Contents

1 Introduction **1**
- 1.1 problem definition . . . 1
- 1.2 Organization Profile . . . 2
- 1.3 Objective of the project . . . 4

2 LITERATURE SURVEY **5**
- 2.1 Initial Investigation . . . 5
- 2.2 Existing System . . . 6
  - 2.2.1 Methods adopted for conducting study . . . 8
  - 2.2.2 Basic functionalities of existing system . . . 8
  - 2.2.3 Users involved with the existing system . . . 9
  - 2.2.4 Pitfalls identified in the existing system . . . 9
  - 2.2.5 Feasibility study of implementing the new system . . . 9
- 2.3 Proposed system . . . 10
  - 2.3.1 Bockchain Technology . . . 10
  - 2.3.2 Functional Modules . . . 14
  - 2.3.3 Parties Involved . . . 14
  - 2.3.4 Software lifecycle technique adopted for the system . . . 14
- 2.4 Feasibility Study . . . 16
  - 2.4.1 Assumptions . . . 16
  - 2.4.2 Technical Feasibility . . . 16
  - 2.4.3 Social Feasibility . . . 17
  - 2.4.4 Economic Feasibility . . . 17
  - 2.4.5 Operational Feasibility . . . 17

3 SYSTEM ANALYSIS AND DESIGN **18**
- 3.1 Software Requirement Specification . . . 19
- 3.2 UML Diagrams . . . 20
  - 3.2.1 Activity Diagram . . . 21
  - 3.2.2 Class Diagram . . . 21
  - 3.2.3 Sequence Diagram . . . 23

3.2.4 Use Case Diagram 24
3.3 System Design 24
3.3.1 Input Design 26
3.3.2 Output Design 26
3.3.3 Database Design 27
3.3.4 Modular Design 31
3.3.5 Entity-Relationship Diagram 31
3.4 Tools And Platforms 32
3.4.1 Platform: ETHEREUM 32
3.4.2 Front end tool: HTML, JavaScript, CSS 34
3.4.3 Back end tool: Node.js, web3.js, solidity 35
3.4.4 Docker 40
3.4.5 TestRPC 42
3.4.6 Truffle 42
3.4.7 MongoDB 43
3.4.8 Linux 45
3.4.9 Microsoft Visual Studio 49

4 SYSTEM TESTING 51
4.1 Unit testing 51
4.1.1 Testing in Proposal Submission Module 52
4.1.2 Testing in Proposal Validation Module 52
4.2 Integration Testing 52
4.2.1 User Acceptance Testing 53
4.2.2 System Testing 53

5 SYSTEM IMPLEMENTATION 54
5.1 Implementation plan 54

6 CONCLUSION 56

7 REFERENCES 57

8 APPENDICES 58
8.1 Screen Shots 58

List of Figures

1.1 Organization Profile-logo 3

2.1 Client-Server model 7
2.2 Blockchain 11

3.1 Activity Diagram 22
3.2 Class Diagram 23
3.3 Sequence Diagram 25
3.4 Use Case Diagram 26
3.5 ENTITY-RELATIONSHIP DIAGRAM 33
3.6 Remix (a) 39
3.7 Remix (b) 39
3.8 Docker 40
3.9 Engine-component-flow 41
3.10 MongoDB 43
3.11 Environment:Remix(a) 47
3.12 Environment:Remix(b) 49
3.13 Visual Studio Code 50

8.1 Enroll Provider Details 58
8.2 Personal information 59
8.3 Provider details 59
8.4 practice information 60
8.5 Mailing information 60
8.6 Miscellaneous Information 61
8.7 Account Details 61
8.8 Submission of Details 62
8.9 Provider Login 62
8.10 Profile of provider(a) 63
8.11 Profile of provider(b) 63
8.12 Profile of provider(c) 64
8.13 Profile of provider(d) 64

8.14 Profile of provider(e) 65
8.15 Profile of provider(f) 65
8.16 Update and logout 66
8.17 PDM Login 66
8.18 Validation(a) 67
8.19 Validation(b) 67
8.20 Search for Doctor 68

List of Tables

3.1 Hardware Requirements 20
3.2 Software Requirement 20
3.3 RDBMS vs MongoDB 44

Chapter 1

Introduction

1.1 problem definition

A blockchain is a continuously growing list of records, called blocks, which are linked and secured using cryptography. Each block typically contains a hash pointer as a link to a previous block, a timestamp and transaction data. By design, blockchains are inherently resistant to modification of the data. It is "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way". For use as a distributed ledger, a blockchain is typically managed by a peer-to-peer network collectively adhering to a protocol for validating new blocks. Once recorded, the data in any given block cannot be altered retroactively without the alteration of all subsequent blocks, which requires collusion of the network majority. Decentralized consensus has therefore been achieved with a blockchain. This makes blockchains potentially suitable for the recording of payment details.

The project 'Blockchain in Electronic Health Record (EHR)' takes a use-case from insurance sector. The stakeholders of this project are insurance parties, providers and patients. The medical records,personal information, practice information,mailing information,miscellaneous information and electronic information of the provider are recorded into the blockchain by the hospitals/clinics, which holds a smart contract with the Insurance companies. This details entered by clinics are validated by the insurance agents called Provider Data Manager (PDM). These validations are also recorded as transactions in blockchain along with the time of validation. Since all these data are stored in blockchain, it ensures more security because of the immutability of blockchain. Thereby it eliminates a lot of difficulties for

clients as well as for the company employees.

Our vision of the application is to build an insurance platform with which integrate payer (insurance companies), PDM (Provider Data Manager) and the patients. Since personal health care records are more precious data item, we need to have good amount of security embedded in our application.

Currently we have implemented single payer system in our appplication ie, the doctor can register into the portal as an individual or an organization. The PDM can also validate the records which is already in the blockchain. The user (insurance companies) can search for records in the portal.

The main aim of this application is that protect the entire information using blockchain and high availability of providers information across multiple companies. The system will meet the present business environment and improve the business overall and security. a hospital/clinic can upload all the details of the provider into the blockchain and the provider can register in insurance company as individual. The provider data Manager can validate the details entered by the hospital without the fear of tampering and fraud issues. An advantage for the system is that it avoids the extra effort of communication between patient, insurance company and hospital and the provider. Another advantage of using the blockchain technology is that the data entered are stored as immutable transactions. So any changes on the data will be considered as transactions. Blockchain is a decentralized distributed structure.it provide high availability of information.

1.2 Organization Profile

UST Global is an American multinational provider of Digital, IT services and solutions, headquartered in Aliso Viejo, California, United States. Stephen J. Ross founded UST Global in 1998 in Laguna Hills. The company has offices in USA, India, Mexico, UK, Malaysia, Philippines, Singapore, Spain and Poland. UST Global specializes in Healthcare, Retail and Consumer Goods, Banking and Financial Services, Telecom, Media and Technology, Insurance, Transportation and Logistics and Manufacturing and Utilities. Sajan Pillai is the Chief Executive Officer of UST Global and a member of the company's board of directors.

Figure 1.1: Organization Profile-logo

Services

UST Global offers a host of services in areas like Digital, Consulting, Managed Innovation, Human Centered Design, Advanced Analytics, Algorithms, Cyber Security, Engineering services, Embedded Engineering Product development, Tech build, Application Development, Enterprise Quality Assurance, Infrastructure, e-Commerce, Business Intelligence, Data Management, and BPO services.

Innovation

The company has three "engines of innovation". "Infinity Labs" is a network of "innovation gyms" where professionals are invited to experiment without fear of mistakes, 'Innovation Hub' is UST's Idea Management System and 'Open Minds' is a collaboration space on the Web. Forrester Research, in its "Maximising Innovation from ERP Service Providers; Insights from Nasscom 2010" report, noted that: "Companies like UST Global allow their employees to spend part of their time in a research lab, developing user cases for new technologies. UST Global won the 'Most Innovative Emerging Corporation Award' at the International Business Awards 2012 held in New Delhi. The Kerala state government on November 6 2017 inked an MOU with UST Global along with Intel for exploring the possibility of transforming the state into an electronic hardware manufacturing hub.

Acquisition

In March 2012, UST Global acquired Andare, a company engaged in developing mobile solutions for large enterprise CRM applications. Andare's iDispatch product is an enterprise mobility solution for the field services industry. In May 2014, UST Global acquired Kanchi Technologies, a company with primary focus in Engineer Services.

Mexico Outsourcing

UST Global has tied up with Centro Fox, presidential library and learning Centre in Mexico, to facilitate entry into Latin America. Former Mexican President Vicente Fox, spearheads Centro Fox activities.

1.3 Objective of the project

- To implement blockchain technology in insurance sector.
- To keep trust between the stakeholders- insurance parties, clinics/hospitals and patients.
- To reduce the problems with storing data in databases (a centralised repository which is not secure).
- To maintain a decentralized distributed immutable ledger of transactions within all the stakeholders of the project.
- To limit the difficulties of clients and employees in submitting the details and verifying it.
- To store the details of the medical providers in the blockchain and to make the transaction details secure.
- To track the details of the medical providers so that it can be effectively used by insurance company.
- Reduce fraud activities and ensure smooth way of for handling transactions with transparency.

Chapter 2

LITERATURE SURVEY

2.1 Initial Investigation

For high availability of resource the system should use any of the distributed database for storing providers information.Blockchain provide secured distributed ledger.

THE FUTURE IS DECENTRALIZED
The potential of block chains to disrupt industrial sectors, commercial processes, governmental structures or economic systems seems to know no bounds. We suggest that the trans-formative power of block chain technology should not be seen as a threat to existing systems of governance; rather, it should be seen as an opportunity for national and international institutions to defend the rights of those they represent and to accelerate our collective progress towards meeting the United Nations' Sustainable Development Goals. Block chains can bring transparency to opaque or corrupt systems, and variability and immutability to commercial processes. They can bring security and resilience to vulnerable infrastructure, ensure individual privacy whilst guaranteeing autonomy, and encourage cooperation and engender trust where they are needed most. Block chains can ease the frictions that prevent a vast array of sustainability, humanitarian, and environmental initiatives from fulfilling their potential. This white paper explains how this unconventional technology works and how it is already being used to pursue conventional ends. It illustrates how block chains have brought new levels of efficiency and effectiveness to the fields of development aid, supply chain management, renewable energy, economic growth, and several others.[1]

2.2 Existing System

In the current scenario, Doctors who take provider registration from an insurance company need to submit their medical records,personal information,practice information,mailing information,miscellaneous information and electronic information directly to one or more insurance companies. Company employees need to verify it with the hospitals in-order to approve the registration. Sometimes this forms a recursive cyclic task.

Another major problem is that the companies use database for storing data. The problems with databases are they are centralized, not distributed and less secured. Data can be tampered easily and hence there is no security for the data. It is a big problem in financial sectors including banking and insurance.it uses client-server model database.existing system using HIPAA for securely storing the medical information. In existing system,the companies are using client-server model database for storing the information.these stored information are available only for that particular company.these are centralized not distributed.if a provider wan to update any details on his profile ,he should update it on every companies database.

The law has emerged into greater prominence in recent years with the proliferation of health data breaches caused by cyberattacks and ransomware attacks on health insurers and providers.

The act, which was signed into law by President Bill Clinton on Aug. 21, 1996, contains five sections, or titles.[2]

Title I: HIPAA Health Insurance Reform Title I protects health insurance coverage for individuals who lose or change jobs. It also prohibits group health plans from denying coverage to individuals with specific diseases and pre-existing conditions, and from setting lifetime coverage limits.

Title II: HIPAA Administrative Simplification Title II directs the U.S. Department of Health and Human Services (HHS) to establish national standards for processing electronic healthcare transactions. It also requires healthcare organizations to implement secure electronic access to health data and to remain in compliance with privacy regulations set by HHS.

Title III: HIPAA Tax-Related Health Provisions Title III includes tax-related provisions and guidelines for medical care.

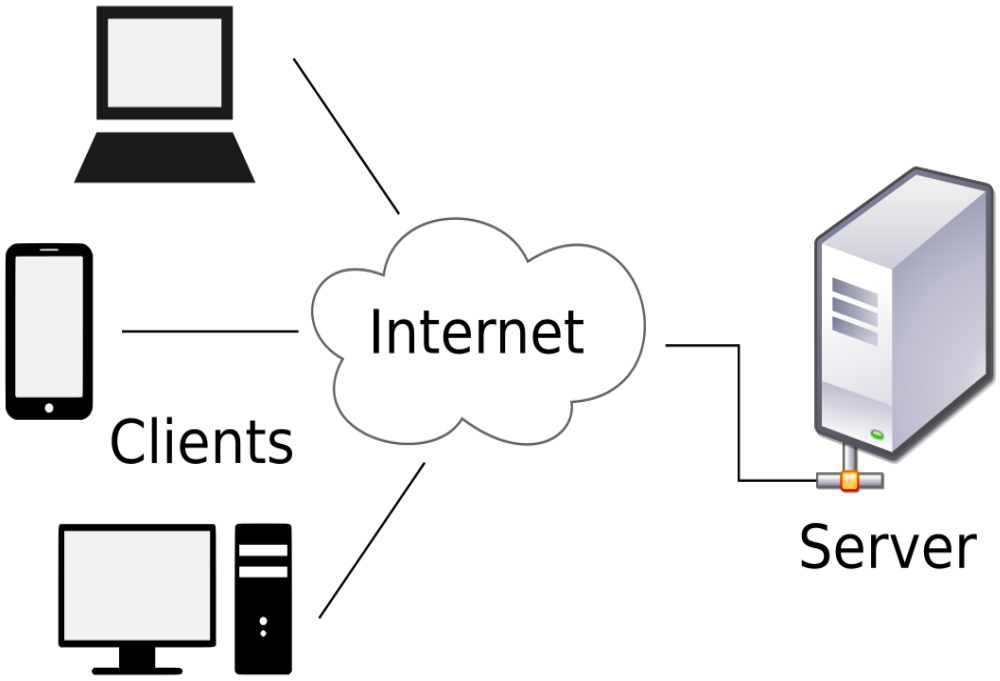

Figure 2.1: Client-Server model

Title IV: Application and Enforcement of Group Health Plan Title IV further defines health insurance reform, including provisions for individuals with pre-existing conditions and those seeking continued coverage.

Title V: Revenue Offsets Title IV further defines health insurance reform, including provisions for individuals with pre-existing conditions and those seeking continued coverage.
In healthcare circles, adhering to HIPAA Title II is what most people mean when they refer to HIPAA compliance. Also known as the Administrative Simplification provisions, Title II includes the following HIPAA compliance requirements:

- **National Provider Identifier Standard.** Each healthcare entity, including individuals, employers, health plans and healthcare providers, must have a unique 10-digit national provider identifier number, or NPI.
- **Transactions and Code Sets Standard.** Healthcare organi-

zations must follow a standardized mechanism for electronic data interchange (EDI) in order to submit and process insurance claims.

- **HIPAA Privacy Rule.** Officially known as the Standards for Privacy of Individually Identifiable Health Information, this rule establishes national standards to protect patient health information.
- **HIPAA Security Rule.** The Security Standards for the Protection of Electronic Protected Health Information sets standards for patient data security.
- **HIPAA Enforcement Rule.** This rule establishes guidelines for investigations into HIPAA compliance violations.

2.2.1 Methods adopted for conducting study

- Examining the existing system
  The existing system such as online insurance portals were studied to get a clear idea about the system. Most of such portals don't have the facility to submit the medical bills to the company to get back reimbursement. The current system used by most of the insurance companies are databases. These data bases are centralized and not distributed. Moreover they can be easily tampered.

- Interviews
  Interview is the most commonly used technique to collect information; face-to-face interviews. The purpose of interview is to find, verify, clarify facts, motivate end-users involved, identify requirements and gather ideas and opinions. It can give the personal views of the users. Most of the information collected were through the interview sections with the stakeholders.

- Questionnaires
  Interview is not a good option when the group of people from which the information collected is vast. Hence, Clients were asked to fill up some questionnaires from which we got a brief idea about the needs, necessaries and difficulties faced during the reimbursement of cost from companies .

2.2.2 Basic functionalities of existing system

In the existing system of Electronic Health Record (EHR), the data is stored in the database. The patients and Insurance companies (or) the hospitals

and insurance companies are linked directly so that there can be chances for fraud activities and since there is no security for the transaction details, it can be modified or destroyed.

2.2.3 Users involved with the existing system

The users currently involved are

1. Doctors
2. insurance companies

2.2.4 Pitfalls identified in the existing system

The problems with existing system i.e., databases. It uses client-server network architecture. Here, a user (known as a client) can modify data, which is stored on a centralized server. Control of the database remains with a designated authority (admin), which authenticates a client's credentials before providing access to the database. Since this authority is responsible for administration of the database, if the security of the authority is compromised, the data can be altered, or even deleted. Moreover it is a centralized system. So any loss of data or modification by unauthorized parties can lead to loss of important financial data.

