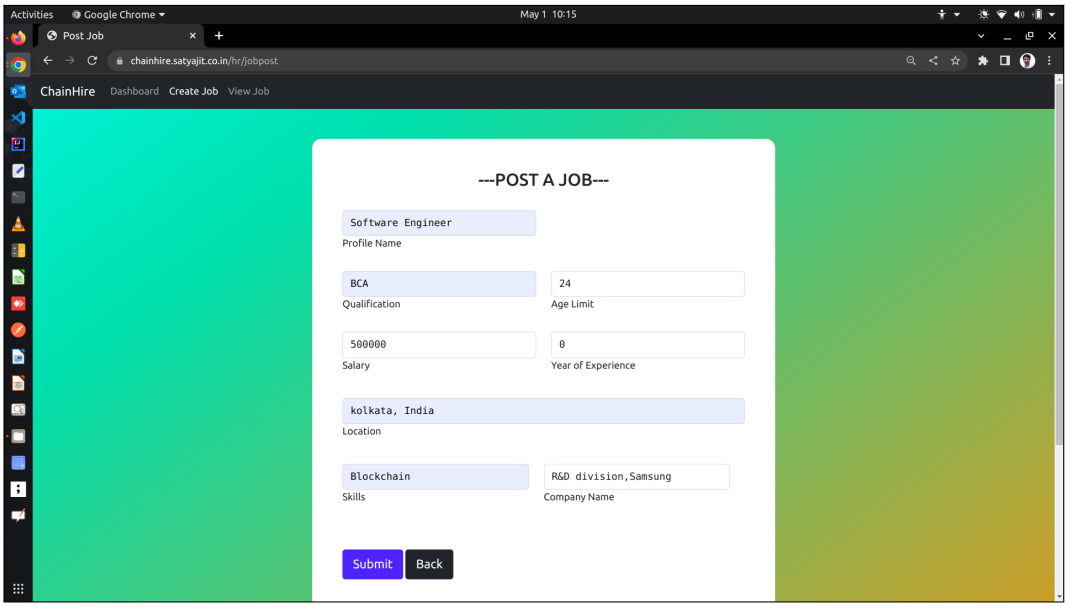


Figure 5.5: Recruiter creates a job advertisement

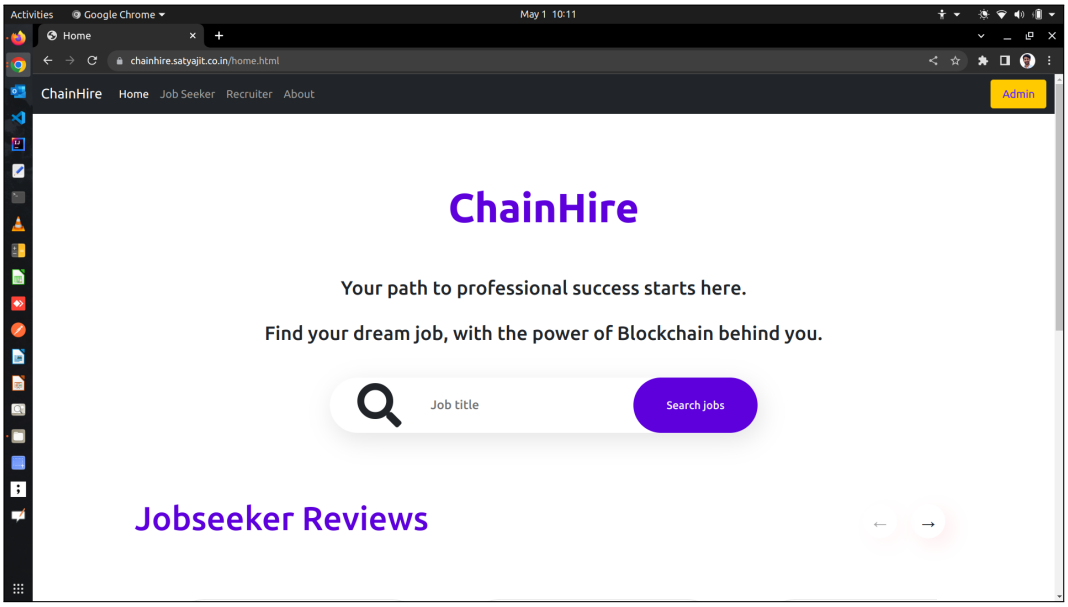


Figure 5.6: Job seeker visits the portal

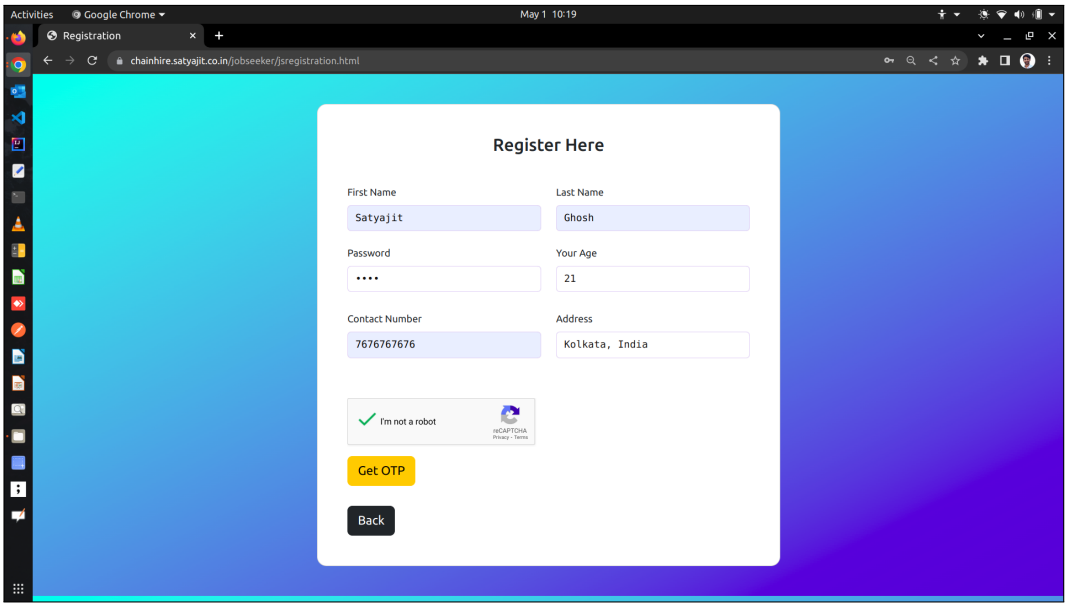


Figure 5.7: Job seeker register himself/herself

