

## 1. INTRODUCTION

All these ideas and solutions above propose to protect data security, by designing a new service paradigm supporting the decoupling of data and application, or by designing a specific blockchain to meet demands of certain applications, or by integrating as a functional component to analyze data security. However, none of them treats the problem of data security from the view of architecture. To fill this gap, SecNet tries to construct a common and general networking architecture by the power of blockchain at a large scale, which can support dynamic update of all these functional component separately at any time as needed, to efficiently and effectively improve the data security for all applications. With the development of information technologies, the trend of integrating cyber, physical and social (CPS) systems to a highly unified information society, rather than just a digital Internet, is becoming increasingly obvious. In such an information society, data is the asset of its owner, and its usage should be under the full control of its owner, although this is not the common case.

## **2. LITERATURE SURVEY**

### **1. HYPERCONNECTED NETWORK: A DECENTRALIZED TRUSTED COMPUTING AND NETWORK PARADIGM**

With the development of the Internet of Things, a complex CPS system has emerged and becoming a promising information infrastructure. In the CPS system, the loss of control over user data has become a very serious challenge, making it difficult to protect privacy, boost innovation, and guarantee data sovereignty. In this article, we propose Hypernet, a novel decentralized trusted computing and networking paradigm, to meet the challenge of loss of control over data. Hypernet is composed of the intelligent PDC, which is considered as the digital clone of a human individual; the decentralized trusted connection between any entities based on blockchain as well as smart contract; and the UDI platform, enabling secure digital object management and an identifier-driven routing mechanism. Hypernet has the capability of protecting data sovereignty, and has the potential to transform the current communication-based information system to the future data-oriented information society.

### **2. LIGHT EIGHT RFID PROTOCOL FOR MEDICAL PRIVACY PROTECTION IN IOT**

Traditional medical privacy data are at a serious risk of disclosure, and many related cases have occurred over the years. For example, personal medical privacy data can be easily leaked to insurance companies, which not only compromises the privacy of individuals, but also hinders the healthy development of the medical industry. With the continuous improvement of cloud computing and big data technologies, the Internet of Things technology

has been rapidly developed. Radio frequency identification (RFID) is one of the core technologies of the Internet of Things. The application of the RFID system to the medical system can effectively solve this problem of medical privacy. RFID tags in the system can collect useful information and conduct data exchange and processing with a back-end server through the reader. The whole process of information interaction is mainly in the form of ciphertext. In the context of the Internet of Things, the paper presents a lightweight RFID medical privacy protection scheme. The scheme ensures security privacy of the collected data via secure authentication. The security analysis and evaluation of the scheme indicate that the protocol can effectively prevent the risk of medical privacy data being easily leaked.

### **3. AMBER: DECOUPLING USER DATA FROM WEB APPLICATIONS**

User-generated content is becoming increase common on the web, but current Web applications isolate their user's data, enabling only restricted sharing and cross-service integration. We believe users should be able to share their data in Seamlessly between their applications and with other users. To that end, we propose Amber, an architecture that decouples user's data from applications, while providing applications with powerful global queries to find user data. We demonstrate how multi-user applications, such as e-mail, can be use these global queries to efficiently collect and monitor relevant data created by other users. Amber puts users in control of which applications they use with their data and with whom it is shared, and enables a new class of applications by removing the artificial partitioning of user's data by application.

### **4. ENHANCING SELECTIVELY IN BIG DATA**

Today's companies collect immense amounts of personal data and enable wide access to it within the company. This exposes the data to external

hackers and privacy-transgressing employees. This study shows that, for a wide and important class of workloads, only a fraction of the data is needed to approach state-of-the-art accuracy. We propose selective data systems that are designed to pinpoint the data that is valuable for a company's current and evolving workloads. These systems limit data exposure by setting aside the data that is not truly valuable.

## **2.1 EXISTING SYSTEM**

In existing private data centers(PDC) is done manually without user permissions and security. Data is given to third party person. All service provider such as online social network or cloud storage will store some type of user data and they can sale that data to other organization for their own benefits and user has no control on his data as that data is saved on third party server.

### **DISADVANTAGE :**

#### **Low level security (password , fingerprint , facial recognition ,OTP & etc)**

- In online banking , the user can't connect directly to the bank server for his requirements like money transactions .There will be the third party control,where they will check the user details and gives access to the bank server, in this way the third party is involved in between user and the bank for their own benefits they can sell the user data to other organizations .

## **2.3 PROPOSED SYSTEM**

To overcome this issue we proposed private data centre with block chain to provide security to users data. We can share data with high security without involving other persons that is third-party user. By block chain ,we designed a secure networking architecture(Sec Net) to significantly improve

the security of data sharing and then the security of whole network.

## **2.4 MERITS OF PROPOSED SYSTEM**

- Full security
- Transparency
- Immutability
- Without dependency on third party

## **2.5 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

## **ECONOMIC FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

## **TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

## **SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system

### **3. SOFTWARE DEVELOPMENT LIFECYCLE (SDLC)**

SDLC is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high- quality software that meets customer expectations. The system development should be complete in the pre- defined time frame and cost. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life cycle has its own process and deliverables that feed into the next phase. SDLC stands for Software Development Lifecycle.

#### **3.1 SOFTWARE PROCESS MODELS**

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models".

Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

Following are the most important and popular SDLC models followed in the industry –

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Model

Other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.