2.2.5 Feasibility study of implementing the new system

- Economic feasibility
  The proposed system using is economically feasible, the system works absolutely free of cost and do not need any extra resources.
- Technical Feasibility
  The existing system can be developed by the resources available with the organization and it can also be merged with their existing system also.
- Operational Feasibility
  The proposed system will meet the present business environment and improve the business overall and security.

2.3 Proposed system

Proposed system is designed in such a way that all the drawbacks of existing will be eliminated. EHR Blockchain for insurance that is speedchain is an application implemented with blockchain technology. In the proposed system, a hospital/clinic can upload all the details of the providers such as medical records,personal information,practice information,mailing information,miscellaneous information and electronic information into the blockchain. The insurance company can validate the details entered by the hospital without the fear of tampering and fraud issues. An advantage for the proposed system is that it avoids the extra effort of communication between patient, insurance company and providers. Another advantage of using the blockchain technology is that the data entered are stored as immutable transactions. So any changes on the data will be considered as transactions. Blockchain is a decentralized distributed structure. All nodes belonging to the blockchain will have a copy of the blockchain so that even if one third of the data is lost, it offers 100 percent throughput. Data stored in blockchain are cryptographically hashed.Also Blockchain is a distributed ledger,so that the provider can easily update their information which are submitted into multiple insurance companies.Every Insurance companies uses the same Blockchain port for accessing the blockchain ledger.All the companies are connected together.Provide high reliability and availability.

2.3.1 Bockchain Technology

The Blockchain can be defined as a *'Smart Immutable Distributed Ledger'*. Blockchain is a continuously growing list of records called 'blocks' which form a chain and are secured using cryptography. Each block has a link to the previous block, and once a block is recorded, the data in a block cannot be altered. Blocks are the integral part of any blockchain platform. Blocks are referenced by their hash. The block is divided into two parts- the header part and the data part.
The header part contains details including,

1. BlockNumber - the unique block number (which will be 0 for the Genesis block).
2. PreviousHash - hash of the previous block's header (which is NULL for Genesis block)
3. DataHash - hash of the data segment of the current block.
4. Timestamp – It is the time at which the block is mined.

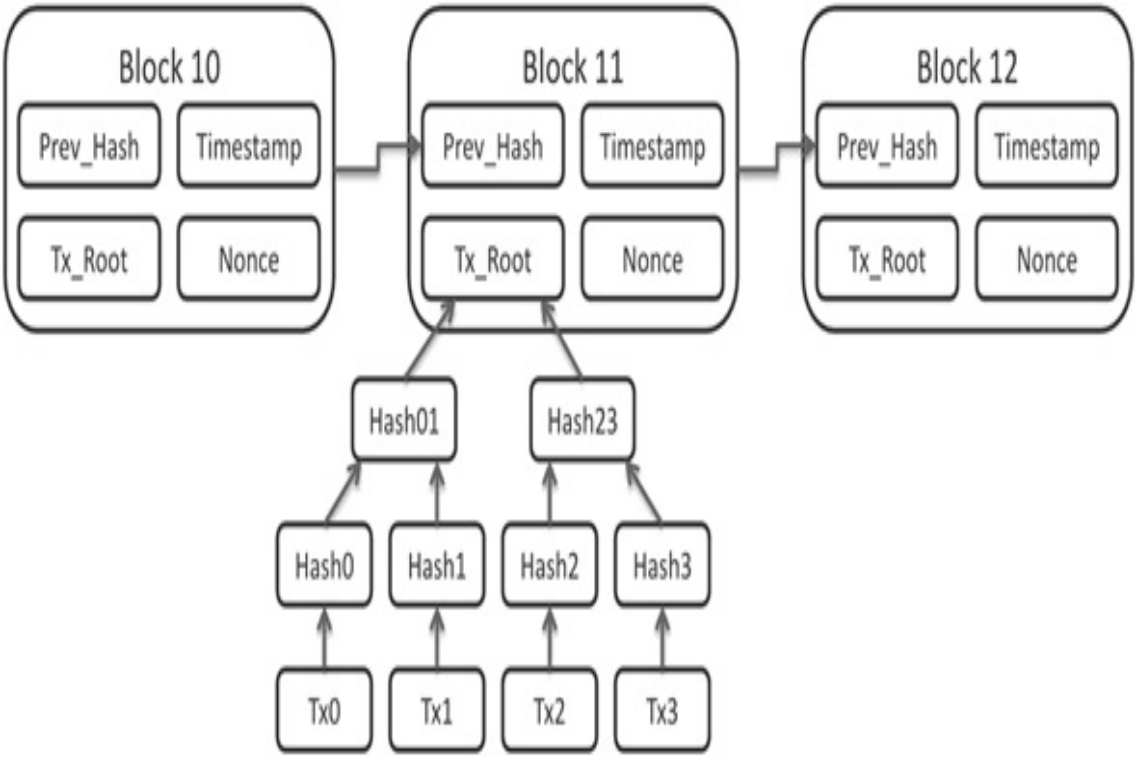

Figure 2.2: Blockchain

5. GasLimit – Scalar value equal to the current limit of gas expenditure per block.

6. GasUsed – Scalar value equal to the total gas used in transactions in this block.

7. Nonce - It is a number added to a hashed block that, when rehashed, meets the difficulty level restrictions

The first block in a chain is special and is called genesis block. The PreviousHash of the genesis blocks is set to NIL, whereas the PreviousHash of the next block holds SHA256 hash of BlockHeader of the previous block. New blocks are added to the end of the chain by *miners*. This process is called Mining.

Blockchain - Working

A blockchain ledger consists of two types of records: individual transactions and blocks. The first block consists of a header and data that pertains to

transactions taking place within a set time period. The block's timestamp is used to help create an alphanumeric string called a hash. After the first block has been created, each subsequent block in the ledger uses the previous block's hash to calculate its own hash. Before a new block can be added to the chain, its authenticity must be verified by a computational process called validation or consensus which is done by the parties called- 'minors'. At this point of the blockchain process, a majority of nodes in the network must agree the new block's hash has been calculated correctly. Consensus ensures that all copies of the distributed ledger share the same state.Once a block has been added, it can be referenced in subsequent blocks, but it cannot be changed. If someone attempts to swap out a block, the hashes for previous and subsequent blocks will also change and disrupt the ledger's shared state. When consensus is no longer possible, other computers in the network are aware that a problem has occurred and no new blocks will be added to the chain until the problem is solved. Typically, the block causing the error will be discarded and the consensus process will be repeated.[6]

Blockchain - Features

1. Enhanced Security
   Blockchain security methods use encryption technology. The basis for this are the so-called public and private "keys". A "public key" (a long, randomly-generated string of numbers) is a users' address on the blockchain. Data sent across the network gets recorded as belonging to that address. The "private key" is like a password that gives its owner access to their data or other digital assets. Store your data on the blockchain and it is incorruptible.

2. Immutability
   Immutability mechanisms of blockchain technologies lead to lowered cost of audit and regulatory compliance with improved transparency. With blockchain, unchangeable records can be created. This characteristic is the pillar that underpins the concept of building trust among strangers or even smart contracts – factors that have put this technology at the epicenter of a revolution that promises to transform the Internet and society. The technological foundation of this immutability is based on the use of cryptographic algorithms that make it possible to guarantee and verify the integrity of a data set. In other words, ensure that the data set has not been altered when it was created.

3. Distributed
   Distributed ledger is a type of database that is shared, replicated, and

synchronized among the members of a network. The distributed ledger records the transactions, such as the exchange of assets or data, among the participants in the network. Participants in the network govern and agree by consensus on the updates to the records in the ledger. No central, third-party mediator, such as a financial institution or clearing-house, is involved. Every record in the distributed ledger has a timestamp and unique cryptographic signature, thus making the ledger an auditable history of all transactions in the network. One implementation of distributed ledger technology is the open source hyper ledger blockchain.

4. Decentralized
   By storing data across its network, the blockchain eliminates the risks that come with data being held centrally. Its network lacks centralized points of vulnerability that computer hackers can exploit. The decentralized peer-to-peer blockchain network prevents any single participant or group of participants from controlling the underlying infrastructure or undermining the entire system. Participants in the network are all equal, adhering to the same protocols. They can be individuals, state actors, organizations, or a combination of all these types of participants.
   At its core, the system records the chronological order of transactions with all nodes agreeing to the validity of transactions using the chosen consensus model. The result is transactions that are irreversible and agreed to by all members in the network.

5. Robustness
   By storing blocks of information that are identical across its network, the blockchain cannot:

   (a) Be controlled by any single entity.

   (b) Has no single point of failure.

6. Transparent and In-corruptness
   The blockchain network lives in a state of consensus. Two important properties result from this:

   (a) Transparency: data is embedded within the network as a whole, by definition it is public.

   (b) It cannot be corrupted: altering any unit of information on the blockchain would mean using a huge amount of computing power to override the entire network.

2.3.2 Functional Modules

The main modules of the project are -

- **Proposal Submission Module**:
  The proposal module is used to enter the details of provider's medical records,personal information,practice information,mailing information,miscellaneous information and electronic information.All these entries are made by the employee of Hospital/Clinic. These entries are added to blockchain as Transactions along with its timestamp.

- **Proposal Validation Module**:
  Proposal Validation module is for ensuring that the particular transaction is a valid transaction or not. The Data Provider Manager (PDM) who is a part of the insurance company is responsible for validating the submitted transactions and these details are added to the Blockchain. He can reject or approve the submitted transactions. Whatever be the response, it is added as a transaction even if he reject the proposal.

- **Provider search Module**:
  Every common user can Search for the providers according to specialization and locality

2.3.3 Parties Involved

There are 3 parties involved in the project.

1. Provider - The providers are the Doctors, whose details are stored in the transaction.

2. Payer - Here, the payer is the Insurance Company

3. Provider Data Manager – The PDM acts from the part of the Insurance Company. It verifies and add each transaction to the block.

2.3.4 Software lifecycle technique adopted for the system

Agile Model

Agile development model is also a type of Incremental model. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained. It is used for

time critical applications. Extreme Programming (XP) is currently one of the most well known agile development life cycle model.
Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like -

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

Advantages of Agile model

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

2.4 Feasibility Study

The first step involved in computerizing a system is to conduct a feasibility study to determine whether it would be possible to install a computer to process the data, keep the records, etc. and whether the cost of such computerization would justify the benefits acquired in terms of increased efficiency and reduction of workload of all the employees concerned. The objectives of this stage are to provide solutions to the stated problems usually in the form of specification to meet the user requirements and to make recommendations for a new computer based system. Feasibility study is a major step in software life cycle. It focuses on three major questions: About the user's needs and way in which candidate system meet them. About available resources. About the likely impact of the new system. The proposed system must be technically feasible and their impact on the organization and staff must be accessed. If compatible, social and technical systems can be devised. Then they must be tested for economic feasibility.

2.4.1 Assumptions

- Estimation is done only for the first stage of the project.
- Not much change is expected to occur in the requirements once the design is finalized.
- The estimation is done on the currently designed system.
- Code generation software to aid in the coding and generation of forms are available.
- The hardware problems will be minimum because of availability of UPS etc.

2.4.2 Technical Feasibility

Technical feasibility centers around the existing computer system-hardware and software and to what extend it can support the proposed system. The question to be answered is 'whether the organization is technically sound to operate the system'. The existing system is enough to support the Inventory and Financial Management System.

2.4.3 Social Feasibility

The assessment of social feasibility will be done alongside with technical feasibility. The system is developed in such a way that users can use it very flexibly. Large volumes of data can be processed and various reports are produced at a faster rate.

2.4.4 Economic Feasibility

Economic analysis is the most frequently used method for evaluating the effectiveness of a candidate system. Most commonly known as the Cost/Benefit. Analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with the cost.

2.4.5 Operational Feasibility

The proposed system will meet the present business environment and improve the business overall and security.

Chapter 3

SYSTEM ANALYSIS AND DESIGN

The first phase of software development is system study analysis. The importance of system analysis phase is the establishment of the requirements for the system to acquire developed and installed. Analyzing the project to understand the complexity forms the vital part of the system study. Problematic areas are identified and information is collected. Fast finding or gathering is essential to any analysis of requirements. It is also highly essential that the analyst familiarize himself with the objectives, activities and functions of organizations in which the system is to be implemented. System study involves studying the ways of how a reminder could be useful for an individual with the goal of determining how to make it better. For this, the system analyst should develop an alternative system and evaluate it in terms of cost, benefit and feasibility. The terms analysis, design and development are used in to that sequence because in practice this sequence of steps is used to construct computer based information system.
System analysis includes investigation and possible changes to the existing system. Analysis is used to gain and understanding of the existing system and what is required of it. At the conclusion of the system analysis there is the system description and set of requirements for a new system. If there is no such existing system then analysis only defines the requirements. This new system may build afresh or by changing the existing system. Development begins by defining a model of the new system and continues this model to a working system. The model of the system shows what the system must do to satisfy these requirements. Finally data modes are converted to a database and processed to user procedures and computer programs.

3.1 Software Requirement Specification

Requirements

1. Multi Payer Nodes:

   The portal should be configured to enter multiple payers in the portal. The payer companies such as Anthem, Geico, AllState etc can be able to participate in the portal.

2. Multi payer Search:

   The portal should also implement multipayer based search. A PDM can be registerd with one or more Payer companies. A PDM should be able to search only the patient records belonging to the particular company.

3. Payer Approval Module:

   When a PDM is registed into the module, the status should be pending. The Payer will approve the PDM through any algorithm or a voting mechanism.

4. System Performance:

   Since v1.0 is having performance issues we need special care in building the performance enchancing modules or mechanism. Can include Hashgraph consensus mechanism.

5. Patient Search:

   Patients can search for doctors and payers in his state. Search for policies and register with doctors.

Hardware Requirements

Table 3.1: Hardware Requirements

Processor	: Intel processor
Operation system	: Ubuntu 16
RAM	: 2 GB
Hardware Devices	: Keyboard with Mouse
Hard Disk	: 10GB or more
Display	: Standard Output Display

Software Requirement

Table 3.2: Software Requirement

Operating system	Ubuntu
Front End	HTML,CSS,JAVASCRIPT
Back End	Node.js,MongoDB
Design Tool	Visual studio code,Remix
Documentaion	Latex
Languages	solidity,java script

3.2 UML Diagrams

System design focuses on the final system and the process by which it is developed. It leads to transition from a user-oriented document which means system proposal a programmer-oriented document. The design methodology followed for this project is the bottom-up design strategy. In this approach, the basic sets of elements are individual modules. Each module is developed individually as a separate project and adds the modules into this as a reference if necessary. The benefit of the design is that it permits the review of the modules during the system development process. We prepare UML diagrams to understand a system in better and simple way. A single diagram is not enough to cover all aspects of the system. So UML defines various kinds of diagrams to cover most of the aspects of a system.[4] Unified modeling language (UML for short), is a standardized language accompanied by a specific set of symbols, used in the field of software engineering among others. UML diagrams make it possible to depict complex processes, systems and architectures in a way that anyone familiar with the unified modeling language

will understand. They can be divided into two main categories; structure diagrams and behavioral diagrams. Some of the UML diagrams are:

3.2.1 Activity Diagram

An activity diagram shows the sequence of steps that make that make up a complex process, such as an algorithm or work flow. An activity diagram shows flow of control, similar to a sequence diagram, but focuses on operations rather than on objects. Activity diagrams are most useful during the early stages of designing algorithms and workflows. The diagram comprised of the following model elements.
The start symbol represents the beginning of a process or workflow in an activity diagram. It can be used by itself or with a note symbol that explains the starting point. The activity symbol is the main component of an activity diagram. These shapes indicate the activities that make up a modeled process.The connector symbol is represented by arrowed lines that show the directional flow, or control flow, of the activity. An incoming arrow starts a step of an activity; once the step is completed, the flow continues with the outgoing arrow.
The join symbol, or synchronization bar, is a thick vertical or horizontal line. It combines two concurrent activities and re-introduces them to a flow where only one activity occurs at a time.
A fork is symbolized with multiple arrowed lines from a join. It splits a single activity flow into two concurrent activities.
The decision symbol is a diamond shape; it represents the branching or merging of various flows with the symbol acting as a frame or container.
The end symbol represents the completion of a process or workflow.