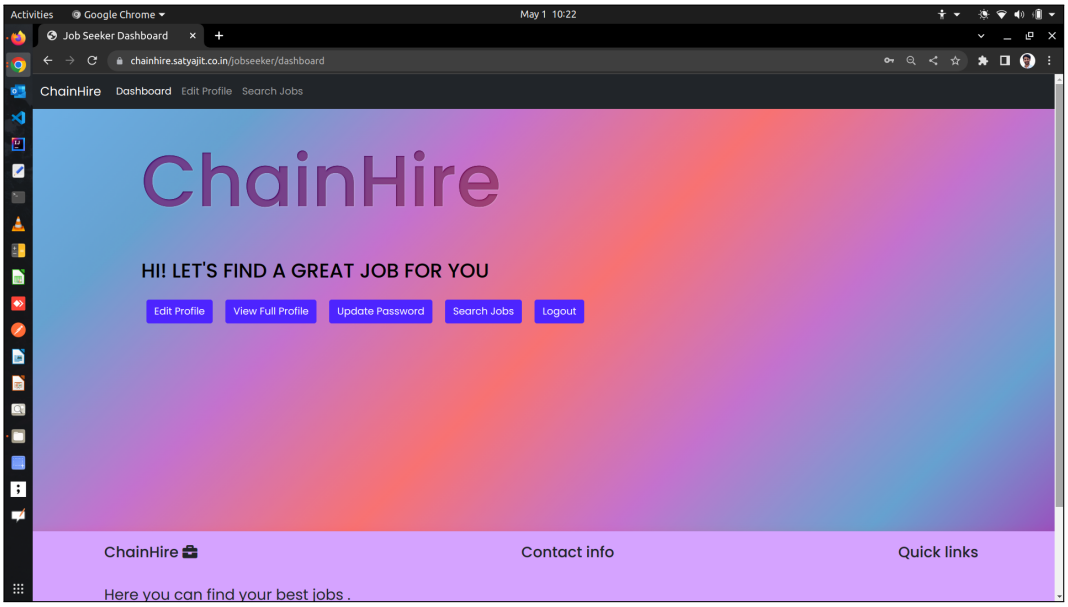


Figure 5.8: Job seeker visits dashboard

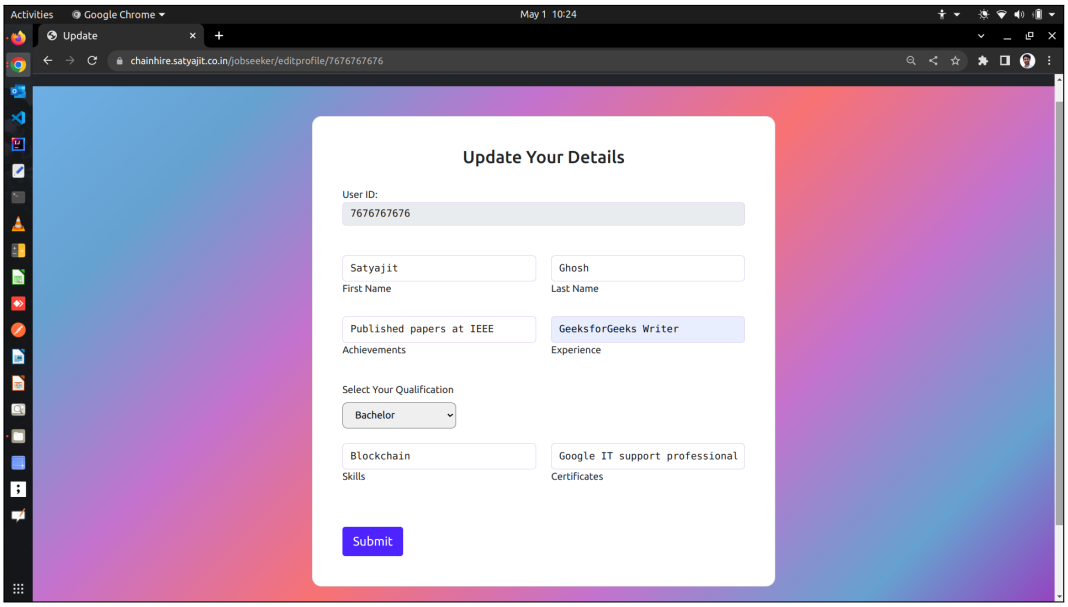


Figure 5.9: Job seeker updates additional details

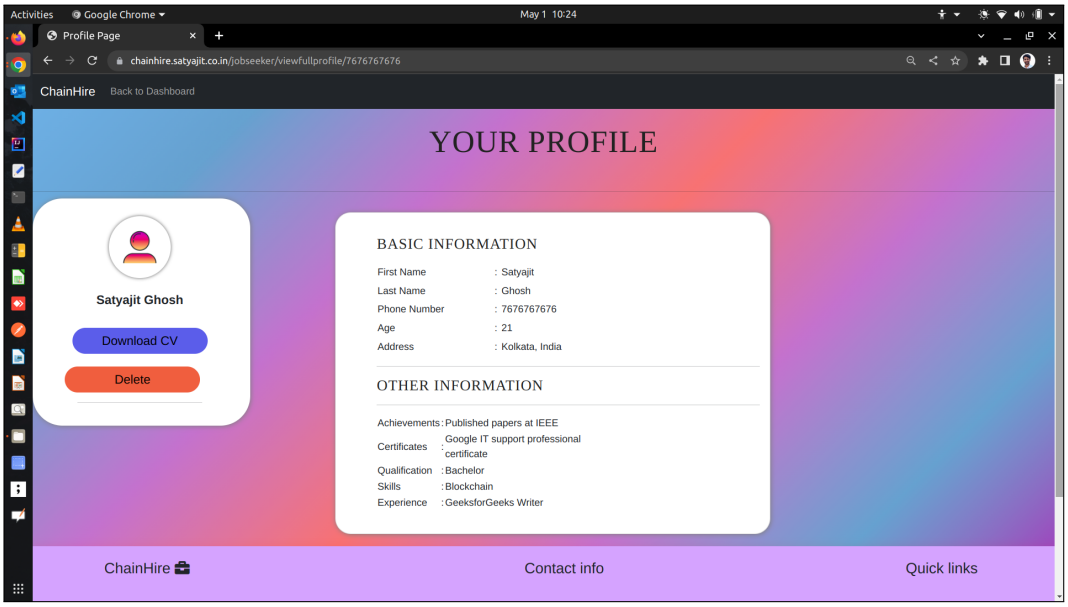


Figure 5.10: Job seeker checks his/her full profile