## **SPIRAL MODEL:**

This document plays a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

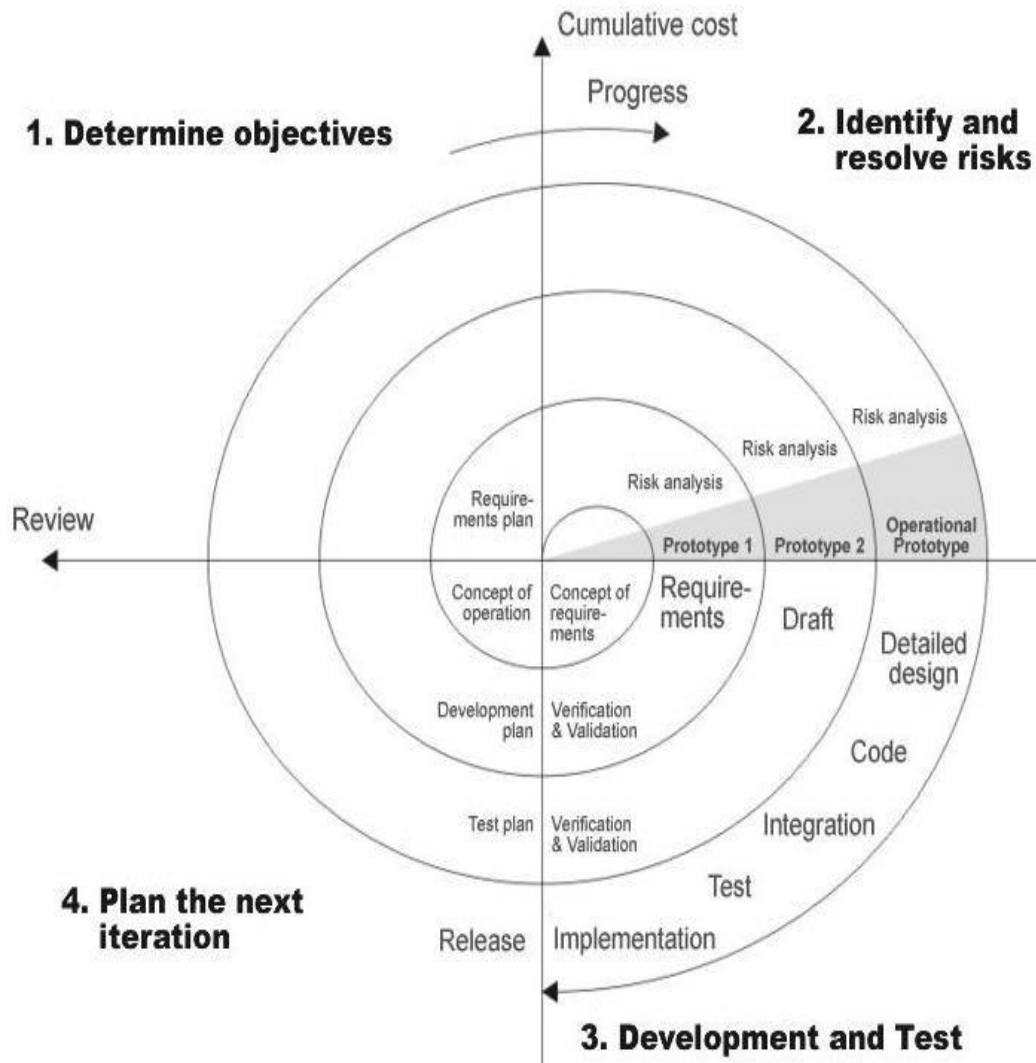
As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

### **The steps for Spiral Model can be generalized as follows:**

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:

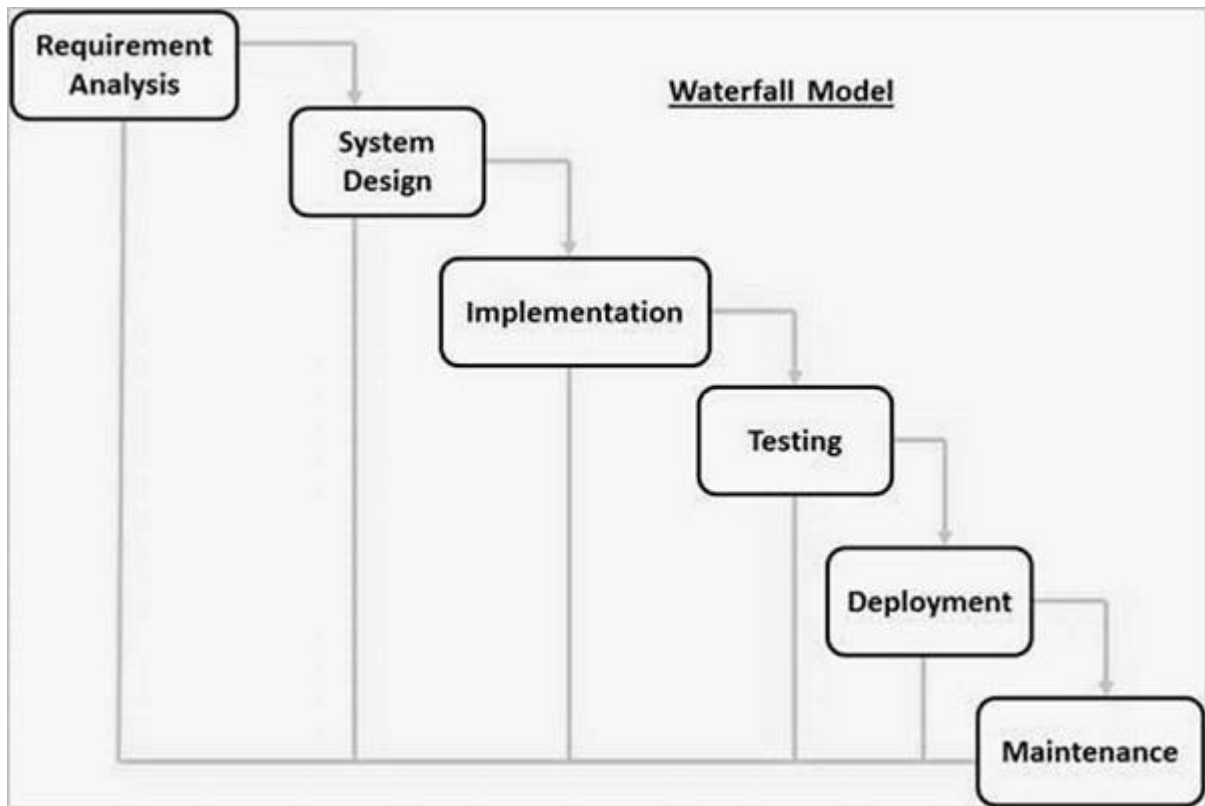


- a. Evaluating the first prototype in terms of its strengths, weakness, and risks.
  - b. Defining the requirements of the second prototype.
  - c. Planning a designing the second prototype.
  - d. Constructing and testing the second prototype.
- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
  - The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
  - The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
  - The final system is constructed, based on the refined prototype.
- 
- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.