3.2.2 Class Diagram

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. Class diagram consists of classes, interfaces, associations and collaboration. Class diagrams basically represent the object oriented view of a system which is static in nature. Class diagram represents the object orientation of a system. So it is generally used for development purpose. This is the most widely used diagram at the time of system construction. Class diagram contains following components, Class box: The class name should write at the top section in boldface. Attributes are listed in the middle section followed by optional details (type & default

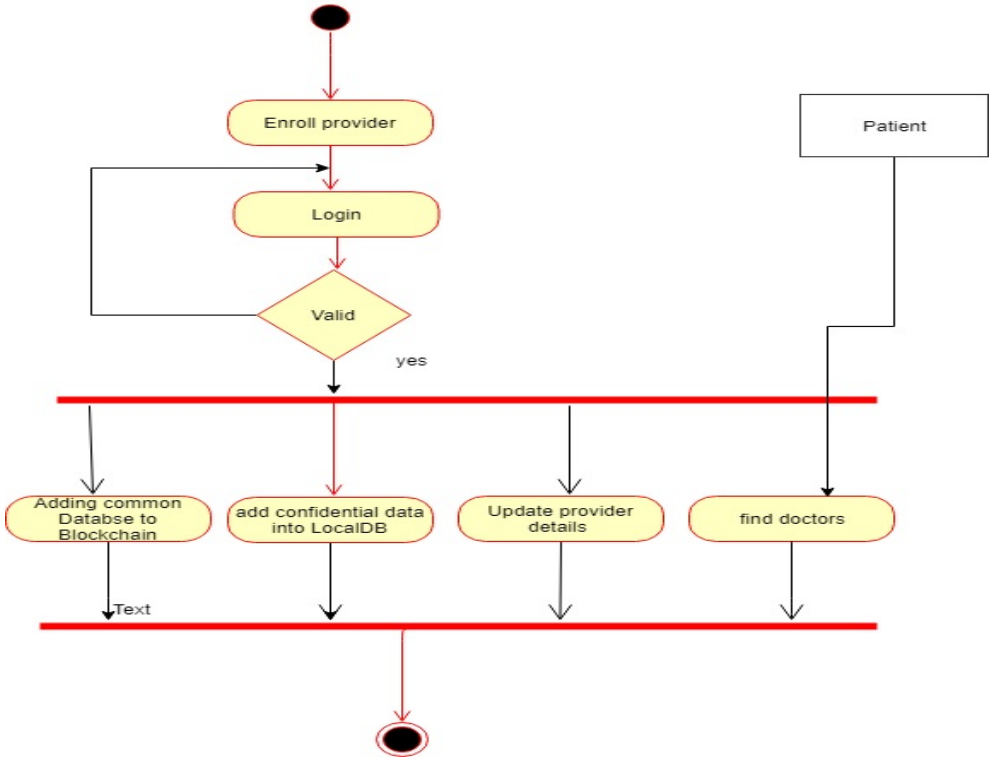

Figure 3.1: Activity Diagram

value). Type is preceded by a colon and default value is preceded by an equal sign. Operations are at the lower section followed by optional details (argument list & result type).

Dependency: Dependency is a relationship between two things in which change in one element also affects the other one.

Generalization: Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes inheritance relationship in the world of objects.

Aggregation: "is part of"

- symbolized by a clear white diamond

Composition: "is entirely made of"

- stronger version of aggregation
- the parts live and die with the whole

- symbolized by a black diamond

An enumeration is a data type whose values are enumerated in the model as user-defined enumeration literals. An enumeration may be shown using the classifier notation (a rectangle) with the keyword ¡¡enumeration¿¿. The name of the enumeration is placed in the upper compartment. A list of enumeration literals may be placed, one to a line, in the bottom compartment.

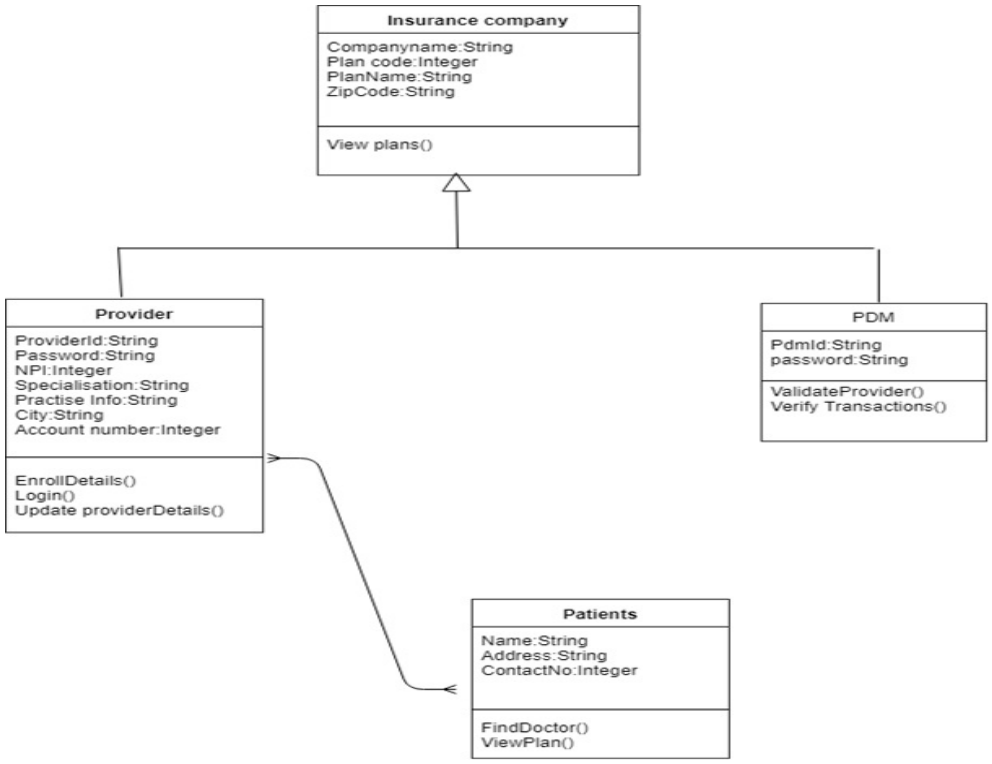

Figure 3.2: Class Diagram

3.2.3 Sequence Diagram

A Sequence Diagram shows the participants in an interaction and the sequence of messages among them. A sequence diagram shows the interaction of a system with its actors to perform all or part of a use case. A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. The diagram comprised of the following model elements.

Class Roles or Participants

Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Activation or Execution Occurrence

Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin gray rectangle placed vertically on its lifeline.
Execution (full name - execution specification, informally called activation) is interaction fragment which represents a period in the participant's lifetime when it is

- executing a unit of behavior or action within the lifeline
- sending a signal to another participant
- waiting for a reply message from another participant

The duration of an execution is represented by two execution occurrences - the start occurrence and the finish occurrence.

3.2.4 Use Case Diagram

The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose. Because other four diagrams (activity, sequence, collaboration and State chart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.
The actors included are the admin of Hospital/Clinic and the Provider Data Manager, who is a part of the Insurance Company. The use-cases includes adding Patient details and Payment details.

3.3 System Design

Design of a system can be defined as the process of applying various techniques and principles for the purpose of defining a device, a process or a

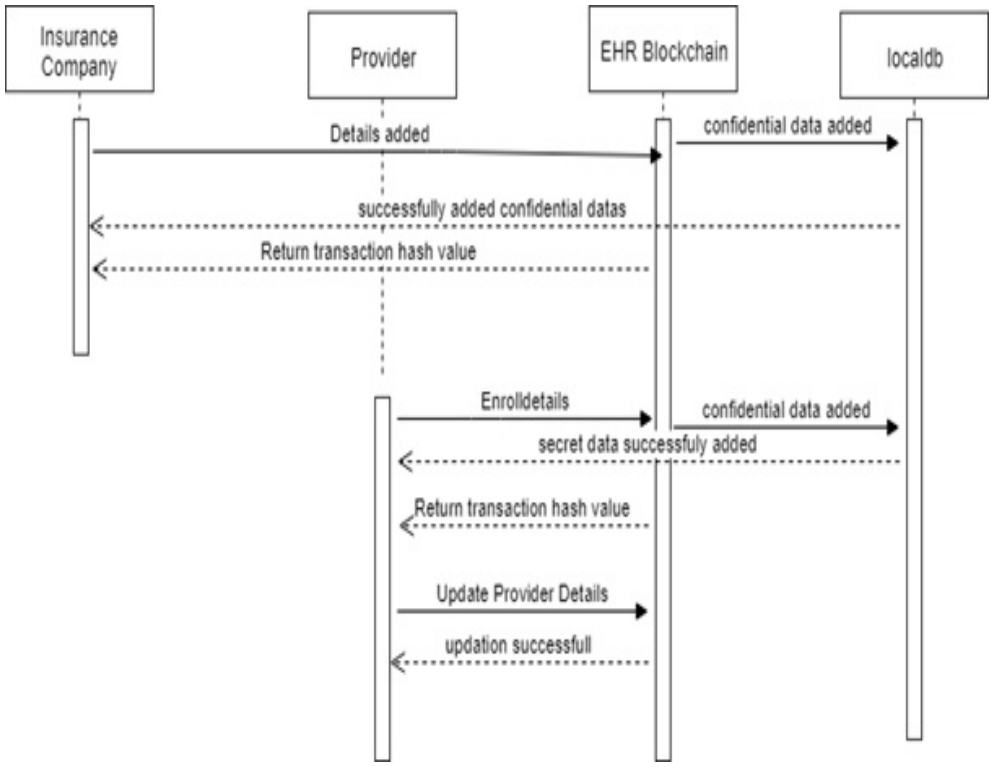

Figure 3.3: Sequence Diagram

system in sufficient detail to permit its physical realization. Thus system design is a solution, "how to" approach to the creation of a new system. This important phase provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. The design step produces a data design, an architectural design and a procedural design. The data design transforms the information domain model created during analysis into the data structures that will be required to implement the software. The architectural design defines the relationships among major structural components into a procedural description of the software. Source code is generated and testing is conducted to integrate and validate the software. From the project management point of view software design is conducted in two steps, preliminary design is concerned with the transformation of requirements into data and software architecture. Detailed data structure and algorithmic representation for software

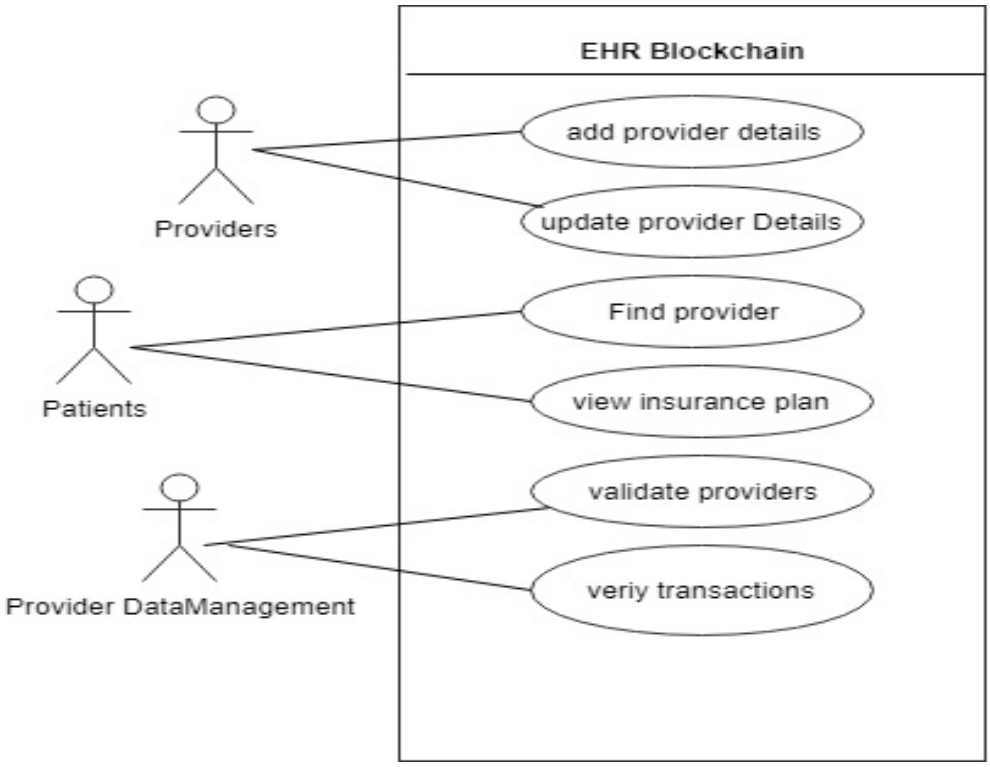

Figure 3.4: Use Case Diagram

3.3.1 Input Design

Input design is a part of overall system design, which requires very careful attention. If data going into the system is incorrect, then the processing and output will magnify these errors.
Thus the designer has a number of clear objectives in the different stages of input design:

- To produce a cost effective method of input
- To achieve the highest possible level of accuracy
- To ensure that input is acceptable to and understood by the user

3.3.2 Output Design

At the beginning of output design various types of outputs (external, internal, operational, interactive and turnaround) are defined. Then the format, content, location, frequency, volume and sequence of the output are specified.

The content of the output must be defined in detail. The system Analyst has two specific objectives at this stage.
To interpret and communicate the results of the computer part of a system to users in a form that they can understand and which meets their requirements. To communicate the output design specifications to programmers in a way, which is unambiguous, compressive and capable of being translated into a programming language.

3.3.3 Database Design

1.MongoDB

The database design is a logical development in the methods used by the computers to access and manipulate data stored in the various parts of the computer systems. Database is defined as an integrated collection of data. The overall objective in the development of database technology has been to treat data as an Organizational resource and as an integrated whole. The main objectives of database are data integration, data integrity and data independence.
MongoDB is a cross-platform,open-source, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.
Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.
A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.[5]

MongoDB Cheat Sheet

Connect to a database after starting mongo client

```
$ mongo use dbname
```

Call this command before using a specific database. This command also creates the database, but the new database is only save when you insert the first document in a collection.
Connect to a particular database when starting mongo client

```
$ mongo dbname
```

Drop a particular database

```
use dbname db.dropDatabase()
```

List all databases

```
show databases
```

List all collections of a database

```
use dbname show collections
```

Get the status of a particular database

```
db.status()
```

As of version 3.2.10, this commands lists an object like the following:

```
{
  "db" : "dbname",
  "collections" : 0,
  "objects" : 0,
  "avgObjSize" : 0,
  "dataSize" : 0,
  "storageSize" : 0,
  "numExtents" : 0,
  "indexes" : 0,
  "indexSize" : 0,
  "fileSize" : 0,
  "ok" : 1
}
```

By default the data size will show up in bytes, but if you want it in KB, use this

```
db.stats(1024).
```

For MB, use

```
db.status(1024*1024)
```

List the current connections to a mongodb server

```
db.serverStatus().connections
```

CRUD Operations

Insert a document

```
db.movies.insertOne({"title":"New Movie", "year":2010, "imdb": "aa
```

This command inserts a document into a movies collection Count the number of documents in a given collection

```
db.movies.count()
```

List documents in a collection without specifying any constraints

```
db.movies.find() db.movies.find().pretty()
```

List documents in a collection using a cursor

```
var cs = db.movies.find() cs.hasNext() cs.next()
```

Finding and Sorting

```
sort({field: 1|-1}): 1 = ascending; -1 = descending
```

Aggregation

```
{
user name: "John", password : "US", children: [{
name: "Rachel", age: 10, watched_movies: [{
      title: "movie 1"
    },
    {
      title: "movie 2"
    }]
  }]
}
```

Given the above document, how would you get the number of watched movies by John's children?

```
db.things.aggregate([
        { $match: { name: "John" } },
        { $unwind: "$children" },
        { $unwind: "$children.watched_movies" },
```

```
        { $group: { _id: null, count: { $sum: 1 } } }
])
```

Get a cursor, add all documents in an array and call tojson() to print them

```
records = [];
var cursor = db.someCollection.find({}, {}).limit(100); while(curs
records.push(cursor.next())
}
print(tojson(records))
```

Dump, Import, Export

Import a JSON file containing a collection of companies into a database called mclass

```
sudo mongoimport --db mclass --collection companies --file compani
```

2.Blockchain

There are a few ways that a Blockchain can be used in distributed storage software. One of the most common is to:

1. Break up data into chunks.
2. Encrypt the data so that you are the only one with access to it.
3. Distribute files across a network in a way that means all your files are available, even if part of the network is down.