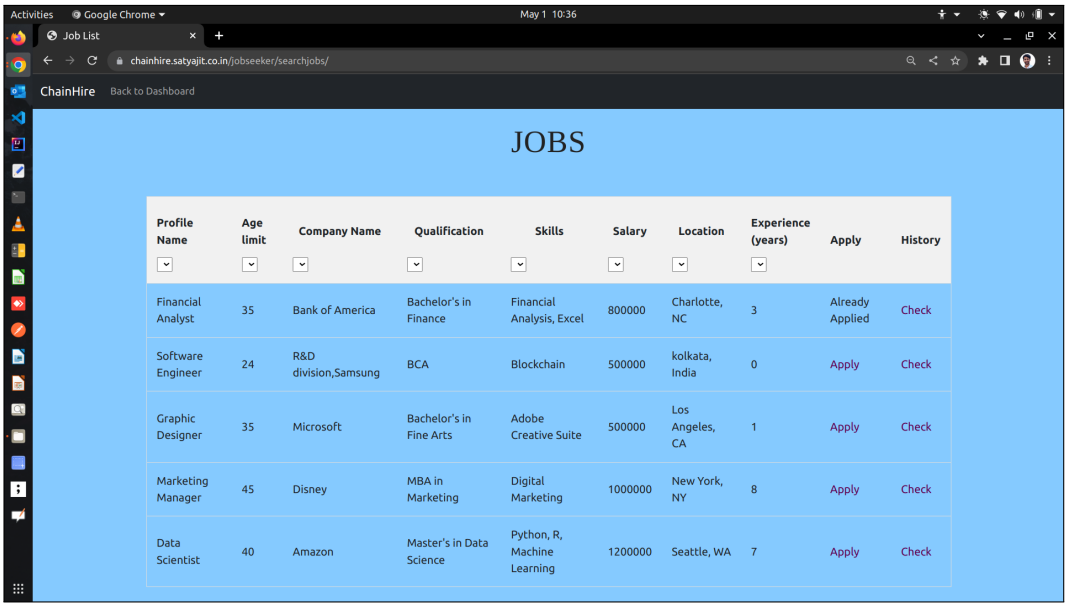


Figure 5.11: Job seeker applies for suitable Job

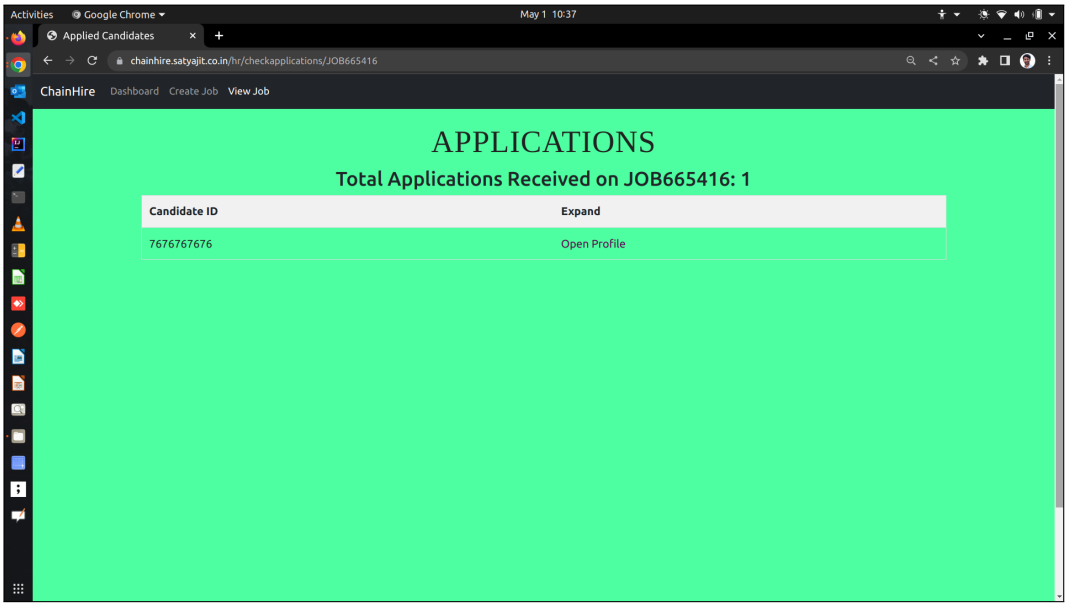


Figure 5.12: Recruiter checks for applied candidates list

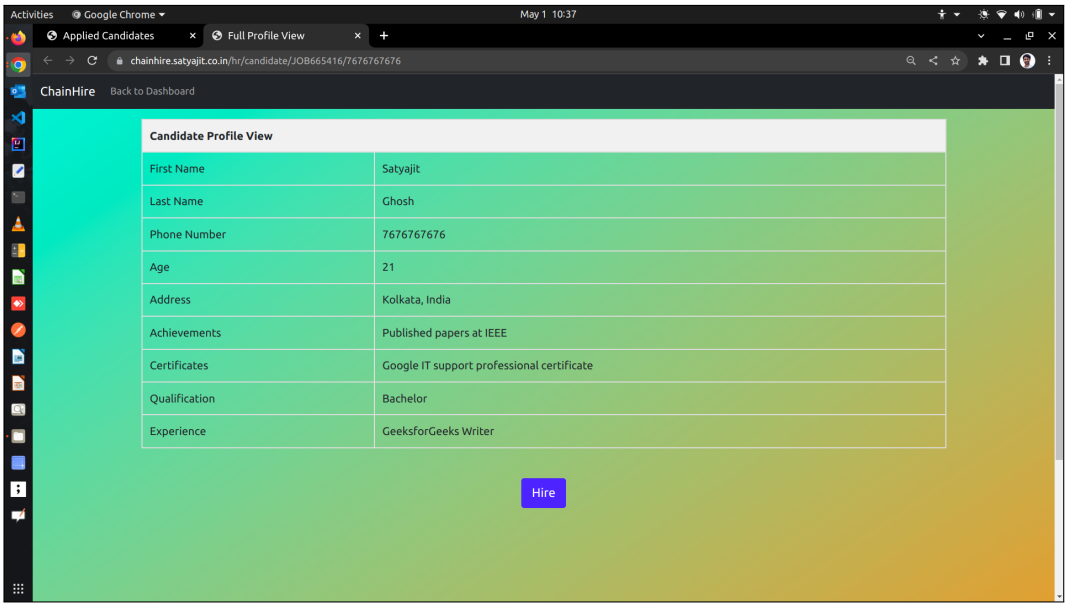


Figure 5.13: Recruiter checks candidate details