## ADVANTAGES

1. Estimates (i.e. budget, schedule etc..) become more realistic as work progresses, because important issues discovered earlier.
2. It is more able to cope with the changes that are software development generally entails.

**WATERFALL MODEL:**

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software's. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

**Stage 1: Planning and Requirement Analysis**

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

## **Stage 2: Defining Requirements**

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

## **Stage 3: Designing the Product Architecture**

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third-party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the

minutest of the details in DDS.

### **Stage 4: Building or Developing the Product**

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high-level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

### **Stage 5: Testing the Product**

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

### **Stage 6: Deployment in the Market and Maintenance**

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

## **4. SYSTEM ANALYSIS**

### **4.1 REQUIREMENTS**

#### **4.1.1 FUNCTIONAL REQUIREMENTS: -**

- Security
- Reliability
- Maintainability

#### **4.1.2 NON-FUNCTIONAL REQUIREMENTS: -**

- Scalability
- Capacity
- Availability

#### **4.1.3 SOFTWARE REQUIREMENTS: -**

- Operating system : Windows 7, XP, 8
- Programming language : python 3.7
- Database : MySQL

#### **4.1.4 HARDWARE REQUIREMENTS: -**

➤ Processor : Pentium IV or higher

➤ RAM : 256 MB

➤ Hard Disk : 512MB

## 4.2 ALGORITHM OF THE PROJECT

### SHA-256

The SHA-256 algorithm is one flavor of SHA-2 (Secure Hash Algorithm 2), which was created by the National Security Agency in 2001 as a successor to SHA-1. SHA-256 is a patented cryptographic hash function that outputs a value that is 256 bits long. In encryption, data is transformed into a secure format that is unreadable unless the recipient has a key. In its encrypted form, the data may be of unlimited size, often just as long as when unencrypted. In hashing, by contrast, data of arbitrary size is mapped to data of fixed size.

mapping (hash => struct) public Data;

mapping (pubkey => int) public balance;

mapping (address => hash) private reverseIndex;

**procedure** RegisterData(hash,description,address, price)

Data[hash].owner  $\leftarrow$  msg.sender

Data[hash].address  $\leftarrow$  address

Data[hash].description  $\leftarrow$  description

Data[hash].price  $\leftarrow$  price

Data[hash].subscribers  $\leftarrow$  []

reverseIndex[address]  $\leftarrow$  hash

return TRUE

**end procedure**



**procedure** WithdrawData(*hash*)

require (Data[hash].owner == msg.sender)

reverseIndex[Data[hash].address]  $\leftarrow$  NULL

Data[hash]  $\leftarrow$  NULL

**end procedure**

**procedure** SubscribeData(*hash*)

require (balance[msg.sender]  $\geq$  Data[hash].price)

balance[msg.sender]  $\mathrel{-}=$  Data[hash].price

Data[hash].subscribers += msg.sender

**end procedure**

**procedure** RequestDataWithAddress(*address*)

require (reverseIndex[address]  $\neq$  NULL)

hash  $\leftarrow$  reverseIndex[address]

require (msg.sender  $\in$  Data[hash].subscribers)

return AccessToken for address with TTL

**end procedure**

**procedure** RequestDataWithHash(*hash*)

require (msg.sender  $\in$  Data[hash].subscribers)

address  $\leftarrow$  Data[hash].address

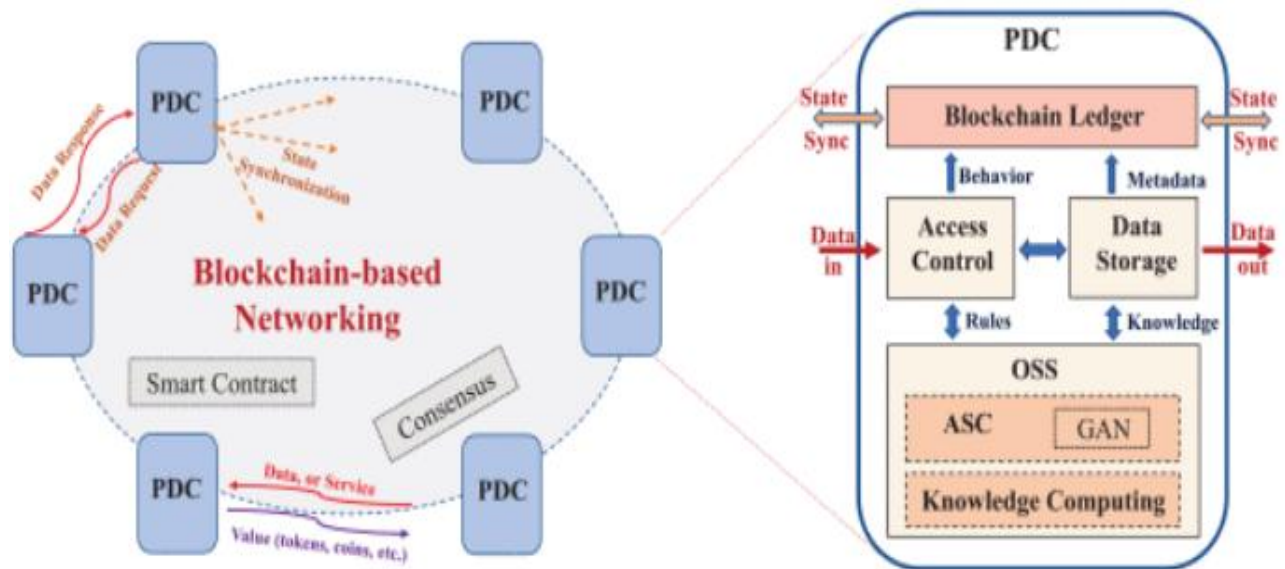
return AccessToken for address with TTL

**end procedure**

## 5. SYSTEM DESIGN

### 5.1 SYSTEM ARCHITECTURE

Architecture diagram is a diagram of a system, in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation.