Essentially, instead of handing your files to a company like Amazon or Microsoft, you distribute it across a network of people all over the world. The cloud is shared by the community, and nobody can read or tamper with anyone else's sensitive data. In other words, you stay in control. This could also be useful in public services to keep public records safe, available, and decentralized.
Another model is to just save a cryptographic signature of a document or file on a Blockchain. This would give users a way to ensure a file is untampered, without needing to save the entire file on the Blockchain. When you look at a file, you can guarantee that it is the same version of the document that existed at another time.
Smart contracts can also be used with Blockchains. These ensure that certain transactions happen when certain conditions are met, meaning records can be programmed to be changed or updated automatically.[7]

Advantages and Benefits

There are some awesome benefits to using a Cloud storage model. Including:

- Peer to peer networks can make download speeds lightning fast (similar to torrenting).
- Your data is distributed all over the world, so it is highly available when you need it.
- You don't have to worry about anyone else having access to any of your data.
- It can be extremely cheap – as low as $2 per TB per month.
- With a large network with many reliable participants, clever techniques could mean a very low level of redundancy factor, even as low as 1.5.
- The immutable nature of Blockchain records means you can know if a file is accurate and unaltered.

3.3.4 Modular Design

Provider module

- Registration and enrolling of provider Details
- Adding the common datas into blockchain and confidential data into Local Database
- Update provider details

Provider search

- Search for the providers according to specialization and locality

3.3.5 Entity-Relationship Diagram

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases. An ER model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of an entity. Some data fields in these tables point to indexes in other tables; such pointers are the physical implementation of the relationships. Components of E-R Diagrams are following,

Entity

Entities are represented by means of rectangles. Rectangles are named with the entity set they represent. Entities in a school database

Attributes

Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).

Simple Attributes

If the attributes are composite, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse. Multivalued attributes are depicted by double ellipse. Derived attributes are depicted by dashed ellipse.

Relationship

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

Binary Relationship and Cardinality

A relationship where two entities are participating is called a binary relationship. Cardinality is the number of instance of an entity from a relation that can be associated with the relation.

3.4 Tools And Platforms

3.4.1 Platform: ETHEREUM

Ethereum is a platform that is intended to allow people to easily write decentralized applications (Dapps) using blockchain technology. A decentralized application is an application which serves some specific purpose to its users, but which has the important property that the application itself does not depend on any specific party existing. Rather than serving as a front-end for selling or providing a specific party's services, a Dapp is a tool for people and organizations on different sides of an interaction used to come together

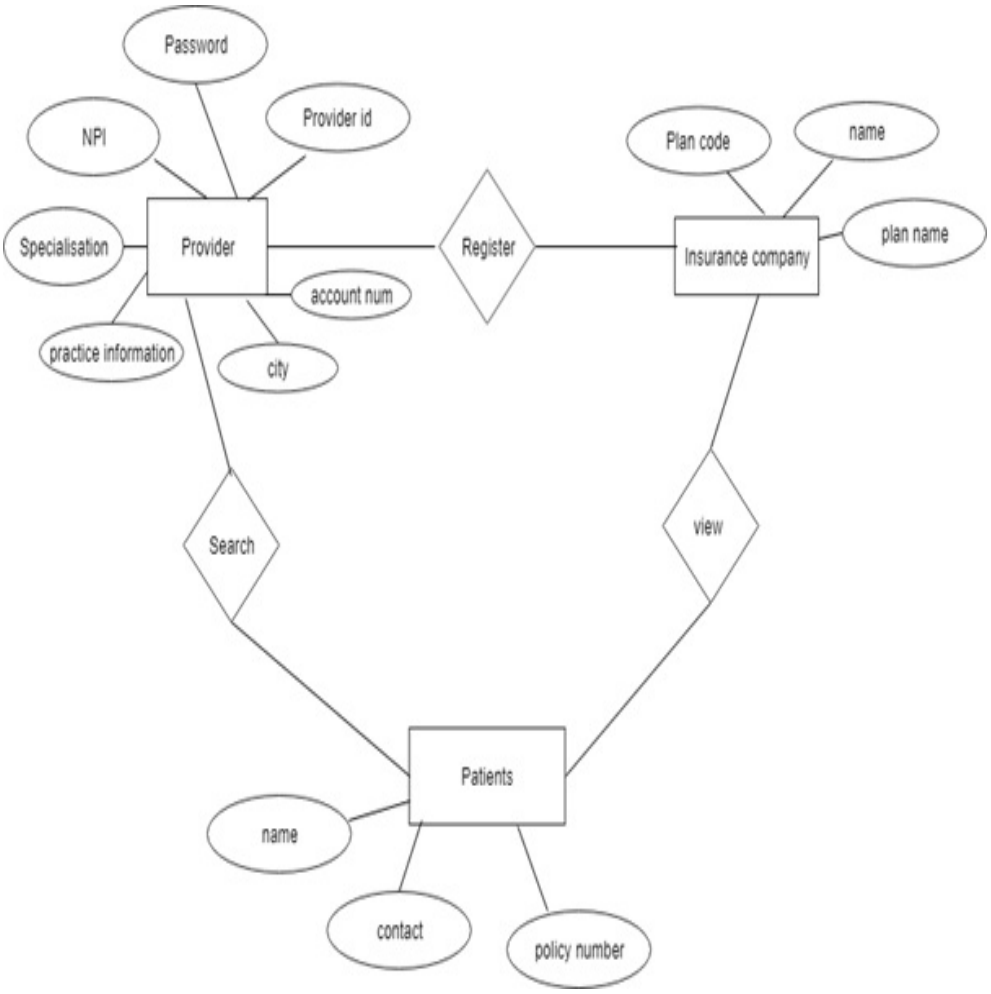

Figure 3.5: ENTITY-RELATIONSHIP DIAGRAM

without any centralized intermediary.
The Ethereum blockchain can be alternately described as a blockchain with a built-in programming language, or as a consensus-based globally executed virtual machine. The part of the protocol that actually handles internal state and computation is referred to as the Ethereum Virtual Machine (EVM). From a practical standpoint, the EVM can be thought of as a large decentralized computer containing millions of objects, called "accounts", which have the ability to maintain an internal database, execute code and talk to each other.
There are two types of accounts:

1. Externally owned account (EOAs): an account controlled by a private

key, and if you own the private key associated with the EOA you have the ability to send ether and messages from it.

2. Contract: an account that has its own code, and is controlled by code.

By default, the Ethereum execution environment is lifeless; nothing happens and the state of every account remains the same. However, any user can trigger an action by sending a transaction from an externally owned account, setting Ethereum's wheels in motion. If the destination of the transaction is another EOA, then the transaction may transfer some ether but otherwise does nothing. However, if the destination is a contract, then the contract in turn activates, and automatically runs its code.

3.4.2 Front end tool: HTML, JavaScript, CSS

Hypertext Markup Language (HTML) is the standard mark-up language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web.[3] Web browsers receive HTML documents from a web server or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.
JavaScript , often abbreviated as JS, is a high-level, dynamic, weakly typed, prototype-based, multi-paradigm and interpreted programming language. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production. It is used to make WebPages interactive and provide online programs, including video games. The majority of websites employ it, and all modern web browsers support it without the need for plug-ins by means of a built-in JavaScript engine. Each of the many JavaScript engines represent a different implementation of JavaScript, all based on the ECMA Script specification, with some engines not supporting the spec fully, and with many engines supporting additional features beyond ECMA.
As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.
Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in

HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.
CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colors, and fonts.[3] This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

3.4.3 Back end tool: Node.js, web3.js, solidity

Nodejs

Node.js an open-source, cross-platformJavaScript run-time environment that executes JavaScript code server-side. Historically, JavaScript was used primarily for client-side scripting, in which scripts written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser. Node.js lets developers use JavaScript for server-side scripting—running scripts server-side to produce dynamic web page content beforethe page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server side and client side scripts.
Though is the conventional filename extension for JavaScript code, the name "Node.js" does not refer to a particular file in this context and is merely the name of the product. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).The Node.js distributed development project, governed by the Node.js Foundation, is facilitated by the Linux Foundation's Collaborative Projects program. **Node Package Manager** - npm opens up an entire world of JavaScript talent for you and your team. It's the world's largest software registry, with approximately 3 billion downloads per week. The registry contains over 600,000 packages (building blocks of code). Open-source developers from every continent use npm to share and borrow packages, and many organizations use npm to manage private development as wellnpm consists

of three distinct components:

- the website
- the Command Line Interface (CLI)
- the registry

Use the website to discover packages, set up profiles, and manage other aspects of your npm experience. For example, you can set up Orgs (organizations) to manage access to public or private packages.
The CLI runs from a terminal. This is how most developers interact with npm.The registry is a large public database of JavaScript software and the meta-information surrounding it.
Use npm to

- Adapt packages of code to your apps, or incorporate packages as they are.
- Download standalone tools you can use right away.
- Run packages without downloading using npx.
- Share code with any npm user, any where.
- Restrict code to specific developers.
- Form Orgs (organizations) to coordinate package maintenance, coding, and developers.
- Form virtual teams by using Orgs.
- Manage multiple versions of code and code dependencies.
- Update applications easily when underlying code is updated.
- Discover multiple ways to solve the same puzzle.
- Find other developers who are working on similar problems and projects.

Node Version Manager (NVM) -Node Version Manager is a tool that allows programmers to seamlessly switch between different versions of Node. You can install each version with a single command and set a default via the command line interface.

The Benefits of Node Version Manager -Aside from saving time and effort, being able to switch between Node versions has a few significant benefits. For instance, let's say a tool claims to support just one specific version of Node.js, but you want to see if it works with another version that you prefer. If you encounter bugs, Version Manager makes it simple to switch node versions for quick troubleshooting. Otherwise, you'd have to continuously uninstall and reinstall node versions and their global packages to switch back and forth.

If you're using a tool that's still in the developer preview phase, Node Version Manager gives you a way to get around the hurdles that can come with each update. For example, when SharePoint Framework was is the preview phase, each drop came with a new version of the Yeoman generator. Therefore, you could be in a situation where you're able to work locally with the latest developer preview, but you'd have to either wait days or even weeks for the supported libraries to become available in Office 365, or you'd just have to hold off installation until the libraries arrived. With Node Version Manager, you can easily keep two versions going with the old and new generators.

If you attend an event like DevKitchen or a workshop where you're given some tools in a pre-release state, Version Manager makes it easy to store and save them for later. Instead of using your main Node.js install, you can quickly switch to a different version of Node.js, work in that environment and save it for later. That way, you will have easy access to those tools later on even if they are unavailable elsewhere.

Knowing how to use Node Version Manager can help you save a lot of time, which is usually extremely important to development teams. Keep this guide handy just in case you ever need a refresher, but with such a slight learning curve, you'll likely memorize the commands fairly quickly

Web3.js

This is the Ethereum compatible JavaScript API which implements the Generic JSON RPC spec. It's available on npm as a node module, for bower and component as an embeddable js and as a meteor.js package. Web3.js is the official Ethereum Javascript API. You use it to interact with your Ethereum smart contracts.web3.js is a collection of libraries which allow you to interact with a local or remote ethereum node, using a HTTP or IPC connection.The web3.js library is a collection of modules which contain specific functionality for the ethereum ecosystem.

Solidity

Solidity is a contract-oriented, high-level language for implementing smart contracts. It was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM).Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features.
This post is a list of Solidity tips and tricks I learned the hard way, by going through Solidity on my own and brute forcing my way through learning. This is really the only way to do it right now if you are learning on your own. It will help to identify the easiest tools and tips in order to get your first dApp deployed on the testnet using Heroku.
To start coding a new language, you want the easiest development environment that will help you through bugs so you can learn from your mistakes quickly as you go. Start by using the Remix Solidity IDE "`https://remix.ethereum.org`."

- It is great for debugging, and gives you an option to step through the contract which is extremely helpful. I found it very useful to use the debugger throughout the whole development of my dApp, but at the start it was 100
- You should use remix until you get your dApp to compile with no bugs.
- There are three options with remix for injecting a blockchain instance into the browser, you can use the Javascript Virtual Machine (JVM), an injected web3 instance (like the chrome extension Metamask), or connecting to a web3 provider (likeInfura). The JVM is the way to go when you are a beginner. It is much more forgiving, it ignores gas limits, it gives you an unlimited amount of ether to play with between 5 accounts, and it is much faster to debug with.

  Make sure the environment is JavaScript VM when starting with your first dApp.Start using Event Logs as much as possible to debug your code. They are the closest thing to console.log() from Javascript, as they can notify you of any changes to the state variables of your contract. In Remix they automatically show up in the user interface on the right side. When you want to see them from the front end of your dApp you will have to watch for the Events are shown in the right side user interface of remix

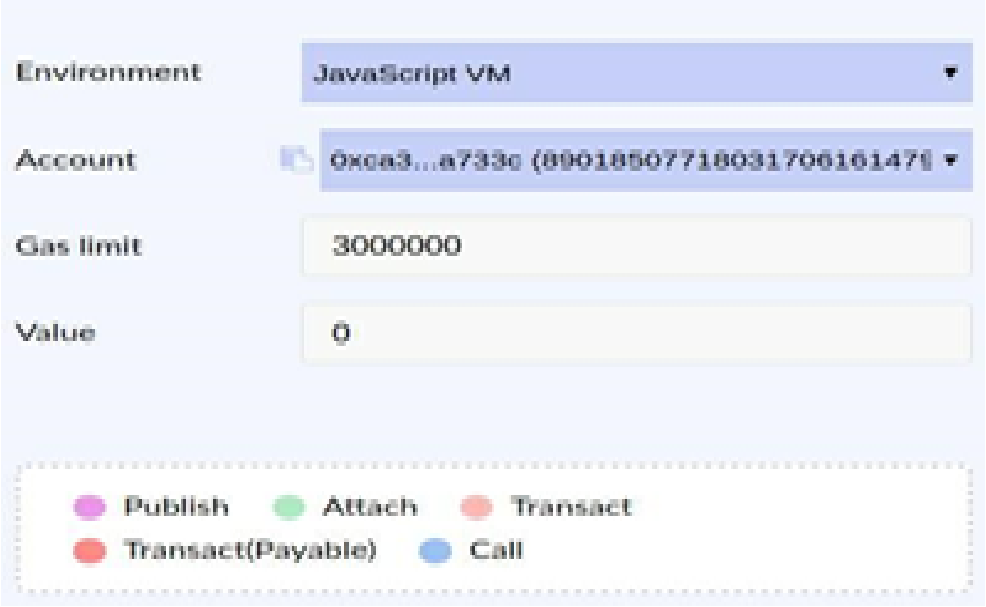

Figure 3.6: Remix (a)

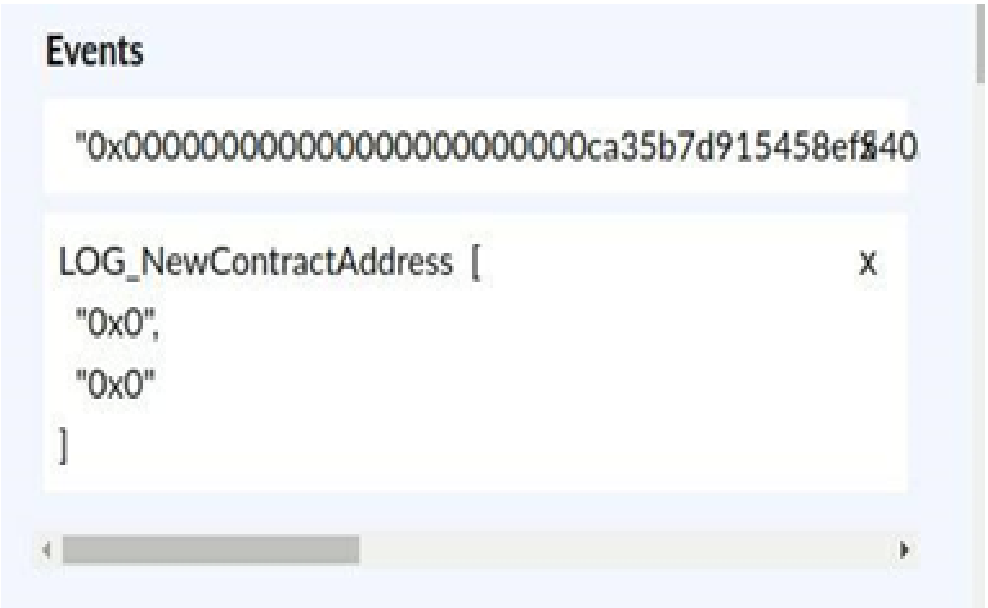

Figure 3.7: Remix (b)

- Note that Remix is glitchy sometimes. If it isn't doing what you think it should be doing, open and close a new chrome window and it often helps. This is new software that is in active development, so bugs are

expected.

- Note that sometimes the numbers in local or state storage while debugging will be extremely large. This might be due to accidental overflow for an integer calculation. However I have noticed that there is a bug in Remix . Sometimes while debugging it will show overflow, but in reality the contract still works and you get the correct results when executing functions in your dApp. So really pay attention when debugging to local and state storage, and if it is showing overflow and your contract still works perfectly, you are probably okay.