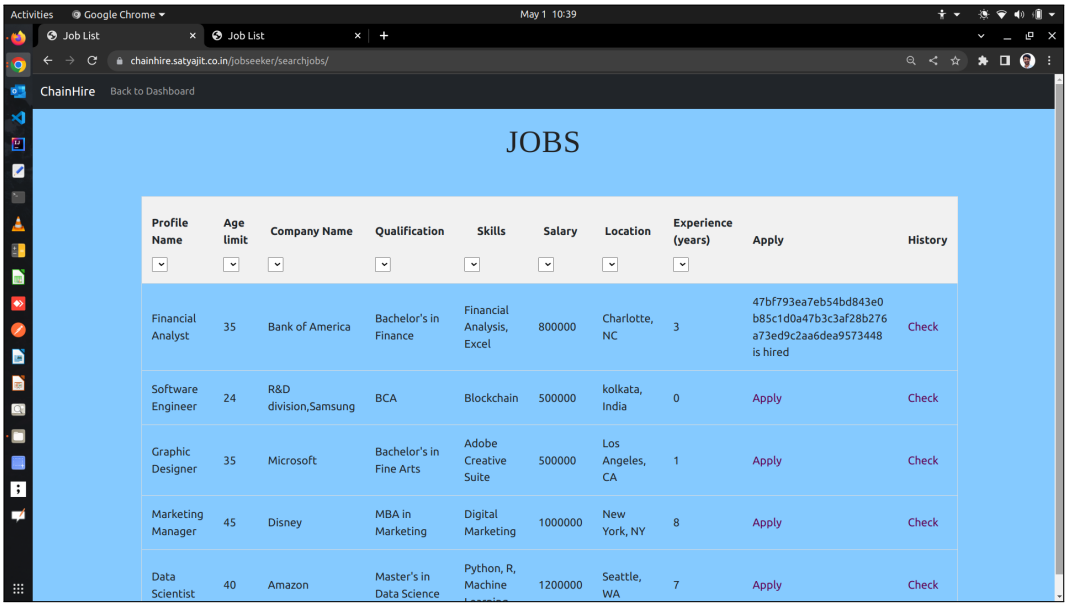


Figure 5.14: Recruiter hires the candidate and that is reflected in the dashboard

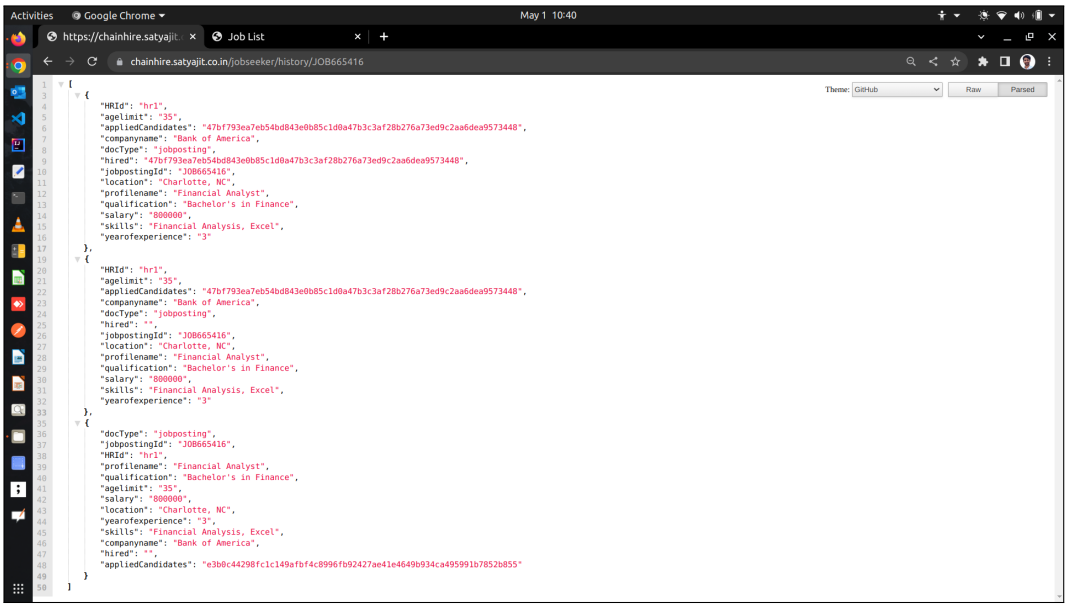


Figure 5.15: Logging all the updates on the job record with the help of blockchain ledger