## 5.2 UNIFIED MODELING LANGUAGE (UML)

### STAR UML

StarUML is a sophisticated software modeler aimed to support agile and concise modeling. The main targets of users are: Agile and small development teams, Professional persons.

- VERSIONS OF UML

- Version 2.0.0 (2014/12/29).
- Version 3.0.0 (2018/06/01).
- Version 4.0.0 (2020/10/29).

- VERSION 4.0.0

The new V4 supports a new modeling language SysML (System Modeling Language) and three more UML diagrams: Timing Diagram, Interaction Overview Diagram and Information Flow Diagram. The new V4 has additional features as below.

### FEATURES

- SysML Support.
- Additional UML Diagrams (Timing, Interaction Overview, Information Flow)
- MacBook Pro's Touch Bar Support.
- Command Palette.
- Tag Editor.
- Resolved Issues. Tag editor #3. Linux distributions for .deb and .rpm (no more AppImage) #172. ...
- Download: macOS | Windows | Linux (.deb) | Linux (.rpm)

## **UML Concepts**

The Unified Modeling Language (UML) is a standard language for writing software blue prints. The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting the artifacts of a software intensive system.

The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modeling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modeling yields an understanding of a system.

### **Building Blocks of the UML:**

The vocabulary of the UML encompasses three kinds of building blocks:

- Things
- Relationships
- Diagrams

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

### **Things in the UML:**

There are four kinds of things in the UML:

- Structural things
- Behavioral things
- Grouping things

- Annotational things

**Structural things** are the nouns of UML models. The structural things used in the project design are:

First, a **class** is a description of a set of objects that share the same attributes, operations, relationships and semantics.

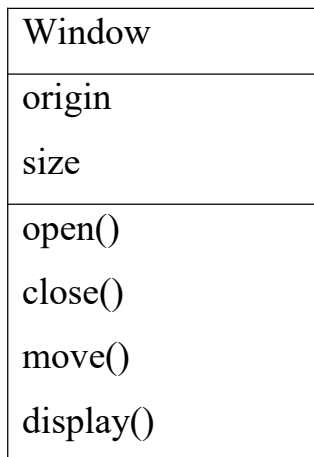


Fig: Classes

Second, a **use case** is a description of set of sequence of actions that a system performs that yields an observable result of value to particular actor.

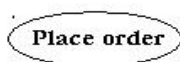


Fig: Use Cases

Third, a node is a physical element that exists at runtime and represents a computational resource, generally having at least some memory and often processing capability.

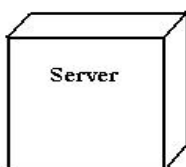


Fig: Nodes

**Behavioral things** are the dynamic parts of UML models. The behavioral thing used is:

### Interaction:

An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves a number of other elements, including messages, action sequences (the behavior invoked by a message, and links (the connection between objects).



Fig: Messages

### Relationships in the UML:

There are four kinds of relationships in the UML:

- Dependency
- Association
- Generalization
- Realization

A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).



Fig: Dependencies

An **association** is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.



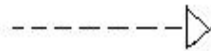
Fig: Association

A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element(the parent).



**Fig: Generalization**

A **realization** is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.



**Fig: Realization**

### 5.2.1 USE CASE DIAGRAM

A use case diagram is a graph of actors set of use cases enclosed by a system boundary, communication associations between the actors and users and generalization among use cases. The use case model defines the outside(actors) and inside(use case) of the system's behavior.

use case diagram is quite simple in nature and depicts two types of elements: one representing the business roles and the other representing the business processes.



an actor in a use case diagram

To identify an actor, search in the problem statement for business terms that portray roles in the system. For example, in the statement "patients visit



the doctor in the clinic for medical tests," "doctor" and "patients" are the business roles and can be easily identified as actors in the system.

**Use case:** A use case in a use case diagram is a visual representation of a distinct business functionality in a system. The key term here is "distinct business functionality." To choose a business process as a likely candidate for modeling as a use case, you need to ensure that the business process is discrete in nature.

As the first step in identifying use cases, you should list the discrete business functions in your problem statement. Each of these business functions can be classified as a potential use case. Remember that identifying use cases is a discovery rather than a creation. As business functionality becomes clearer, the underlying use cases become more easily evident. A use case is shown as an ellipse in a use case diagram.

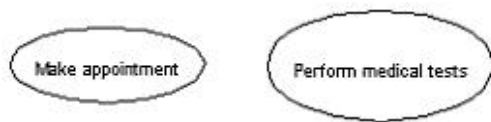
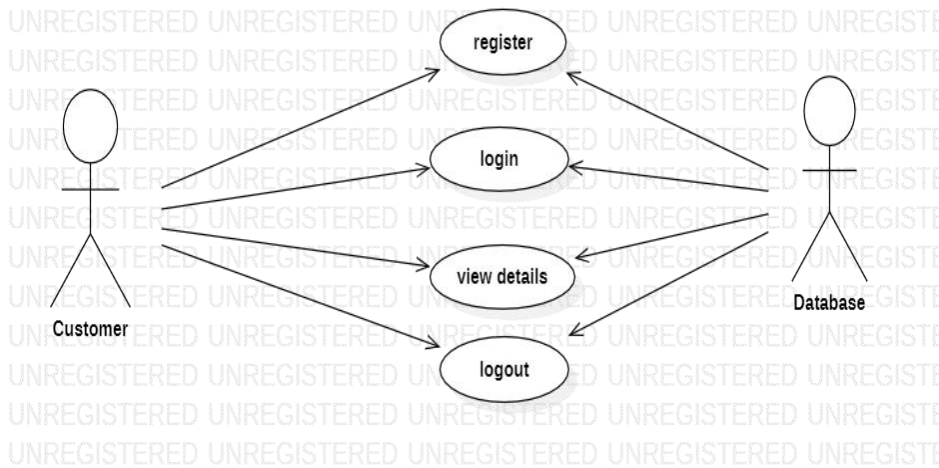


Figure: use cases in a use case diagram

Figure shows two uses cases: "Make appointment" and "Perform medical tests" in the use case diagram of a clinic system. As another example, consider that a business process such as "manage patient records" can in turn have sub-processes like "manage patient's personal information" and "manage patient's medical information." Discovering such implicit use cases is possible only with a thorough understanding of all the business processes of the system through discussions with potential users of the system and relevant domain knowledge.