Using the Solidity Locals and Solidity State variable viewer in the debugger of remix is extremely helpful to see if your functions are working properly.Keep in mind the first dApp you build will be slow when you are calling functions because you have to wait for blocks to be mined to get a response. This won't rear its ugly head while using the JVM, as everything happens instantaneously. As you move away from the JVM it will be more frustrating to debug, so do as much debugging as you can here.

3.4.4 Docker

Docker is a software which provides centralized platform to execute your application. It wraps software components into a complete standardized unit which contains everything require to run. Whether it is code, runtime environment, tools or libraries. It guarantees that the software will always run as expected.

Figure 3.8: Docker

Docker provides the facility to run an application in a isolated environment which is called container. You can run many containers simultaneously on a given host. It is lightweight, so starts instantly and uses less RAM. It is secure by default because each container is isolated from one another.

Docker Engine

It is a client server application that contains the following major components.

- A server which is a type of long-running program called a daemon process.
- The REST API is used to specify interfaces that programs can use to talk to the daemon and instruct it what to do.
- A command line interface client.

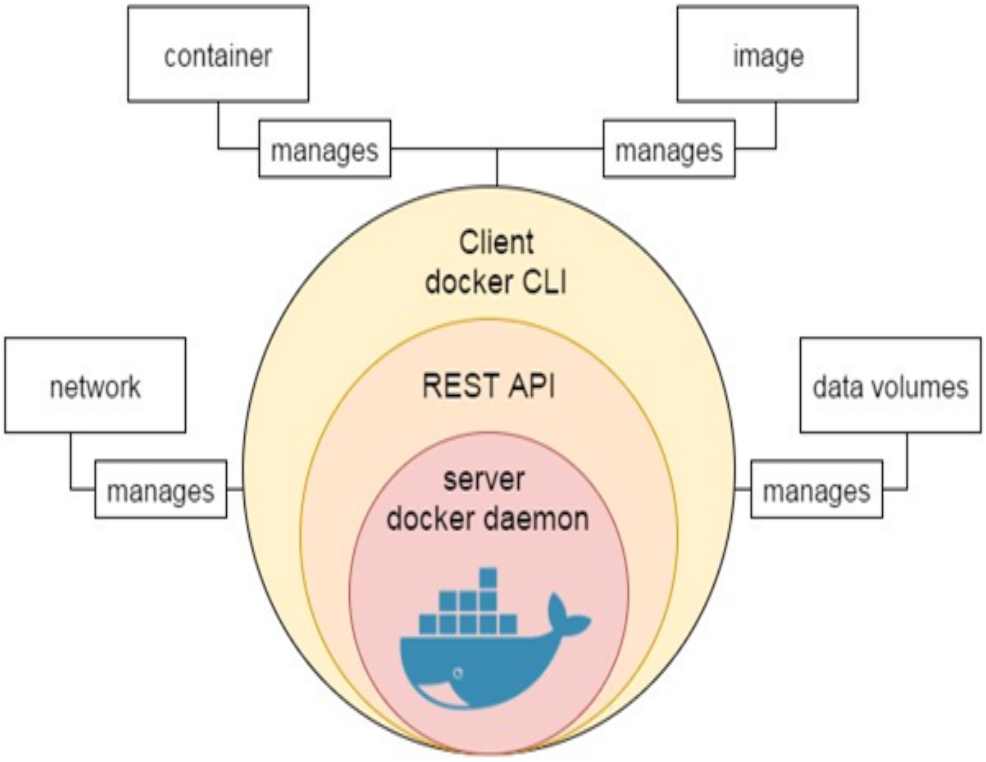

Figure 3.9: Engine-component-flow

Docker Features

Although Docker provides lots of features, we are listing some major features which are given below.

- Easy and Faster Configuration
- Increase productivity • Application Isolation
- Swarm
- Routing Mesh
- Services

3.4.5 TestRPC

TestRPC is a Node.js based Ethereum client for testing and development. It uses ethereumjs to simulate full client behavior and make developing Ethereum applications much faster. It also includes all popular RPC functions and features (like events) and can be run deterministically to make development a breeze. Ethereum TestRPC is a fast and flexible way to emulate calls to the blockchain without the overheads of running an actual Ethereum node. Features include:

- Accounts can be re-used and reset.
- The live blockchain can be forked at a given block to test existing in-the-wild contracts.
- Specific accounts can be instantiated with a fixed amount of Ether (no need for faucets or mining).
- Gas price and mining speed can be modifed.

3.4.6 Truffle

Truffle is a framework for rapid development, testing and deployment of smart contracts with Solidity and Javascript. Solidity is a JS-like high level contract language that compiles down to bytecode used in the EVM (Ethereum Virtual Machine). It compiles your blockchain contracts, injects them into your web app.

3.4.7 MongoDB

MongoDB is a cross-platform,open-source, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.[3]

Figure 3.10: MongoDB

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data. The following table shows the relationship of RDBMS terminology with MongoDB.

Table 3.3: RDBMS vs MongoDB

RDBMS	**MongoDB**
Database	Database
Table	Collection
column	Field
Tuple/Row	Document
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)

Sample Document

```
{
        _id: ObjectId(7df78ad8902c)
        title: 'MongoDB Overview',
        description: 'MongoDB is no sql database',
        by: 'tutorials point',
        url: 'http://www.tutorialspoint.com',
        tags: ['mongodb', 'database', 'NoSQL'],
                likes: 100,
                comments: [

                        {

                        user:'user1',
                message: 'My first comment',
                dateCreated: new Date(2011,1,20,2,15),
                                like: 0
                        },
                        {
                        user:'user2',
                message: 'My second comments',
                dateCreated: new Date(2011,1,25,7,45),
                                        like: 5
                        }
                                ]
```

Advantages of MongoDB over RDBMS

- Schema less - MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
- Structure of a single object is clear.
- No complex joins.
- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
- Tuning.
- Ease of scale-out -MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) working set, enabling faster access of data.

3.4.8 Linux

Linux is the best-known and most-used open source operating system. As an operating system, Linux is software that sits underneath all of the other software on a computer, receiving requests from those programs and relaying these requests to the computer's hardware. Linux is different from other operating systems in many important ways. First, and perhaps most importantly, Linux is open source software. The code used to create Linux is free and available to the public to view, edit, and-for users with the appropriate skills-to contribute to. Companies and individuals choose Linux for their servers because it is secure, and you can receive excellent support from a large community of users. One main advantage of open-source technologies such as Linux is the wide range of options available to users and the increased security. With Linux being open-source, several distributions are available to the end-user. Debian, Fedora, Ubuntu and Mint are just a few of the distributions available to end users, and these distributions are completely free to download. Security is the other main advantage. Several whitehat hackers have contributed to the overall security of Linux, and by making the source available to anyone, security experts can help identify any main security flaws in the operating system. The advantage over operating systems

such as Windows is that security flaws are caught before they become an issue for the public.

Remix

Remix provides an integrated development environment (IDE) for smart contract development. It focuses on the development and deployment of Solidity written smart contracts.
Remix is a good solution if you intend to:

- Develop smart contracts (remix integrates a solidity editor).
- Debug a smart contract's execution.
- Access the state and properties of already deployed smart contract.
- Debug already committed transaction.
- Analyze solidity code to reduce coding mistakes and to enforce best practices.
- Together with Mist (or any tool which inject web3), Remix can be used to test and debug a Dapp (see Debugging a Dapp using Remix - Mist - Geth)

Environment

- **JavaScript VM**: All the transactions will be executed in a sandbox blockchain in the browser. This means nothing will be persisted and a page reload will restart a new blockchain from scratch, the old one will not be saved.
- **Injected Provider**: Remix will connect to an injected web3 provider. Mist and Metamaskare example of providers that inject web3, thus can be used with this option.
- **Web3 Provider**: Remix will connect to a remote node. You will need to provide the URL address to the selected provider: geth, parity or any Ethereum client.
- **Account**: the list of accounts associated with the current environment (and their associated balances).
- **Gas Limit**: the maximum amount of gas that can be set for all the transactions created in Remix.

- **Value**: the amount of value for the next created transaction (this value is always reset to 0 after each transaction execution).

Figure 3.11: Environment:Remix(a)

Initiate Instance

This section contains the list of compiled contracts and 2 actions:

- **At Address** assumes the given address is an instance of the selected contract. It is then possible to interact with an already deployed contract. There's no check at this point, so be careful when using this feature, and be sure you trust the contract at that address.
- **Create** send a transaction that deploys the selected contract. When the transaction is mined, the newly created instance will be added (this might take several seconds). Note that if the **constructor** has parameters, you need to specify them.

Pending Instances

Validating a transaction take several seconds. During this time, the GUI shows it in a pending mode. When transaction is mined the number of pending transactions is updated and the transaction is added to the log (see

Terminal)
Instance List - This section contains a list of instances to interact with. Several cases apply:

- The called function is declared as in Solidity. The action has a blue background, clicking it does not create a new transaction. Clicking it is not necessary because there are not state changes - but it will update the return value of the function
- The called function has no special keywords. The action has a light red background, clicking on does create a new transaction. But this transaction cannot accept any amount of Ether
- The called function is declared as payable in Solidity. The action has a red background, clicking it does create a new transaction and this transaction can accept value

For more information about Solidity modifier, see Solidity modifier . If a function requires input parameters, it is required to specify them.

using ABI

Using Create or At Address is a classic use case of Remix. It is possible though to interact with a contract by using its ABI. The ABI is a JSON array which describe its interface.
To interact with a contract using the ABI, create a new file in Remix with extension abi and copy the ABI content to it. Then in the input next to At Address, put the Address of the contract you want to interact with. Click on At Address, a new "connection" with the contract will popup below.

using the recorder

The Recorder allows to save a bunch of transactions in a JSON file and rerun them later either in the same environment or in another.
Saving to JSON allows to easily check the transaction list, tweak input parameters, change linked library, etc...
We can find many use cases for the recorder, for instance:

- After having coded and tested contracts in a constrained environment (like the JavaScript VM), it could be interesting to redeploy them easily in a more persisted environment (like a Geth node) in order to check whether everything behaves normally in a classic environment

- Deploying contract does often require more than creating one transaction.
- Working in a dev environment does often require to setup the state in a first place

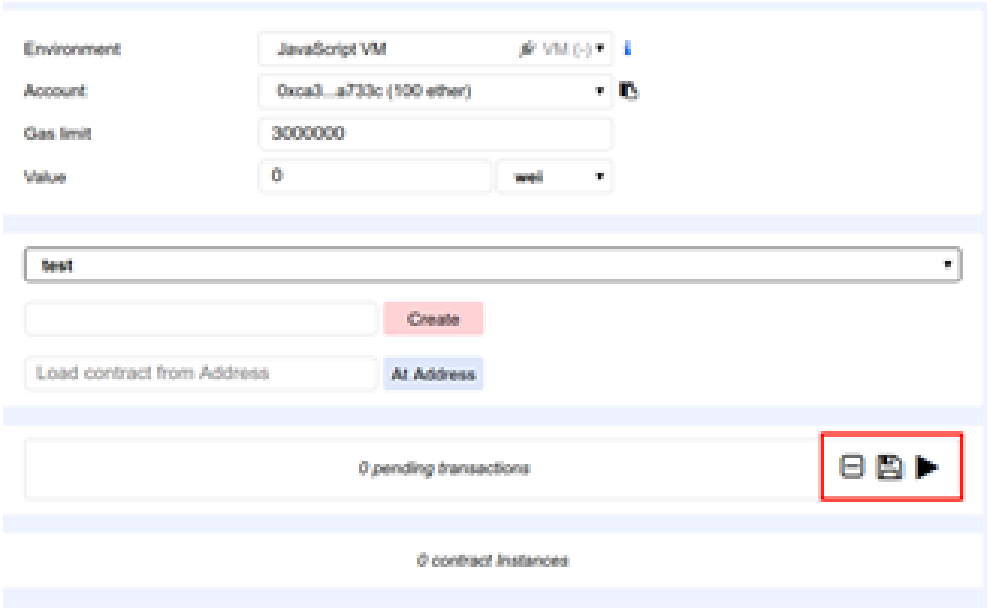

Figure 3.12: Environment:Remix(b)

Saving a record ends up with the creation of this type of content (see below):In that specific record, 3 transactions are executed:The first corresponds to the deployment of the lib.
The second corresponds to the deployment of the contract te, the first parameter of the constructor is set to 11.

3.4.9 Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as web sites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Visual Studio includes a code editorsupporting IntelliSense (the code completioncomponent) as well as code refactoring. The integrated debugger

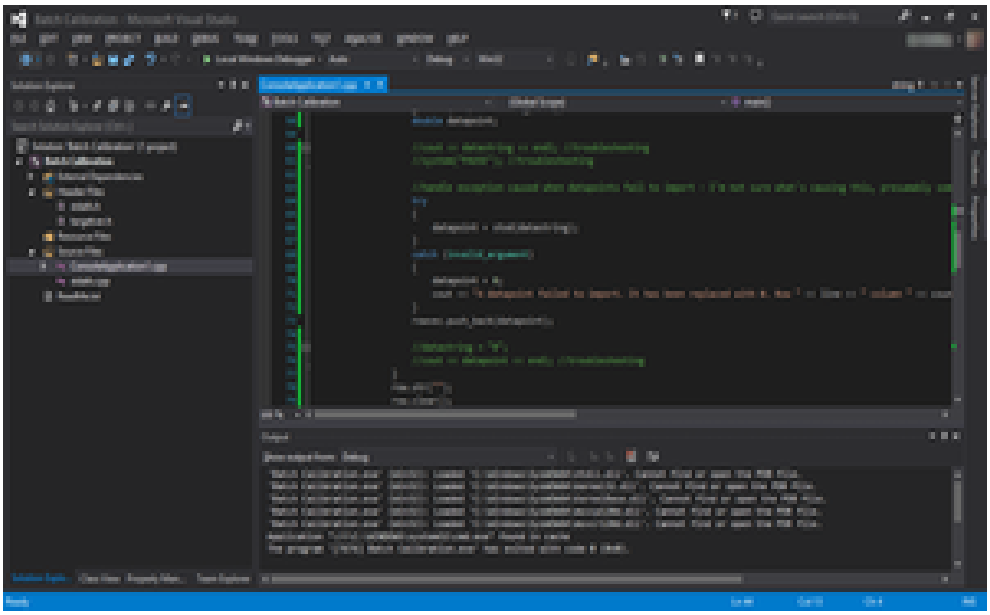

Figure 3.13: Visual Studio Code

works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source controlsystems (like Subversion) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Serverclient: Team Explorer).
Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++, C++/CLI, Visual Basic .NET, C#, F#,JavaScript, TypeScript, XML, XSLT, HTML and CSS. Support for other languages such as Python,Ruby, Node.js, and M among others is available via plug-ins. Java (and J#) were supported in the past.

Chapter 4

SYSTEM TESTING

System testing of software is testing conducted on complete, integrated system to evaluate the system compliance with its specified requirements. System testing falls within the scope of black box testing, and such, should require no knowledge of the inner design of the code or logic. As its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system. The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is more limiting type of testing; it seeks to defects both within the "inter-assemblages" and also within the system as whole.
System testing is actually done to the entire system against the functional Requirement Specification (FRS) or the System Requirement Specification (SRS).Moreover, the system testing is an investigatory testing phase, where the focus is to have almost a destructive attitude and test not only the design, but also the behaviour and even the believed expectations of the customer. It is also intended to test up to the and beyond the bounds defined in the software /hardware requirements specification. One could view system testing as the final destructive testing phase before user acceptance testing.

4.1 Unit testing

It is done to check the individual part of the system; each module should be separately tested for errors. So we design test for each module, here we have to consider the following points such as, boundary conditions, independent Path ,error handling method , exceptional handling.
In the organizations we are required to test our functions before we commit

the updating the build. After releasing a build the testing team also test the functionalities for errors.

4.1.1 Testing in Proposal Submission Module

The unit testing is performed to determine if there are any issues with the webpage. Here we have two types, the white box testing and the black box testing for each of the module. The unit testing which is done in the provider module will helps us to test the each and every details in this module. It will test that if the patient and insurance details are added correctly. This module mainly deals with entering the payment details of the patient to blockchain. The admin of clinic/hospitals will add the payment details to the system. Since the blockchain is immutable, wrong data once entered will be permanently recorded.

4.1.2 Testing in Proposal Validation Module

The Proposal Validation Module plays an important role in this system hence it is important to test this module seriously. Here the unit testing is performed for each and every details entered to the blockchain by the clinic and the also the details about the validation which is added by the regulatory body are also stored in to the system.