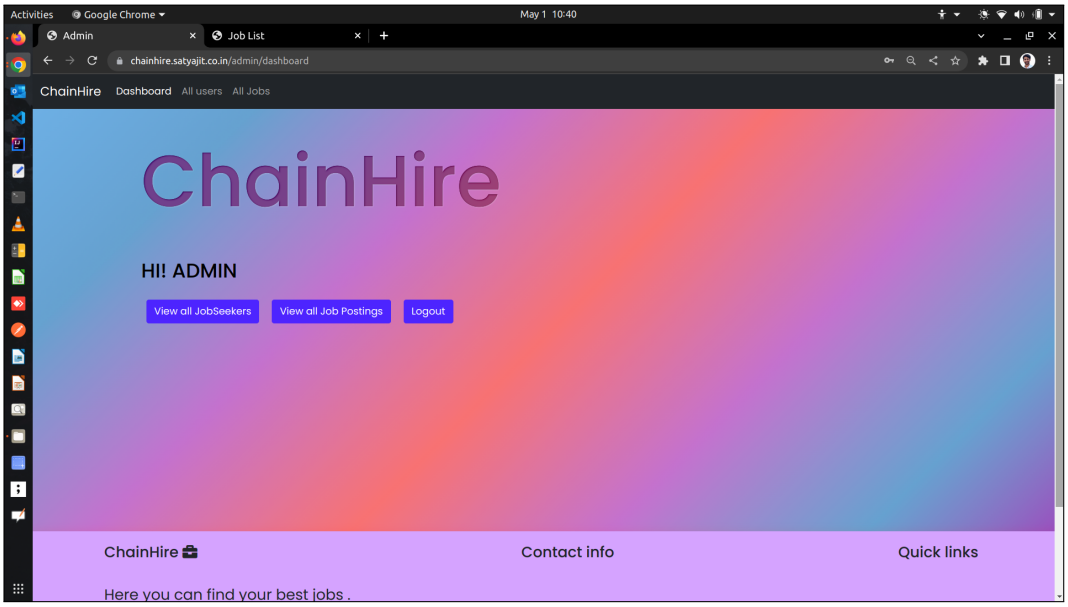


Figure 5.16: Admin enters into dashboard

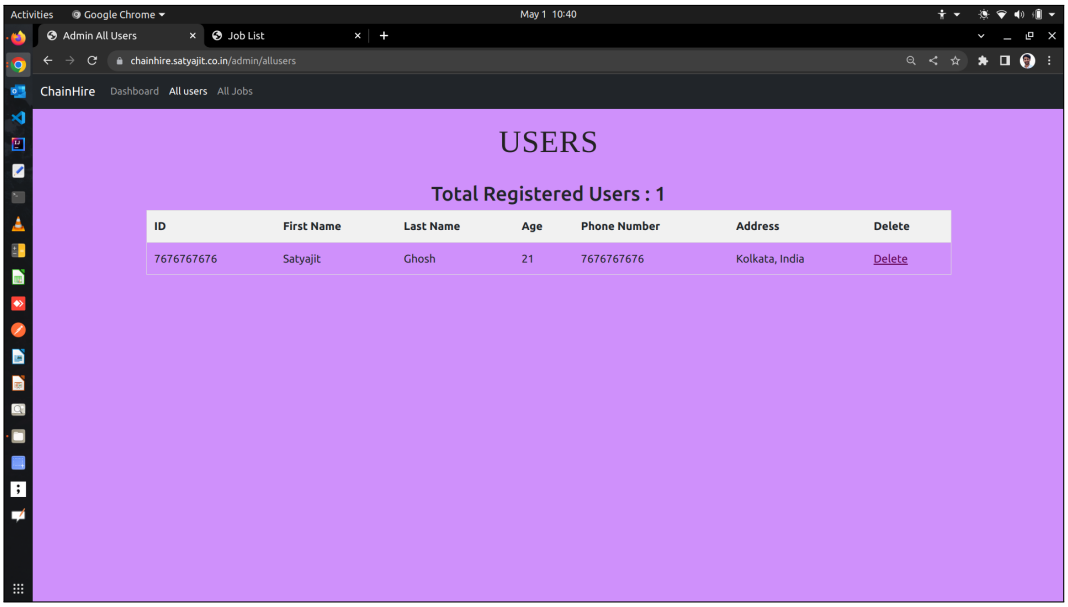


Figure 5.17: Admin moderates the candidates

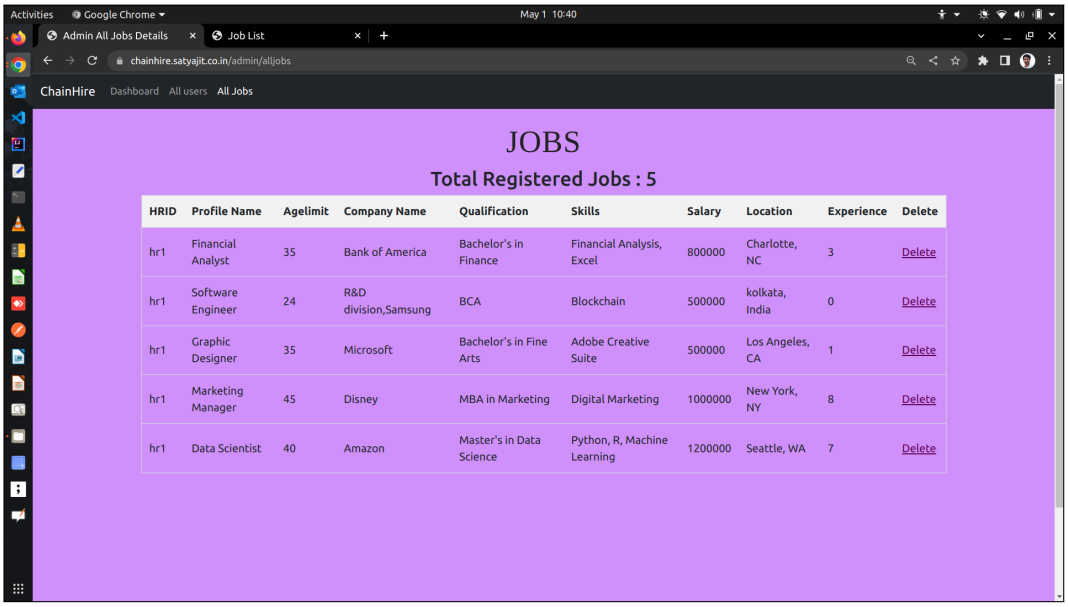


Figure 5.18: Admin moderates the job advertisements

Chapter 6

Evaluation

To evaluate the blockchain performance metrics, we have used Hyperledger Caliper[1]. Caliper is a benchmarking tool for blockchains that enables users to assess the effectiveness of a blockchain implementation using a number of predefined use cases. Calliper provides information about Max latency, Min Latency, Avg Latency, Throughput, Success rate, etc. The term "transaction latency" refers to the amount of time it takes for one user to receive a response after sending a request. The term "transaction throughput" refers to the volume of successfully sent data per second. We have tested the blockchain network on a system with AMD Ryzen 3 5300U processor having a base Frequency of 2.6GHz and Max Frequency of 3.8GHz with 8GB of DDR4-3200 RAM. We have configured HLF with Raft [33] ordering service with two organizations and two peers on each organization. The endorsement can be done by any peer of the organization. We have classified the smart contract methods used for Read, Write and Update operations and taken average values for each type of operation. Caliper is configured to send 1000 transactions on fixed-load configuration. This configuration tries to maximize the throughput by maintaining a defined number of backlog transactions which is also called Transaction Load in Caliper. We have configured the number of backlog transactions to be 5, 10, 20, 40 and 80. As depicted in Fig. 6.1(a), a steady rise in latency is observed in **Read** operation. On the other hand in the case of the **Update** and **Write** operation, a rise in 5 to 10 transaction load was observed initially after that it receives a steady decrease and reached saturation level at around 40 transaction load.

The throughput for all types of operation receives a steep rise for a load of up to 40 and then receives some downward trends which are depicted in Fig. 6.1(b). We can say that around 40 transaction backlogs were been a saturation point for our system and the analysis of the number of allowed backlog transactions can be helpful for deployed system to maximize the