## CUSTOMER MODULE :



## COMPANY MODULE:

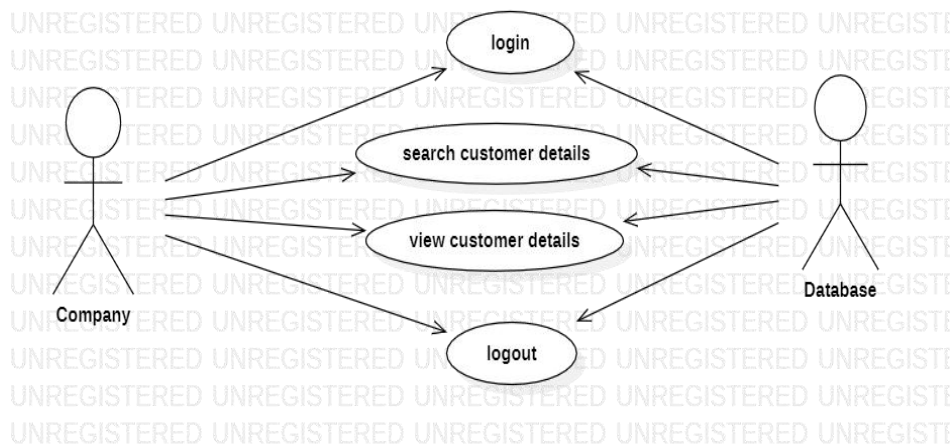


Figure 4.2 Use case Diagram

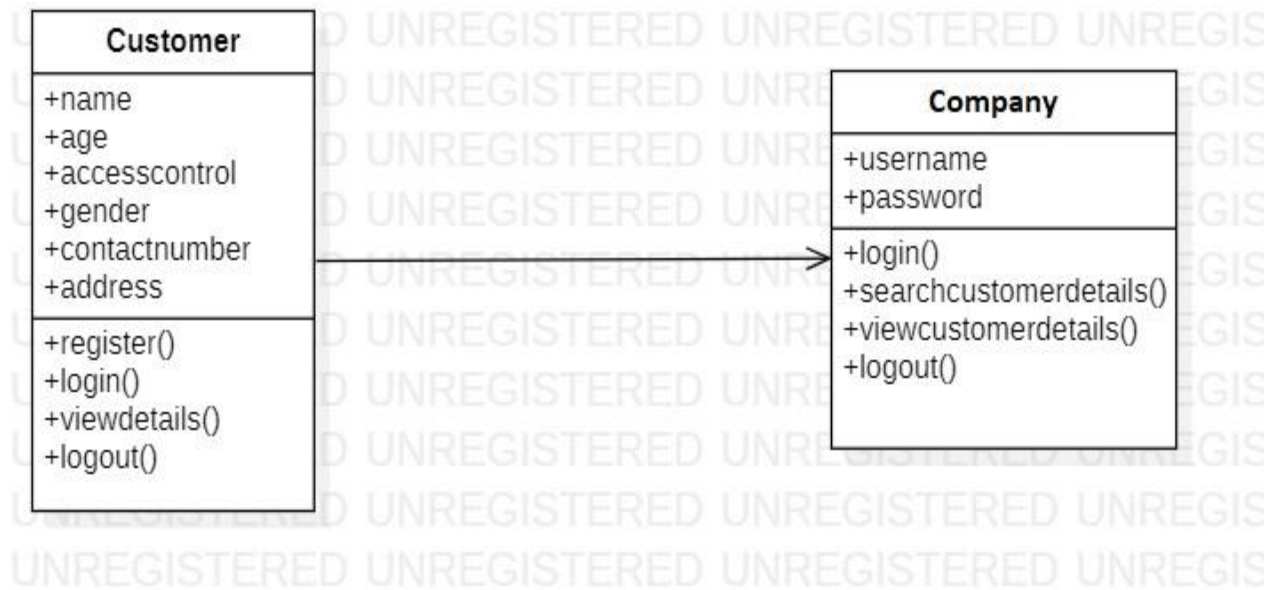
### 5.2.2 CLASS DIAGRAM

An object is any person, place, thing, concept, event, screen, or report applicable to your system. Objects both know things (they have attributes) and they do things (they have methods).

A class is a representation of an object and, in many ways, it is simply a template from which objects are created. Classes form the main building blocks of an object-oriented application. Although thousands of students attend the university, you would only model one class, called Student, which would represent the entire collection of students.

#### **Responsibilities**

Classes are typically modeled as rectangles with three sections: the top section for the name of the class, the middle section for the attributes of the class, and the bottom section for the methods of the class. Attributes are the information stored about an object, while methods are the things an object or class do. For example, students have student numbers, names, addresses, and phone numbers. Those are all examples of the attributes of a student. Students also enroll in courses, drop courses, and request transcripts. Those are all examples of the things a student does, which get implemented (coded) as methods. You should think of methods as the object oriented equivalent of functions and procedures.



**Figure 5.3 Class Diagram**

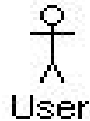
### 5.2.3 SEQUENCE DIAGRAM

UML sequence diagrams are used to represent the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interactions of the header elements, which are displayed horizontally at the top of the diagram.

Sequence Diagrams are used primarily to design, document and validate the architecture, interfaces and logic of the system by describing the sequence of actions that need to be performed to complete a task or scenario. UML sequence diagrams are useful design tools because they provide a dynamic view of the system behavior which can be difficult to extract from static diagrams or specifications.