Test Cases

Test Case 1: To check if the details of a payment once entered are repeated or not.
Result: A patient Id once entered for a date is saved only once.

Test Case 2: Check all the details added to the block chain is accurate and it cannot be alter by any outside parties.
Result:The details once entered cant be altered.

Test Case 3: Check all the nodes of the Ethereum network are properly connected and the block chain is implemented.

4.2 Integration Testing

Integration testing is the phase in software testing which individual software modules are combined and tested as a group. Here we have mainly two modules included in the system. The Proposal Submission module and

the Proposal Validation Module(PVM) are the two modules which plays an important roles in the system. The integration testing is performed by integrating the two modules. Here the Submission and Validation modules are combined and tested to check whether both the functionalities are working properly. The entire details of the provider entered by the clinic/Hospital are validated and added to blockchain in the Validation module. The integration testing is very important to the system like a unit testing.

4.2.1 User Acceptance Testing

User acceptance testing is a test conducted to determine if the requirements of a specification or contract are met. It may involve black-box testing performed on a system prior to its delivery. This testing is done by the end user of the system to check and verify the all the functionalities of the system is properly working as per the requirements of the user. The end user also checks the system to determine how the block chain technology is implemented in the system and how the Ethereum network is set up with all the nodes. The user can test each and every functionalities of the system from the starting one by one to ensure the accuracy and correctness of the system. The user acceptance testing is very important to a system.

4.2.2 System Testing

When a system is developed it is hoped that it perform properly. System testing is the process of checking if the developed system is working according to the original objectives and requirements. In this I have done system testing for ensuring whether the software is friendly to the user. Also we have checked that when we are giving the correct input whether the output is correct or not. Also we have tested how the system will react to the incorrect inputs. In our organization there is a separate testing team available for testing. Before release of client product they will lock the work and test for any error and report it.

Chapter 5

SYSTEM IMPLEMENTATION

System implementation covers a broad spectrum of activities from a detailed workflow analysis to the formal go-live of the new system. During system implementation organizations may refine the initial workflow analysis that had been completed as part of the requirements analysis phase. With the aid of the vendor they may also start mapping out the proposed new workflow. The system implementation phase requires the vendor to play a very prominent role. In addition to the workflow analysis it is during this phase that full system testing is completed. Other key activities that would occur during this phase include piloting of the new system, formal go-live and the immediate post implementation period during which any application issues are resolved. This stage will help ensure that potential implementation "pitfalls" are avoided and that security is maximized from the outset. We typically review the implementation plan and recommend how to incorporate the necessary controls during implementation. System implementation is a critical stage of the project, when the technical components of a solution must be unified and integrated. The pace of today's business environment means that organizations no longer have the luxury of extensive development cycles.
This project need to install on server that holds a large number of users and concurrent requests. Also high speed server is needed to run this application.

5.1 Implementation plan

The proposed system is planned to be implemented using the direct cutover method. Direct cut-over means that an old system is replaced by a new system at the same time. Once the system is tested and ready, the whole

organization is transferred on one day to the new system. This is less expensive than running parallel systems and makes the transfer process less time-consuming. It is not very time consuming as once the old system has stopped being used the new system is immediately being set up. If the system has not been implemented properly the new system may fail to work and this will affect the whole organization.

Chapter 6

CONCLUSION

The Speedchain EHR project is implemented using Blockchain technology. The project is built in Ethereum platform. This ensures security and transparency for the ecosystem. The payer of the system will be the insurance sector and the providers will be Doctors. With the implementation of the system fraud activities and other issues can be minimized.
It is mainly focused on availability of doctors details in multiple places at same time.The Blockchain Technology provide distributed ledger technology for distributed storage.The providers can make registration by submitting their medical records,personal information,practice information,mailing information,miscellaneous information and electronic information etc.The primary informations are stored in the local database (MongoDB) and other information are stored into the block chain and available for everyone who are connected with the same blockchain port.

Chapter 7

REFERENCES

[1] Blockchain White Paper. `https://www.blockchain.com/whitepaper/index.html`.

[2] HIPAA (Health Insurance Portability and Accountability Act). `https://searchhealthit.techtarget.com/definition/HIPAA`.

[3] MongoDB Tutorials. `https://docs.mongodb.com/manual/tutorial/`.

[4] Brian Dobing and Jeffrey Parsons. How uml is used. *Communications of the ACM*, 49(5):109–113, 2006.

[5] Peter Membrey, Eelco Plugge, and DUPTim Hawkins. *The definitive guide to MongoDB: the noSQL database for cloud and desktop computing*. Apress, 2011.

[6] Marc Pilkington. 11 blockchain technology: principles and applications. *Research handbook on digital transformations*, page 225, 2016.

[7] Aaron Wright and Primavera De Filippi. Decentralized blockchain technology and the rise of lex cryptographia. 2015.