[1] `https://www.hyperledger.org/use/caliper`

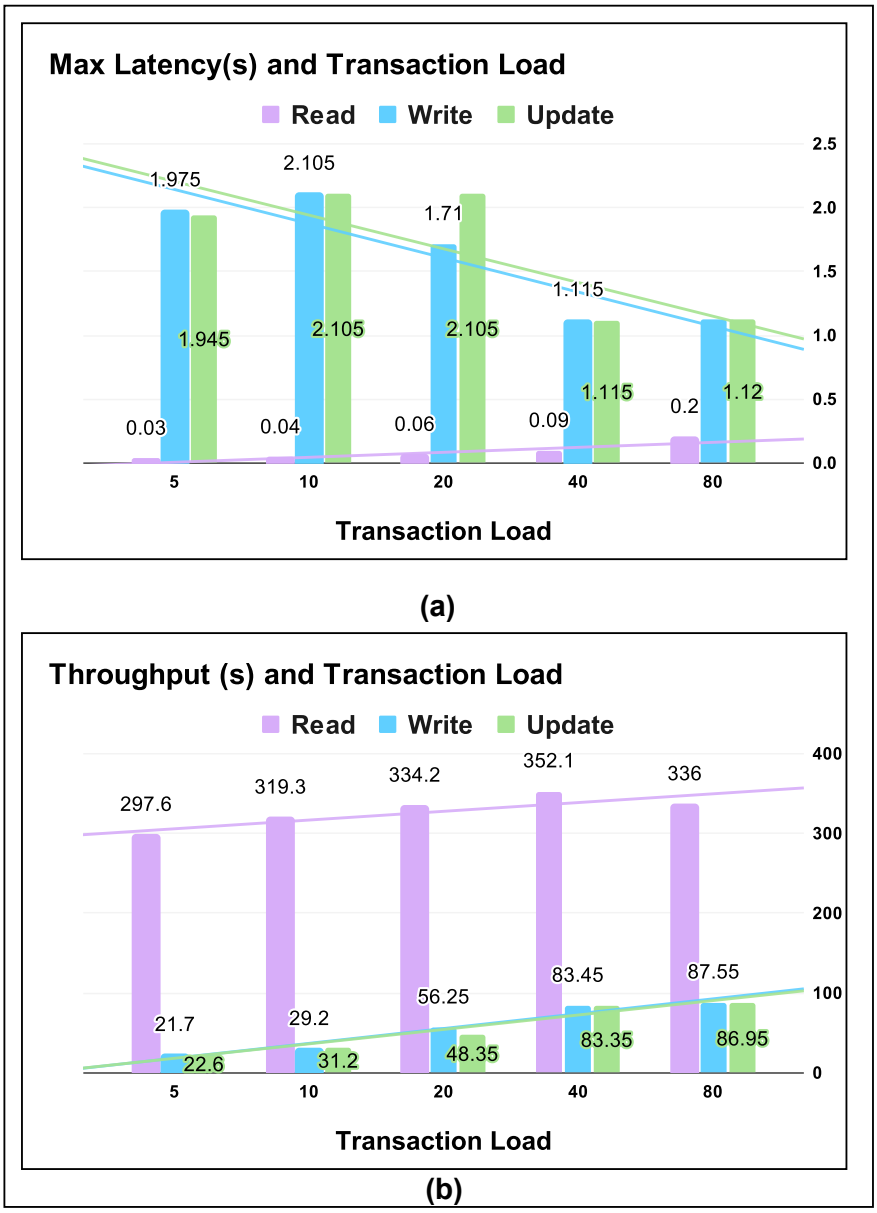


Figure 6.1: Performance of network on different types of operation

Table 6.1: Detailed Performance Report of the Blockchain Network

Operation Type	Transaction Load	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (s)
Write Operation	5	22.15	1.975	0.06	0.28	21.7
	10	30.45	2.105	0.065	0.23	29.2
	20	59.05	1.71	0.07	0.2	56.25
	40	83.95	1.115	0.07	0.29	83.45
	80	88.35	1.13	0.095	0.525	87.55
Update (Read+Write) Operation	5	23.1	1.945	0.065	0.265	22.6
	10	33.4	2.105	0.065	0.22	31.2
	20	53.75	2.105	0.075	0.215	48.35
	40	83.85	1.115	0.075	0.295	83.35
	80	87.6	1.12	0.08	0.53	86.95
Read Operation	5	298.2	0.03	0.01	0.01	297.6
	10	320	0.04	0.01	0.02	319.3
	20	335.3	0.06	0.01	0.03	334.2
	40	353.1	0.09	0.01	0.04	352.1
	80	336.8	0.2	0.01	0.08	336

throughput and minimize the latency of the network. A detailed overview of other performance parameters is displayed in Table 6.1. Other studies related to the performance of HLF found that it can handle up to 20,000 TPS with specific configurations [34].

Chapter 7

Conclusion and Future Work

Our work presents a blockchain-based solution for online job portals that address the issues of privacy and transparency in traditional database-based portals. Our solution uses Smart Contracts to protect the privacy of job seekers and ensure transparency for both recruiters and job seekers. It effectively addresses the shortcomings of traditional portals and provides a solution to the problems previously identified.

Blockchain technology has gained widespread adoption in the financial sector, particularly through the use of public networks such as Bitcoin and Ethereum. However, the use of blockchain in non-financial sectors, such as our proposed online job portal, is less explored. It is important to examine the potential for using blockchain in a permissioned environment to secure and protect data while maintaining its key benefits. Further research is needed to fully understand the potential and limitations of using blockchain technology in this way.

In the future, we plan to provide Decentralized Identifiers[1](DIDs) to job seekers as a way to verify their identities. This would allow job seekers to use their identity with multiple organizations without needing to go through the verification process multiple times. We also plan to compare the performance and suitability of other private blockchain frameworks for our system and conduct a detailed security analysis to ensure the safety and security of our proposed solution.

All the supplementary files and Smart Contract codes are available at `https://github.com/SATYAJIT1910/PTJSP`