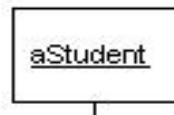
#### **Actor**

Represents an external person or entity that interacts with the system



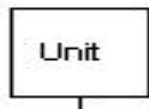
## Object

Represents an object in the system or one of its components



## Unit

Represents a subsystem, component, unit, or other logical entity in the system (may or may not be implemented by objects)



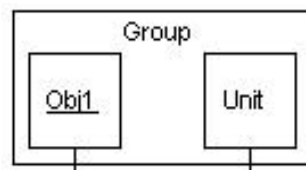
## Separator

Represents an interface or boundary between subsystems, components or units (e.g., air interface, Internet, network)



## Group

Groups related header elements into subsystems or components



## Sequence Diagram Body Elements

### Action

Represents an action taken by an actor, object or unit



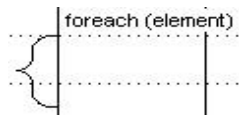
### Asynchronous Message

An asynchronous message between header elements



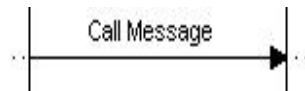
### Block

A block representing a loop or conditional for a particular header element



### Call Message

A call (procedure) message between header elements



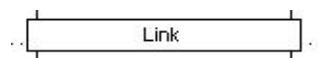
### Create Message

A "create" message that creates a header element (represented by lifeline going from dashed to solid pattern)

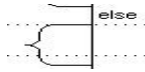


### Diagram Link

Represents a portion of a diagram being treated as a functional block. Similar to a procedure or function call that abstracts functionality or details not shown at this level. Can optionally be linked to another diagram for elaboration.

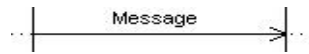


Else Block Represents an "else" block portion of a diagram block



## Message

A simple message between header elements



## Return Message

A return message between header elements

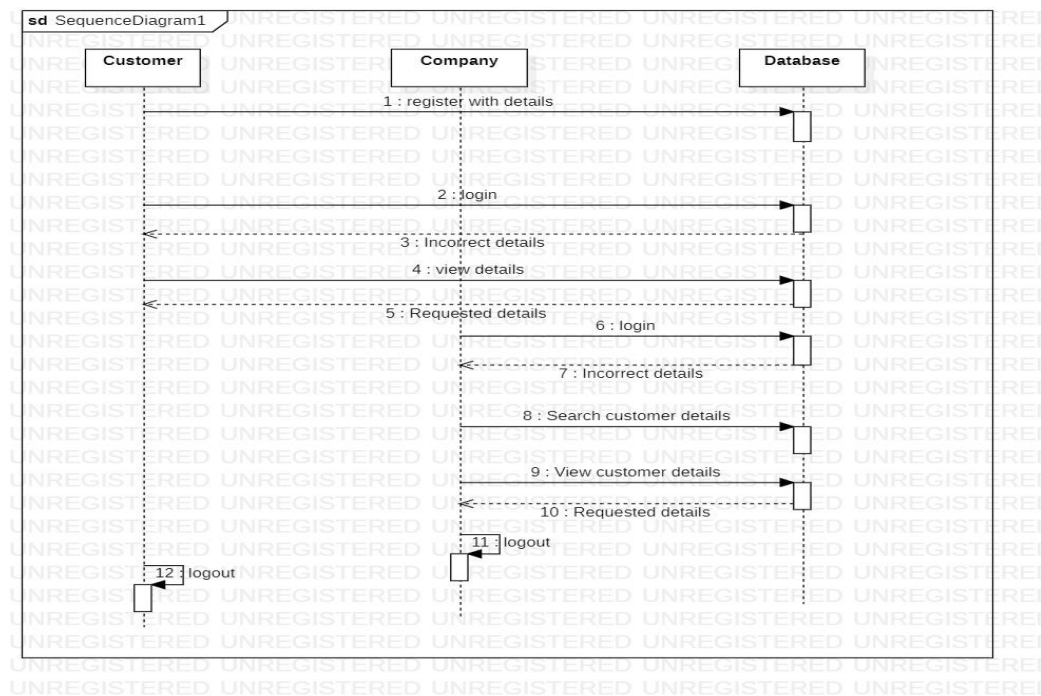
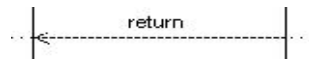


Figure 5.4 Sequence Diagram

## 5.2.4 ACTIVITY DIAGRAM

Activity diagrams represent the business and operational workflows of a system. An Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state.

So, what is the importance of an Activity diagram, as opposed to a State diagram? A State diagram shows the different states an object is in during the lifecycle of its existence in the system, and the transitions in the states of the objects. These transitions depict the activities causing these transitions, shown by arrows.

An Activity diagram talks more about these transitions and activities causing the changes in the object states.

### Defining an Activity diagram

Let us take a look at the building blocks of an Activity diagram.

### Elements of an Activity diagram

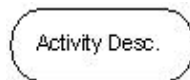
An Activity diagram consists of the following behavioral elements:

**Initial Activity:** This shows the starting point or first activity of the flow. Denoted by a solid circle. This is similar to the notation used for Initial State.



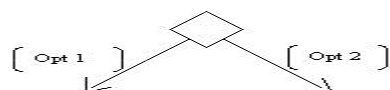
**Fig: Initial State**

**Activity:** Represented by a rectangle with rounded (almost oval) edges.



**Fig: Action State**

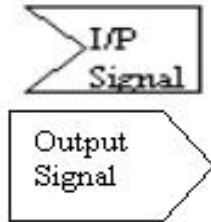
**Decisions:** Similar to flowcharts, a logic where a decision is to be made is depicted by a diamond, with the options written on either sides of the arrows emerging from diamond within box brackets.



**Fig: Decision**

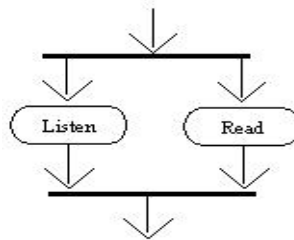


**Signal:** When an activity sends or receives a message, that activity is called a signal. Signals are of two types: Input signal (Message receiving activity) shown by a concave polygon and Output signal (Message sending activity) shown by a convex polygon.



**Fig: Signal**

**Concurrent Activities:** Some activities occur simultaneously or in parallel. Such activities are called concurrent activities. For example, listening to the lecturer and looking at the blackboard is a parallel activity. This is represented by a horizontal split (thick dark line) and the two concurrent activities next to each other, and the horizontal line again to show the end of the parallel activity.



**Fig: Horizontal**

**Final Activity:** The end of the Activity diagram is shown by a bull's eye symbol, also called as a final activity.



**Fig: Final State**

### **5.2.6 OBJECT DIAGRAM**

An object diagram in the Unified Modeling Language (UML), is a diagram that shows a complete or partial view of the structure of a modeled system at a specific time.