Chapter 8

APPENDICES

8.1 Screen Shots

Enroll Provider Details

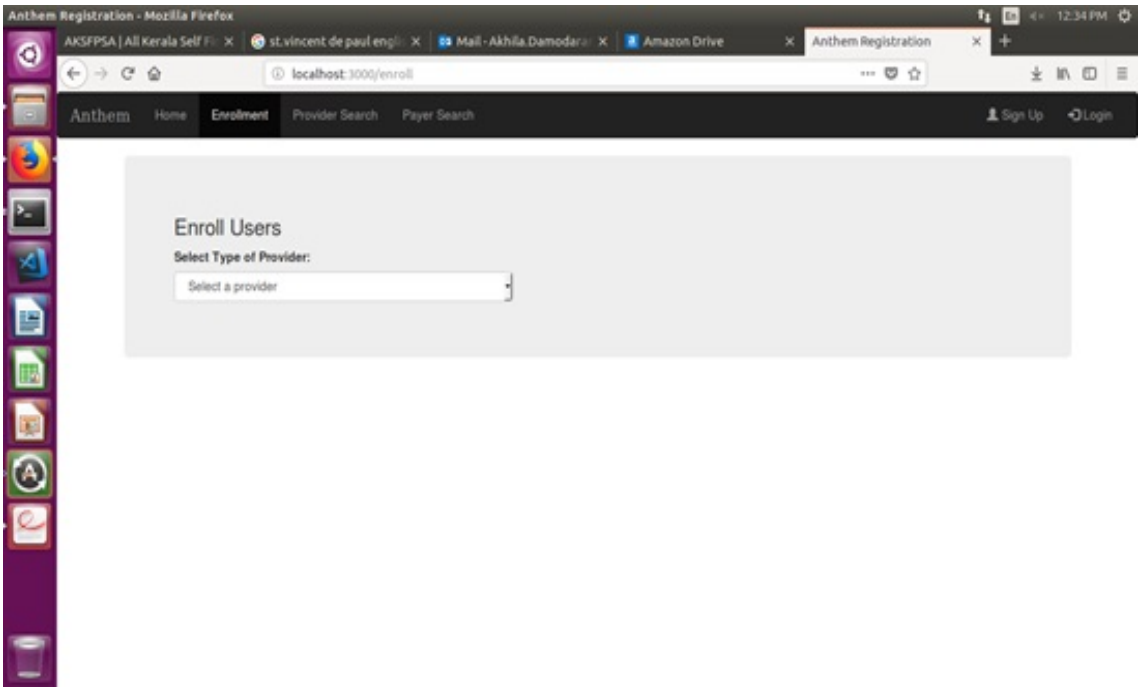

Figure 8.1: Enroll Provider Details

Personal information

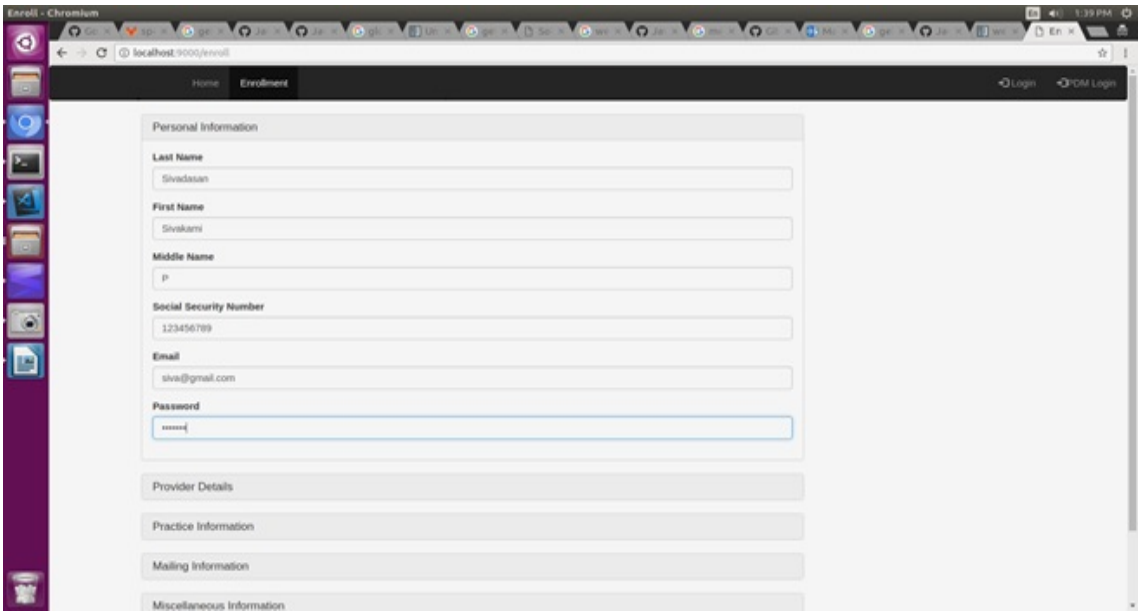

Figure 8.2: Personal information

Provider details

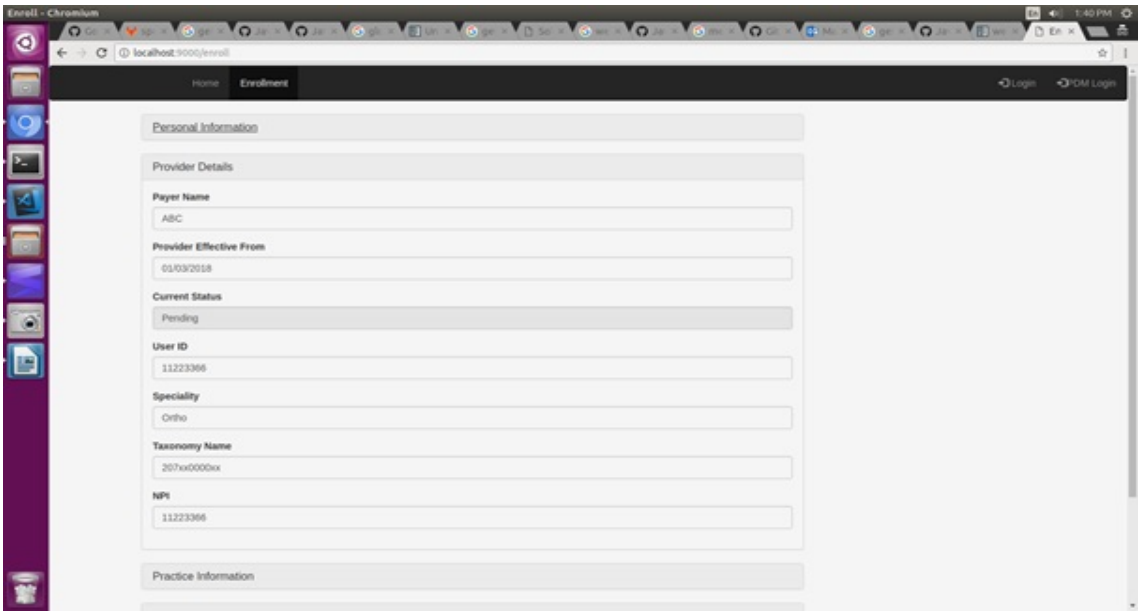

Figure 8.3: Provider details

practice information

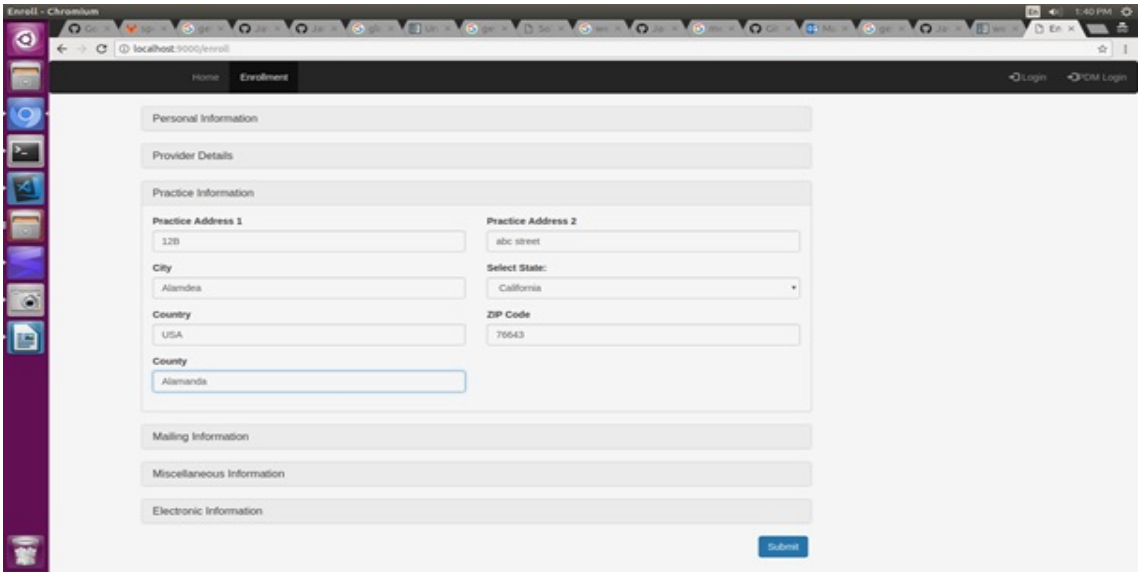

Figure 8.4: practice information

Mailing information

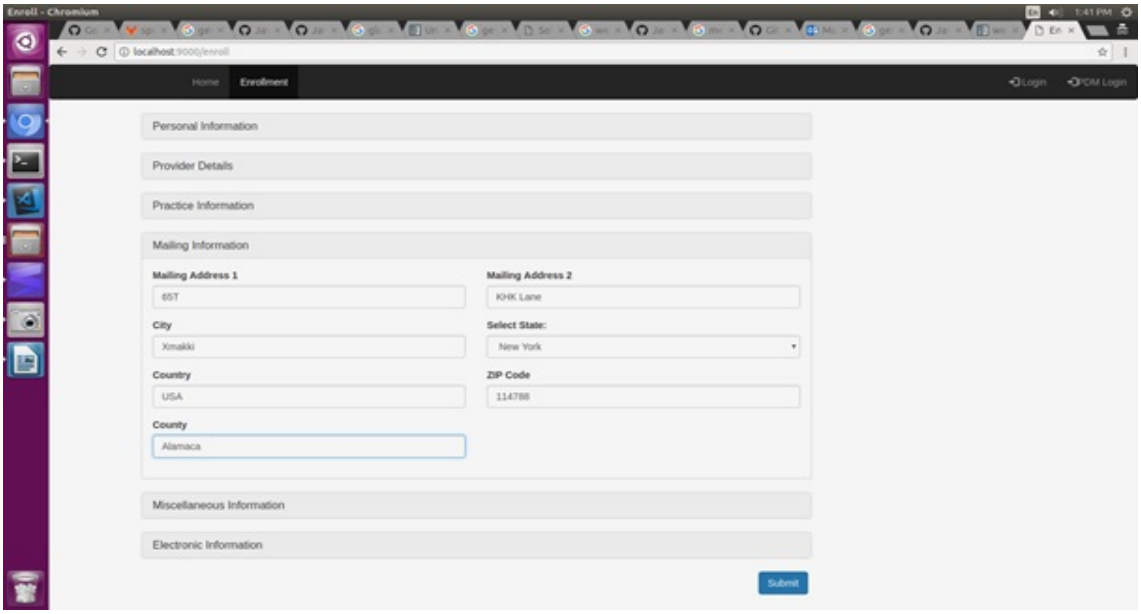

Figure 8.5: Mailing information

Miscellaneous Information

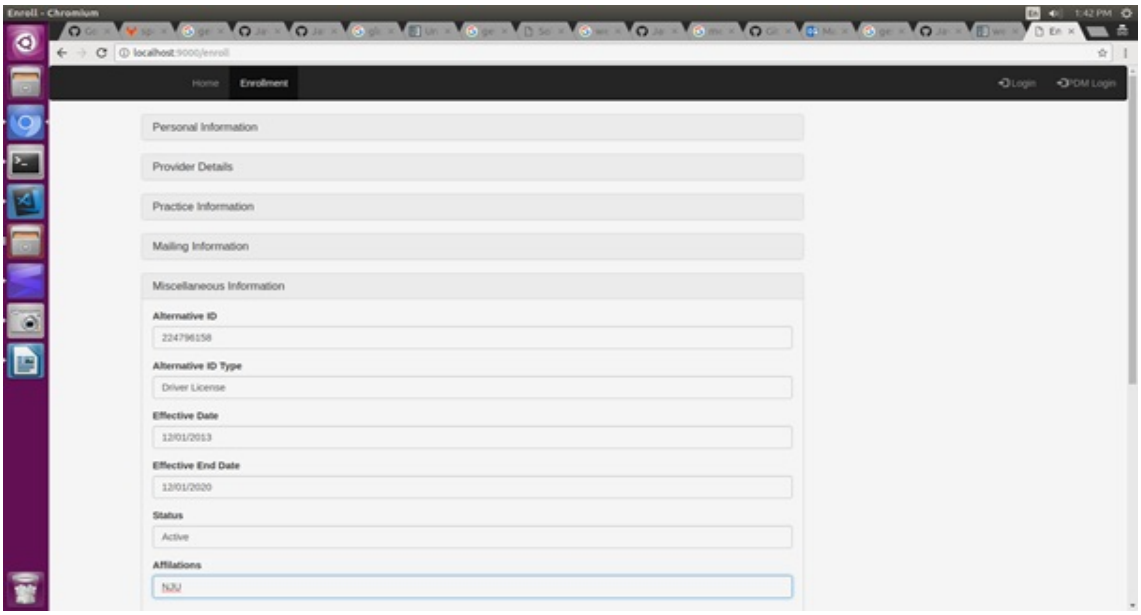

Figure 8.6: Miscellaneous Information

Account Details

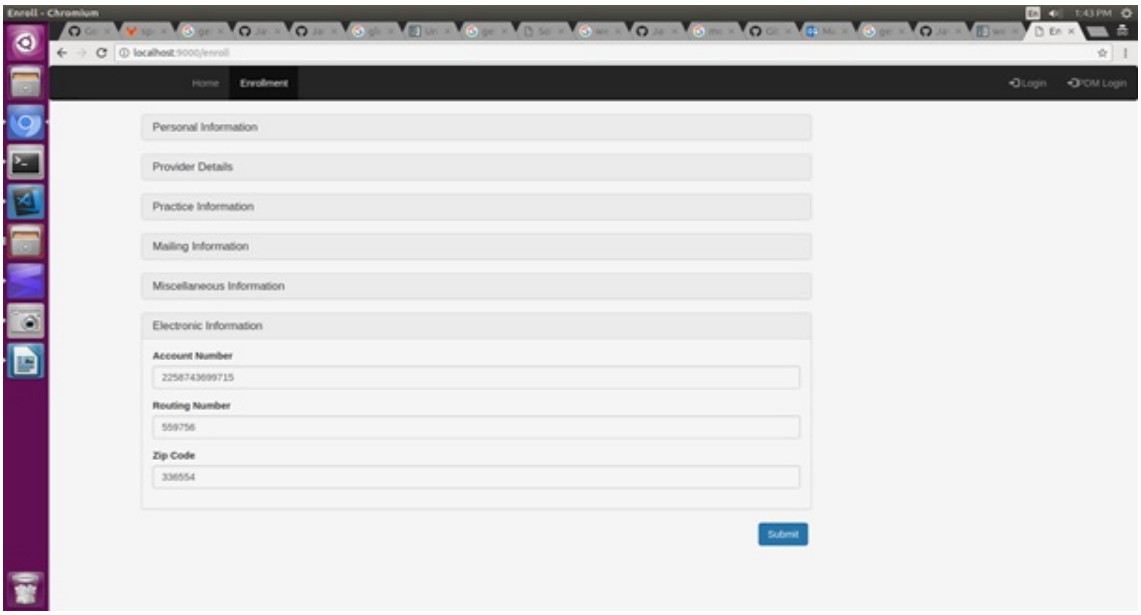

Figure 8.7: Account Details

Submission of Details

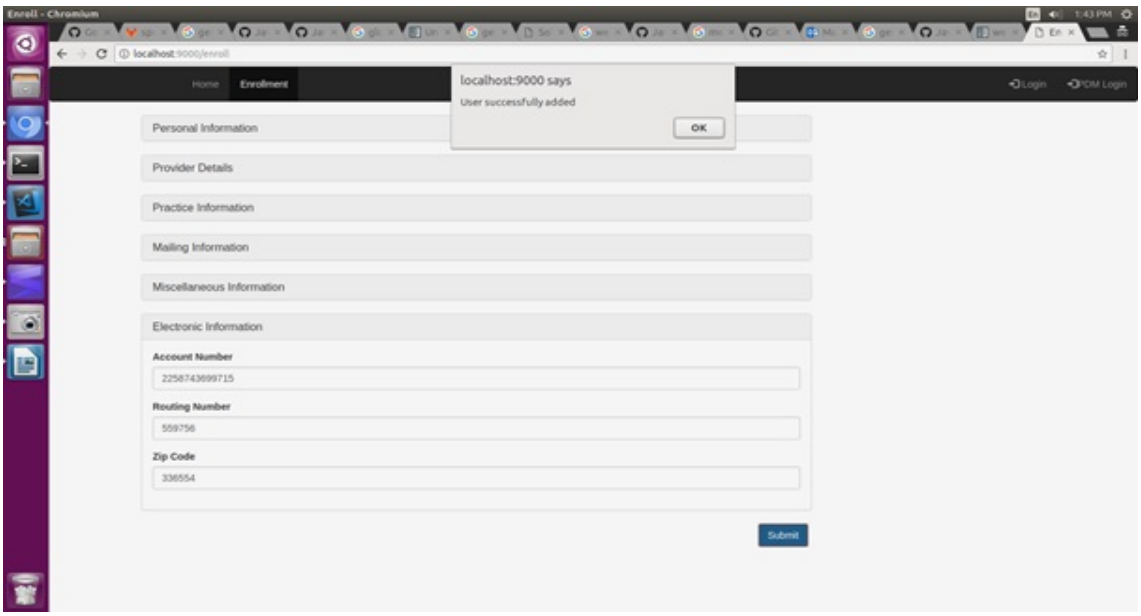

Figure 8.8: Submission of Details

Provider Login

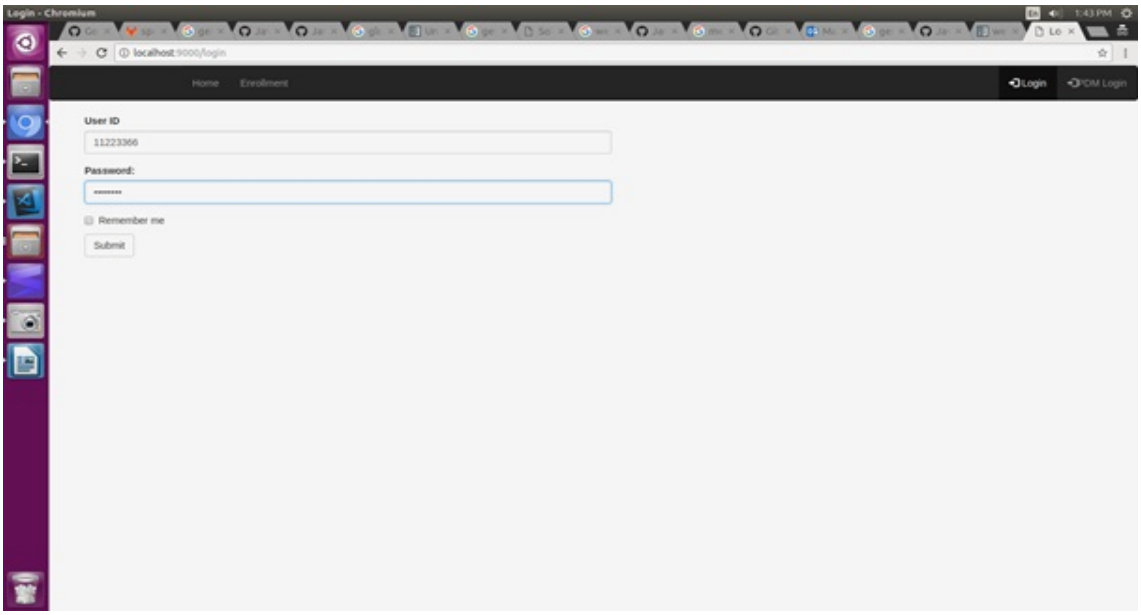

Figure 8.9: Provider Login

Profile of provider

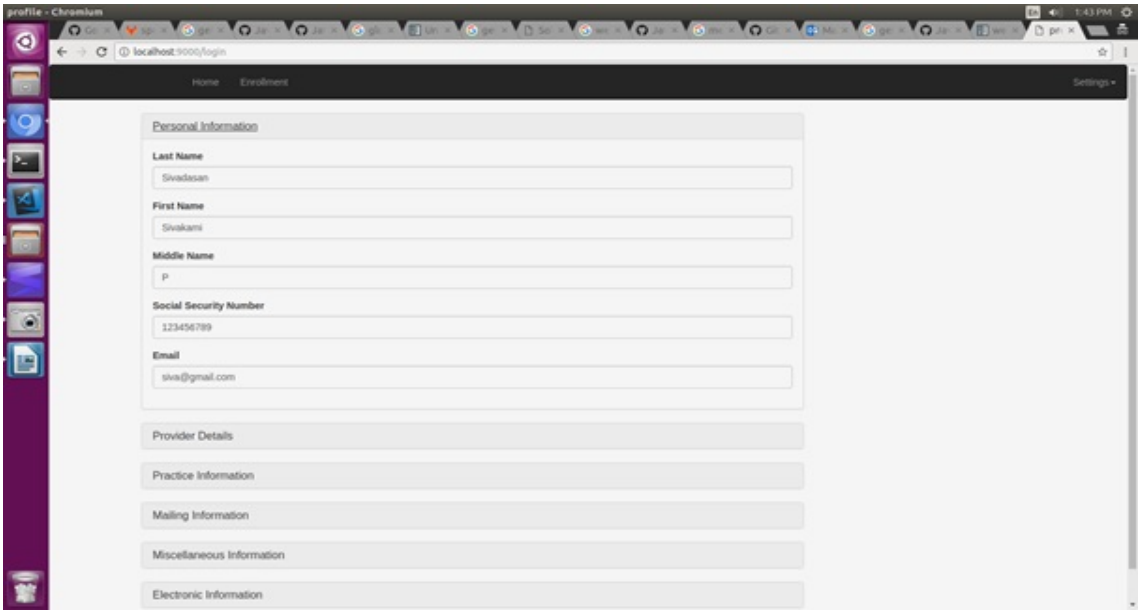

Figure 8.10: Profile of provider(a)

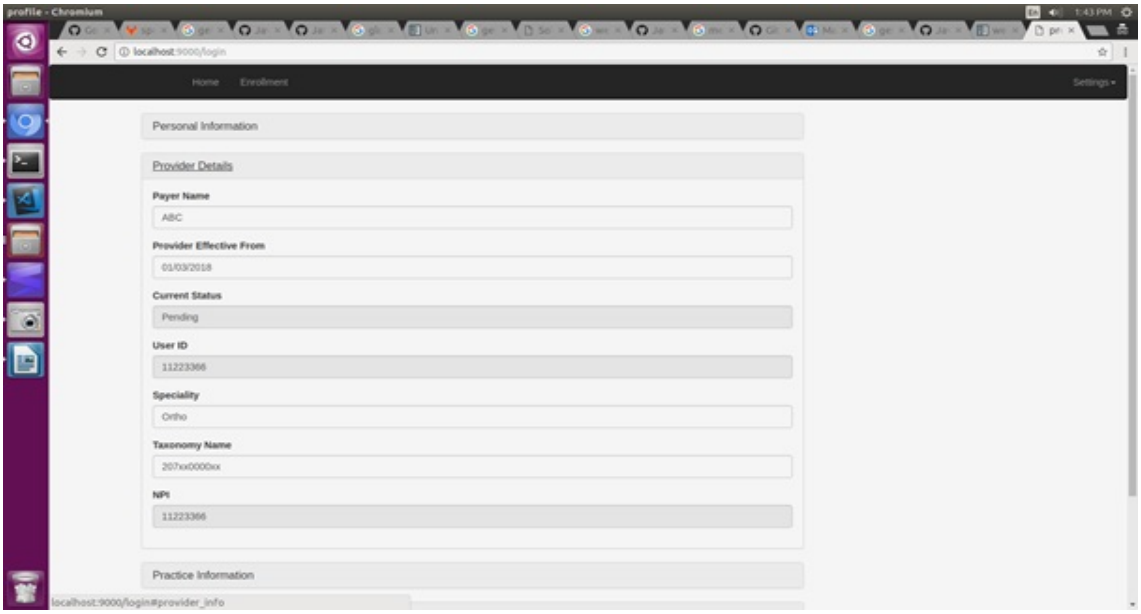

Figure 8.11: Profile of provider(b)

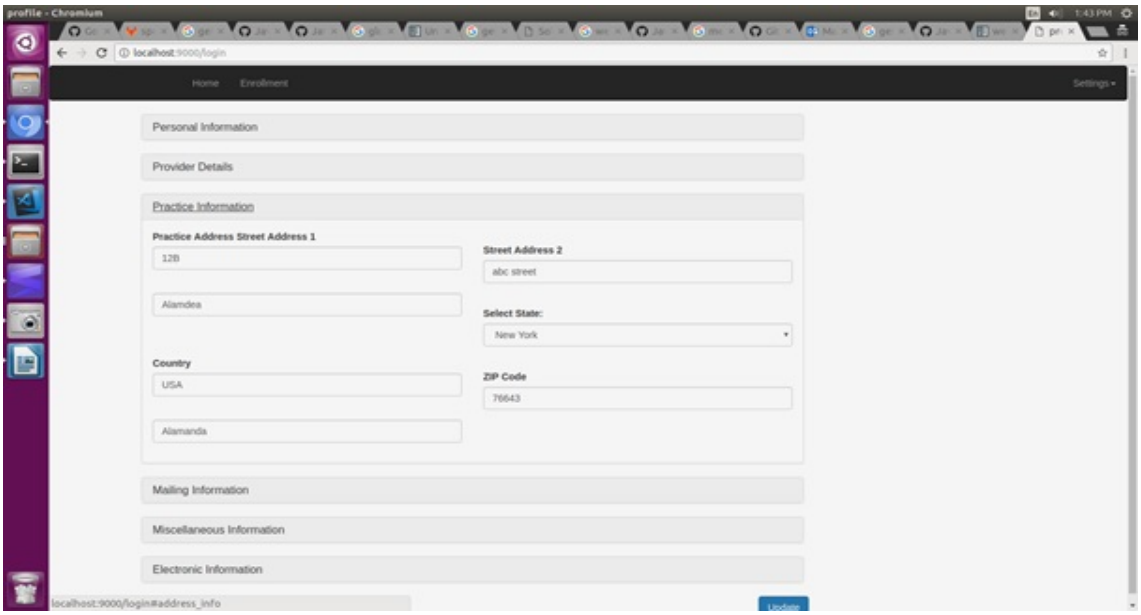

Figure 8.12: Profile of provider(c)

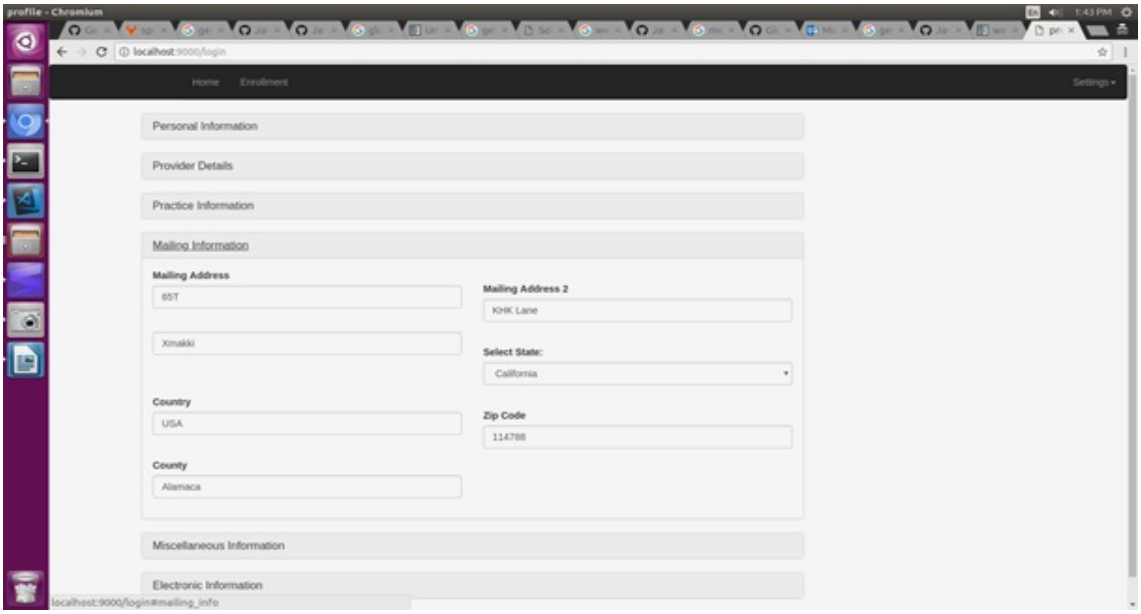

Figure 8.13: Profile of provider(d)