[1] `https://www.w3.org/TR/did-core/`

References

[1] S. Vidros, C. Kolias, and G. Kambourakis, "Online recruitment services: another playground for fraudsters," *Computer Fraud & Security*, vol. 2016, no. 3, pp. 8–13, 2016.

[2] "Bengal recruitment scam: 21k candidates recruited illegally, cbi to court," *Business Standard*, Dec 2022.

[3] R. Mathisugan, "Online job portal," tech. rep., 2018.

[4] M. M. R. Prodhan and B. K. Saha, "Online job portal," tech. rep., 2017.

[5] M. Katariya, H. Shah, and N. Patel, "Online job portal," tech. rep., 2020.

[6] M. Pinjari, N. De, R. Kokne, A. Siddiqui, and D. Chitre, "Online job portal," *International Research Journal of Engineering and Technology (IRJET)*, vol. 6, Apr 2019.

[7] N. Karandikar, A. Chakravorty, and C. Rong, "Renewledger : Renewable energy management powered by hyperledger fabric," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, 2020.

[8] S. J. Moon, I. H. Park, B. S. Lee, and J. Ju Wook, "A hyperledger-based p2p energy trading scheme using cloud computing with low capabillity devices," in *2019 IEEE International Conference on Smart Cloud (SmartCloud)*, pp. 190–192, 2019.

[9] J. W. Kim, J. G. Song, H. W. Shin, and J. W. Jang, "Amm based p2p energy trading system using hyperledger fabric blockchain," in *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1747–1749, 2021.

[10] K. S. S. Wai, E. C. Htoon, and N. N. M. Thein, "Storage structure of student record based on hyperledger fabric blockchain," in *2019 International Conference on Advanced Information Technologies (ICAIT)*, pp. 108–113, 2019.

[11] N. Kumar S. and M. Dakshayini, "Secure sharing of health data using hyperledger fabric based on blockchain technology," in *2020 International Conference on Mainstreaming Block Chain Implementation (ICOMBI)*, pp. 1–5, 2020.

[12] T. Alshalali, K. M'Bale, and D. Josyula, "Security and privacy of electronic health records sharing using hyperledger fabric," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 760–763, 2018.

[13] S. B. Atreyapurapu, K. Amarendra, and M. M. Alishah, "Hyperledger fabric based medical record security," in *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 223–228, 2022.

[14] M. A. H. Wadud, T. M. Amir-Ul-Haque Bhuiyan, M. A. Uddin, and M. M. Rahman, "A patient centric agent assisted private blockchain on hyperledger fabric for managing remote patient monitoring," in *2020 11th International Conference on Electrical and Computer Engineering (ICECE)*, pp. 194–197, 2020.

[15] M. R. Amin, M. F. Zuhairi, and M. N. Saadat, "Transparent data dealing: Hyperledger fabric based biomedical engineering supply chain," in *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pp. 1–5, 2021.