An Object diagram focuses on some particular set of object instances and attributes, and the links between the instances. A correlated set of object diagrams provides insight into how an arbitrary view of a system is expected to evolve over time. Object diagrams are more concrete than class diagrams, and are often used to provide examples, or act as test cases for the class diagrams. Only those aspects of a model that are of current interest need be shown on an object diagram.

### **5.2.7 STATE CHART DIAGRAM**

State chart diagram is used to describe the states of different objects in its life cycle. So the emphasis is given on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately. State chart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.<sup>4</sup>

Before drawing a State chart diagram, we must have clarified the following points:

- Identify important objects to be analyzed.
- Identify the states.
- Identify the events.

## 6.SYSTEM IMPLEMENTATION

### 6.1 MODULES

This project consists of three modules:

➤ **Customer** : Customer first create his profile with all details and then select desired company with whom he wishes to share/subscribe data. While creating profile application will create Blockchain object with allowable permission and it will allow only those companies to access data.

**Customer Login** : Customer can login to application with his profile id.

➤ **Admin** : The admin will check the authorization of the company and customer for providing the demanded information .

➤ **Company** :

Company 1 and Company 2 are using in this application as two organizations with whom customer can share data . At a time any company can login to application .

## 6.2 SAMPLE CODE

```
Views.py

from django.shortcuts import

render from django.template

import RequestContext

from django.contrib import

messages import pymysql

from django.http import

HttpResponse import datetime

import

t json

import

t time

from hashlib import sha256

class Blockchain:

#difficulty of our PoW
```

algorithm difficulty = 2

def init(self):

self.unconfirmed\_transactions = []

def create\_genesis\_block(self):

"""

A function to generate genesis block and

appends it to the chain. The block has index

0, previous\_hash as 0, and a valid hash.

"""

genesis\_block = Block(0, [], time.time(),

"0") genesis\_block.hash =

genesis\_block.compute\_hash()

self.chain.append(genesis\_block)

@property

def

```
last_block(se
```

```
lf): return
```

```
self.chain[-1]
```

```
def add_block(self, block, proof):  
    """
```

A function that adds the block to the chain after verification.

Verification includes:

- \* Checking if the proof is valid.

- \* The previous\_hash referred in the block and the hash

of latest block in the chain match.

```
    """
```

```
previous_hash = self.last_block.hash
```

```
if previous_hash != block.previous_hash:
```

```
    return False
```

```
if not self.is_valid_proof(block, proof):
```

```
    return False
```

```
        block.hash = proof

        print("main

        "+str(block.hash))

        self.chain.append(blo

        ck)

        return True

def is_valid_proof(self, block, block_hash):

    """

    Check if block_hash is valid hash of block

    and satisfies the difficulty criteria.

    """

    print("compare "+str(block_hash == block.compute_hash())+"

    "+block.compute_hash()+" "+str(block_hash.startswith('0' *

    Blockchain.difficulty)))

    return (block_hash.startswith('0' *

    Blockchain.difficulty) and
```

```
block_hash ==
```

```
block.compute_hash()
```

```
def proof_of_work(self, block):
```

```
    """
```

Function that tries different values of nonce

to get a hash that satisfies our difficulty

criteria.

```
    block.nonce = 0
```

```
    computed_hash = block.compute_hash()
```

```
    while not computed_hash.startswith('0' * Blockchain.difficulty):
```

```
        block.nonce += 1
```

```
        computed_hash =
```

```
        block.compute_hash()
```

```
    return computed_hash
```

```
def add_new_transaction(self,
```

```
transaction):
```

```
    self.unconfirmed_transactions.append(t
```



ransaction)

```
def mine(self):
```

```
    """
```

This function serves as an interface to add the

pending transactions to the blockchain by

adding them to the block and figuring out

Proof Of Work.

```
    """
```

```
    if not self.unconfirmed_transactions:
```

```
        return False
```

```
    last_block = self.last_block
```

```
    new_block =
```

```
        Block(index=last_block.index + 1,
```

```
              transactions=self.unconfirmed_tr
```

```
              ansactions,
```

```
              timestamp=time.time(),
```

```
              previous_hash=last_block.hash)
```

```
proof = self.proof_of_work(new_block)
```

```
self.add_block(new_block, proof)
```

```
self.unconfirmed_transaction
```

```
ns = [] return proof
```

```
class Block:
```

```
def __init__(self, index, transactions, timestamp, previous_hash):
```

```
    self.index = index
```

```
    self.transactions =
```

```
        transactions
```

```
    self.timestamp =
```

```
        timestamp
```

```
    self.previous_hash =
```

```
        previous_hash self.nonce = 0
```

```
    def compute_hash(self):
```

```
        """
```

A function that return the hash of the

```

        block contents. """

    block_string = json.dumps(self._dict_,

                                sort_keys=True) return

    sha256(block_string.encode()).hexdigest()

    def AccessData(request):

        if request.method == 'GET':

            return render(request, 'AccessData.html', {})

        def index(request):

            if request.method ==

                'GET': return render(request,

                    'index.html', {})

        def

            CreateProfile(reque

                st): if

                    request.method ==

                        'GET':

                            return render(request, 'CreateProfile.html', {})

```

```

def Hospital(request):
    if request.method == 'GET':
        return render(request, 'Hospital.html', {})

def Patient(request):
    if request.method == 'GET':
        return render(request, 'Patient.html', {})

def PatientLogin(request):
    if request.method == 'POST':
        pid = request.POST.get('t1', False)

strdata = '<table border=1 align=center
width=100%><tr><th>Patient ID</th><th>Patient
Name</th><th>Age</th><th>Problem
Description</th><th>Profile Date</th><th>Access
Control</th><th>Gender</th><th>Contact
No</th><th>Address</th><th>Block Chain
Hashcode</th><th>Revenue</th></th></tr><tr>'

con = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root',