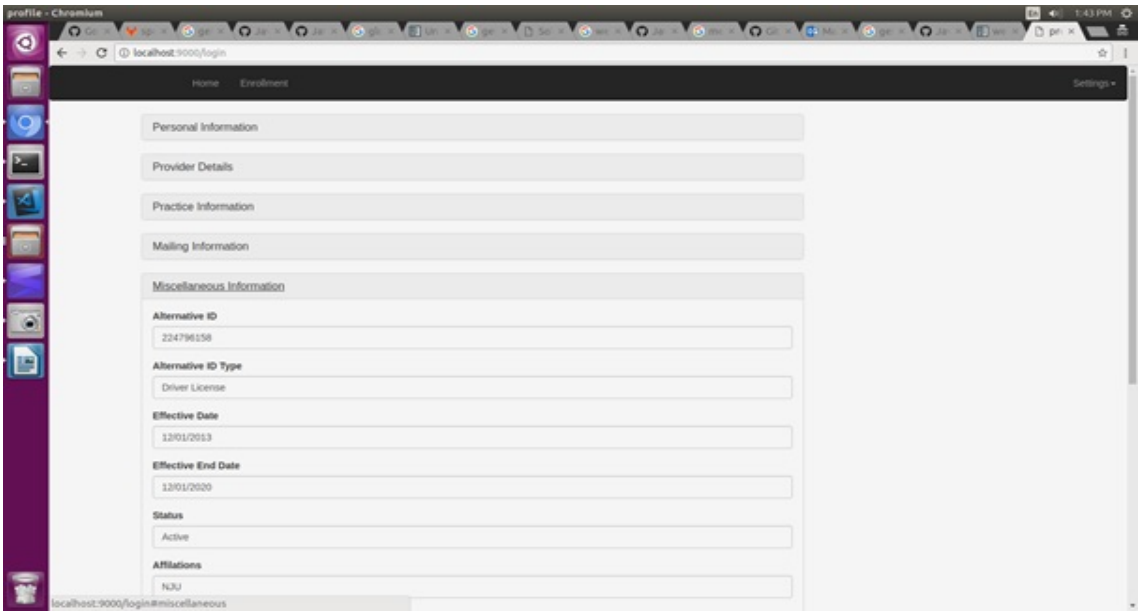

Figure 8.14: Profile of provider(e)

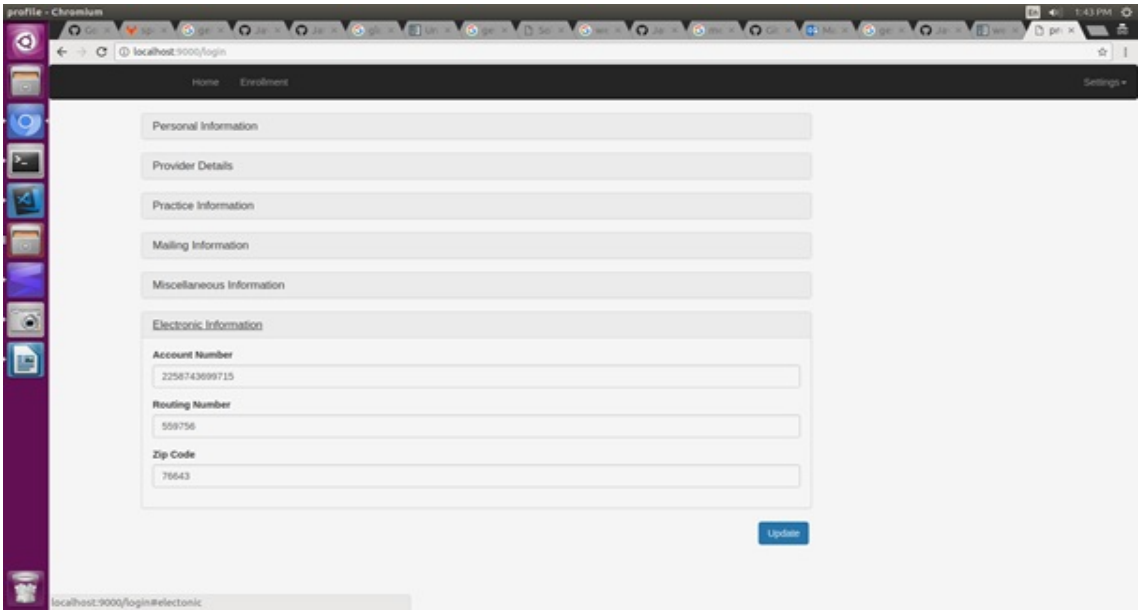

Figure 8.15: Profile of provider(f)

Update and logout

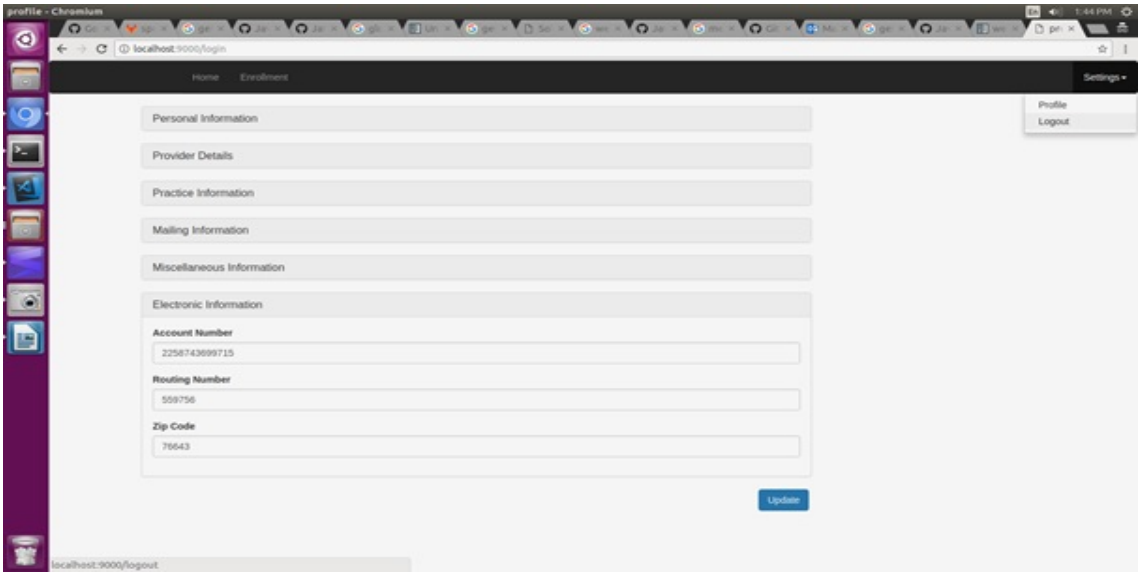

Figure 8.16: Update and logout

PDM Login

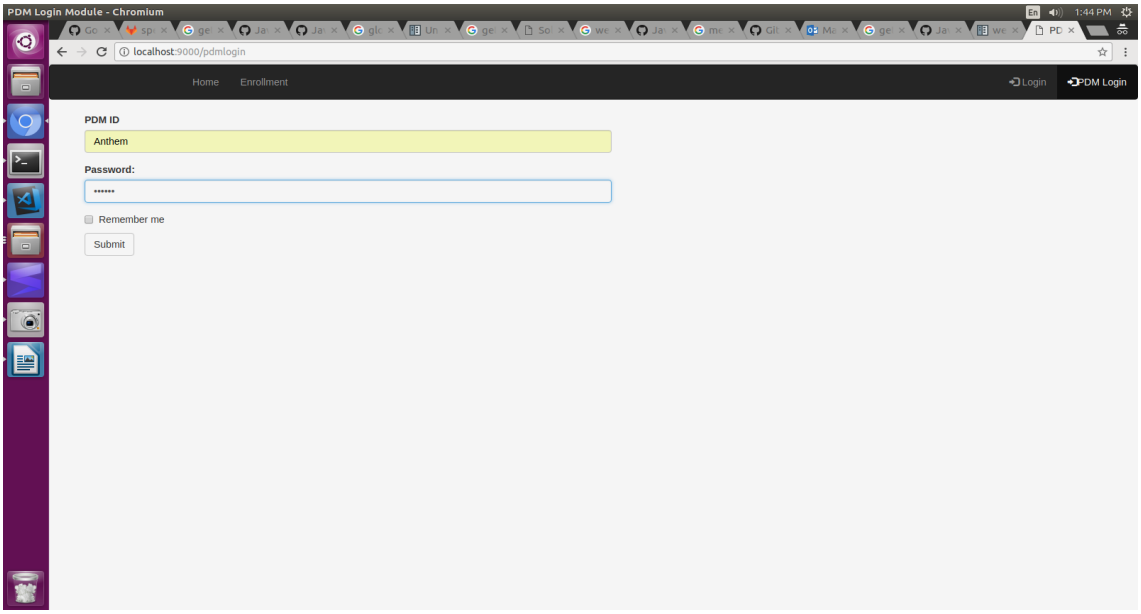

Figure 8.17: PDM Login

Validation(Approve)

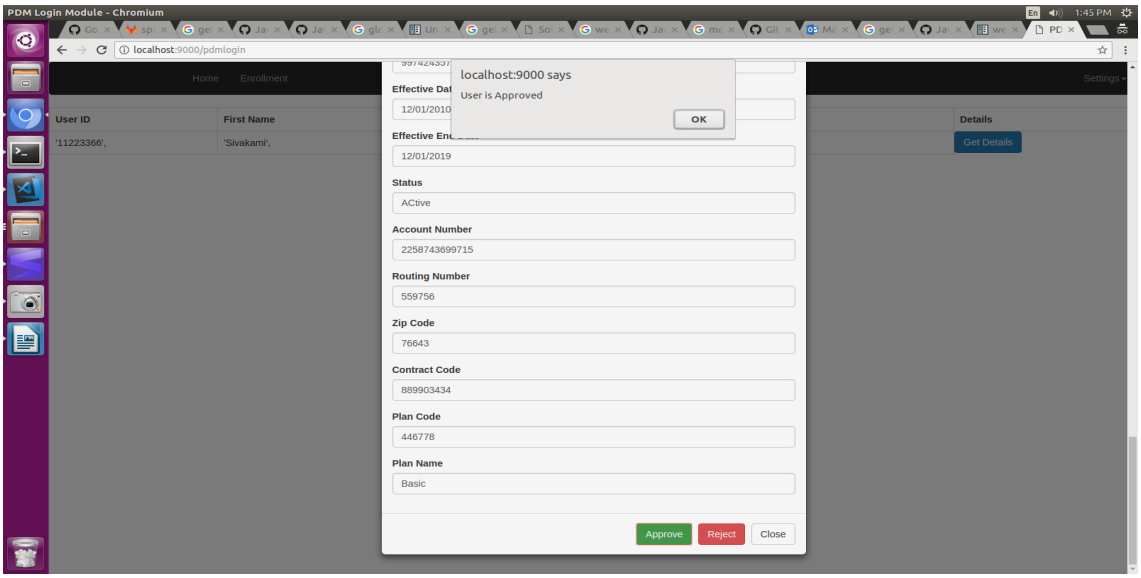

Figure 8.18: Validation(a)

Validation(Reject)

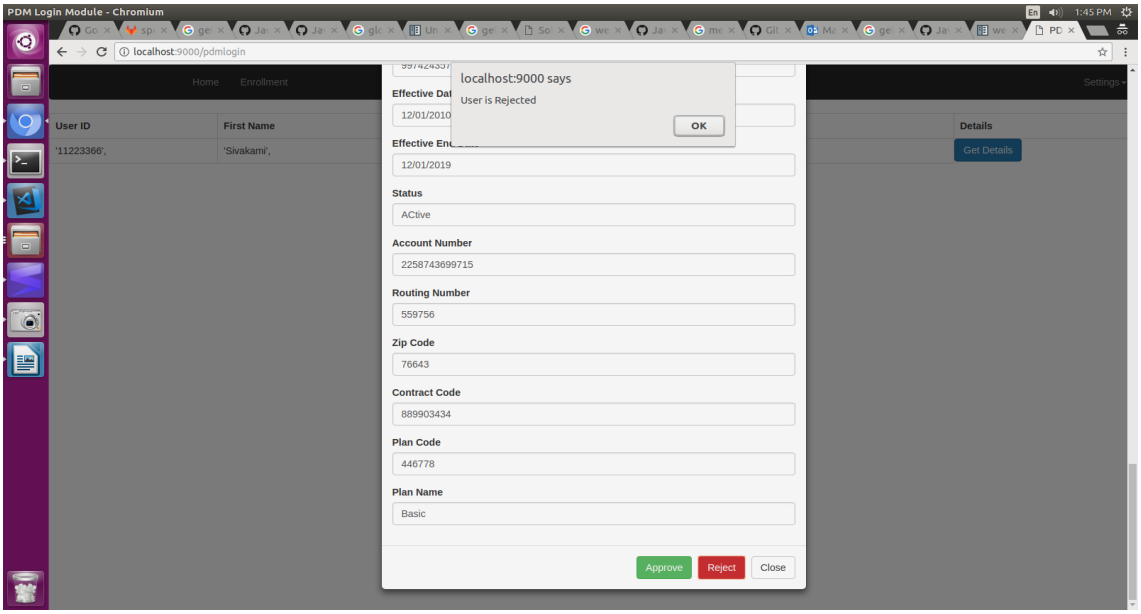

Figure 8.19: Validation(b)

Search for Doctor

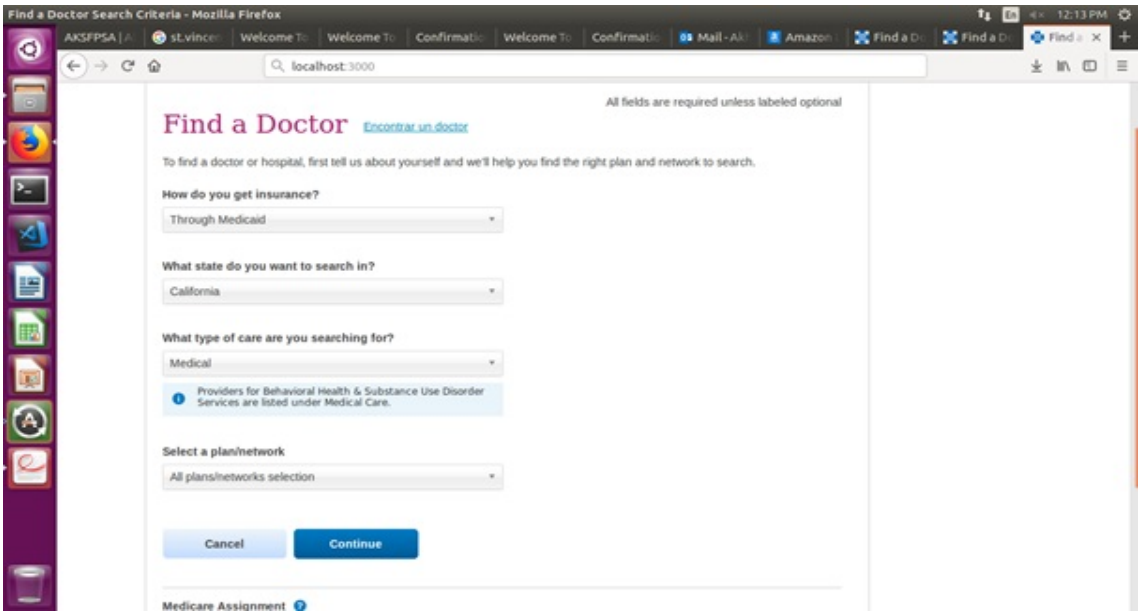

Figure 8.20: Search for Doctor