[16] H. T. Le, T. T. L. Nguyen, T. A. Nguyen, X. S. Ha, and N. Duong-Trung, "Bloodchain: A blood donation network managed by blockchain technologies," *Network*, vol. 2, no. 1, pp. 21–35, 2022.

[17] Q. Tang, Z. Xia, N. Shu, X. Zuo, L. Zhao, S. Li, and S. Ren, "Research on epidemic material management method based on fabric blockchain," in *2022 4th International Conference on Communications, Information System and Computer Engineering (CISCE)*, pp. 623–626, 2022.

[18] J. Li, J. Yuan, and Y. Xiao, "A traditional medicine intellectual property protection scheme based on hyperledger fabric," in *2022 4th International Conference on Advances in Computer Technology, Information Science and Communications (CTISC)*, pp. 1–5, 2022.

[19] N. Ariffin and A. Z. Ismail, "The design and implementation of trade finance application based on hyperledger fabric permissioned blockchain platform," in *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pp. 488–493, 2019.

[20] Y.-T. Lee, J.-J. Lin, J. Y.-J. Hsu, and J.-L. Wu, "A time bank system design on the basis of hyperledger fabric framework," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 1–3, 2020.

[21] V. Aleksieva, H. Valchanov, and A. Huliyan, "Implementation of smart contracts based on hyperledger fabric blockchain for the purpose of insurance services," in *2020 International Conference on Biomedical Innovations and Applications (BIA)*, pp. 113–116, 2020.

[22] A. Poniszewska-Marańda, S. Rojek, and M. Pawlak, "Decentralized electronic voting system using hyperledger fabric," in *2022 IEEE International Conference on Services Computing (SCC)*, pp. 339–348, 2022.

[23] S. Vidwans, A. Deshpande, P. Thakur, A. Verma, and S. Palwe, "Permissioned blockchain voting system using hyperledger fabric," in *2022 International Conference on IoT and Blockchain Technology (ICIBT)*, pp. 1–6, 2022.

[24] P. Naveen, P. Manikandan, M. Maragatharajan, S. R. Ashok Kumar, M. Saravanan, and A. Puviarasu, "Traceability of tea powder supply chain based on hyperledger fabric," in *2022 3rd International Conference on Smart Electronics and Communication (ICOSEC)*, pp. 303–307, 2022.

[25] R. Shivers, M. A. Rahman, M. J. H. Faruk, H. Shahriar, A. Cuzzocrea, and V. Clincy, "Ride-hailing for autonomous vehicles: Hyperledger fabric-based secure and decentralize blockchain platform," in *2021 IEEE International Conference on Big Data (Big Data)*, pp. 5450–5459, 2021.

[26] N. Ullah, K. A. Al-Dhlan, and W. M. Al-Rahmi, "Kyc optimization by blockchain based hyperledger fabric network," in *2021 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*, pp. 1294–1299, 2021.

[27] N. Banoun and N. Diarra, "Authentication of mobile iot devices using hyperledger fabric blockchain," in *2021 Eighth International Conference on Software Defined Systems (SDS)*, pp. 1–6, 2021.

[28] N. Mishra and H. Levkowitz, "Performance evaluation of permissioned-based personal data vault implemented using hyperledger fabric v2.x," in *2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, pp. 138–145, 2022.

[29] C. Harris, "Improving telecom industry processes using ordered transactions in hyperledger fabric," in *2019 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, 2019.

[30] A. Bouras, H. Gasmi, A. Belhi, A. Hammi, and B. Aouni, "Enterprise information systems enhancement: A hyperledger fabric based application," in *2021 9th International Symposium on Digital Forensics and Security (ISDFS)*, pp. 1–5, 2021.

[31] Hyperledger, "Hyperledger architecture, volume 2."

[32] Hyperledger, "Peers."

[33] D. Ongaro and J. Ousterhout, "The raft consensus algorithm," *Lecture Notes CS*, vol. 190, 2015.

[34] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "Fastfabric: Scaling hyperledger fabric to 20,000 transactions per second," 2019.

Document Information

Analyzed document	ChainHire_v1.0.pdf (D166218718)
Submitted	2023-05-08 11:24:00
Submitted by	Dr. Abhishek Roy
Submitter email	abhishek.roy@adamasuniversity.ac.in
Similarity	5%
Analysis address	abhishek.roy.adamas@analysis.urkund.com

Sources included in the report

W URL: https://www.hyperledger.org/wp-content/uploads/2018/04/Hyperledger_Arch_WG_Paper_2_SmartContra...
Fetched: 2020-04-24 16:05:24 1

W URL: https://medium.com/coinmonks/hyperledger-fabric-blockchain-performance-benchmark-using-hyperle...
Fetched: 2022-11-22 08:29:28 1

Entire Document

Abstract Finding a suitable employment position is a crucial and difficult task. Nowa- days, Job seekers may instantly apply for multiple job openings using job search platforms. Personal information has to be exchanged as part of ev- ery job application. Sharing personal information across unsafe channels, however, increases the risk of data theft. Apart from that, attempts to tamper with hiring data are unfortunately common. Previously, traditional databases were only used for job search platforms, which do not provide sufficient transparency or protection against tampering. The incorporation of blockchain technology in job search portals can address these issues and provide a more transparent and secure process. Our proposed solution uses Hyperledger Fabric (HLF), an open-source blockchain framework, to create a secure and transparent job search platform. In this platform, both re- cruiters and job seekers can participate in a permissioned network, where smart contracts are used to ensure a transparent and privacy-friendly hiring process. To demonstrate the feasibility of this solution, we have implemented and deployed a prototype using Amazon Managed Blockchain Service. To understand the optimal configuration for our system, we tested the perfor- mance of our network using the Hyperledger Caliper tool. Although further research is necessary to fully understand the capabilities and limitations of using blockchain technology in job search portals. 1