```

```
password = 'root', database =
```

```
'SecuringData',charset='utf8')
```

```
with con:
```

```
cur = con.cursor()
```

```
cur.execute("select * FROM
```

```
patients")
```

```
rows =
```

```
cur.fetchall
```

```
() for row
```

```
in rows:
```

```
if str(row[0]) == pid:
```

```
strdata+='<td>'+str(row[0])+'</td><td>'+row[1]+'</td><td>'+str(row[2])+
```

```
'</td><
```

```
td>'+str(row[3])+'</td><td>'+str(row[4])+'</td><td>'+row[5]+'</td><td>'
```

```
+row[6
```

```
]+</td><td>'+row[7]+'</td><td>'+row[8]+'</td><td>'+row[9]+'</td><td>
```

```
>'+str(row[10])+'</td></tr><tr>'
```

```
context= {'data':strdata}  
  
return render(request, 'ViewData.html', context)
```

```
def HospitalLogin(request):  
  
    if request.method ==  
  
        'POST': username =  
  
            request.POST.get('t1', False)  
  
            password =  
  
            request.POST.get('t2', False)  
  
            file =  
  
                open('session.txt', '  
  
                    w')  
  
                file.write(username  
  
                    e) file.close()  
  
            if username == 'Hospital1' and password == 'Hospital1' or username  
                == 'Hospital2'  
  
                and password ==
```

```
'Hospital2': context=

{'data': 'welcome '+username}

return render(request,

                'HospitalScreen.html

                ', context) else:

    context= {'data': 'login failed'}

return render(request, 'Hospital.html', context)

def updateRevenue(value):

db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user =

                                'root', password

                                = 'root', database =

                                'SecuringData',charset='utf8')

db_cursor =

db_connection.cursor()

db_cursor.execute("update patients set revenue=revenue+0.5 where

                    patient_id='"+va

                    lue+"")

db_connection.co
```

```
mmmit()
```

```
def
```

```
    PatientDataAccess(re
```

```
quest): if
```

```
request.method ==
```

```
    'POST':
```

```
search =
```

```
request.POST.
```

```
get('t1', False)
```

```
user = "
```

```
with open("session.txt", "r") as file:
```

```
    for line in file:
```

```
        user = line.strip('\n')
```

```
strdata = '<table border=1 align=center
```

```
width=100%><tr><th>Patient
```

```
ID</th><th>Patient
```



Name</th><th>Age</th><th>Problem

Description</th><th>Profile Date</th><th>Access

Control</th><th>Gender</th><th>Contact No</th></th></tr><tr>

con = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root',

password = 'root', database =

'SecuringData',charset='utf8')

print("select \* FROM patients where problem\_desc like

"%" + search + "%") with con:

cur = con.cursor()

cur.execute("select \* FROM patients where problem\_desc like

"%" + search + "%") rows = cur.fetchall()

for row in rows:

arr =

row[5].split

(" ") if

len(arr) ==

1:

```
if arr[0] == user:
```

```
    updateRevenue(str(
```

```
        row[0]))
```

```
    strdata+='<td>'+str(row[0])+'</td><td>'+row[1]+'</td><td>'+str(row[2])+
```

```
        '</td><
```

```
td>'+str(row[3])+'</td><td>'+str(row[4])+'</td><td>'+row[5]+'</td><td>'
```

```
        +row[6]
```

```
    ]+'</td><td>'+row[7]+'</td></tr><tr>'
```

```
    if len(arr) == 2:
```

```
        if arr[0] == user or arr[1]
```

```
            == user:
```

```
                updateRevenue(str(ro
```

```
                    ow[0]))
```

```
    strdata+='<td>'+str(row[0])+'</td><td>'+row[1]+'</td><td>'+str(r
```

```
        ow[2])+'</td><
```

```

td>'+str(row[3])+'</td><td>'+str(row[4])+'</td><td>'+row[5]+'</
td><td>'+row[6

```

```

]+'</td><td>'+row[7]+'</td>

```

```

</tr><tr>' context=

```

```

{'data':strdata}

```

```

return render(request, 'ViewAccessData.html', context)

```

```

def

```

```

CreateProfileData(re

```

```

quest): if

```

```

request.method ==

```

```

'POST':

```

```

name = request.POST.get('t1',

```

```

False) age =

```

```

request.POST.get('t2', False)

```

```

problem =

```

```

request.POST.get('t3', False)

```

```
access_list =

request.POST.getlist('t4', False)

gender = request.POST.get('t5',

False)

contact =

request.POST.get('t6', False)

address =

request.POST.get('t7', False)

revenue = 0

cou

nt =

0

acce

ss =

"

for i in
```

```
range(len(access_li
st)):

access+=access_list

[i]+" " access =

access.strip()

db_connection = pymysql.connect(host='127.0.0.1',port =

3308,user = 'root', password = 'root', database =

'SecuringData',charset='utf8')

with db_connection:

cur = db_connection.cursor()

cur.execute("select count(*)

FROM patients") rows =

cur.fetchall()

for row in

rows: count

= row[0]
```

```
count =

count + 1

now = datetime.datetime.now()

current_time = now.strftime("%Y-%m-%d %H:%M:%S")

blockchain = Blockchain()

x = '{ "Patient_id":'+str(count)+'", "patient_name":'+name+'",

"age":'+age+'", "problem_desc":'+problem+'",

"profile_date":'+str(current_time)+'",

"access_data":'+str(access)+'", "gender":'+gender+'"}'

blockchain.add_new_transaction(json.loads(x))

hash = blockchain.mine()

db_cursor =

db_connection.cursor()

student_sql_query =

"INSERT INTO

patients(Patient_id,patient_name,age,problem_desc,profile_date,access_d

ata,gend

er,contact_no,address,blockchain_hash,revenue)
```

```
VALUES(""+str(count)+"", ""+name+"", ""+age+"", ""+problem+"", ""+c
    urrent_time+"",
    ""+str(access)+"", ""+gender+"", ""+contact+"", ""+address+"", ""+hash+
    "", ""+str(revenue)+"")"

db_cursor.execute(student_sql_
query) db_connection.commit()

print(db_cursor.rowcount, "Record
Inserted") if db_cursor.rowcount
    == 1:

context= {'data': 'Profile Creation Process Completed. Your Patient ID :
'+str(count)} return render(request, 'CreateProfile.html', context)

else:

context= {'data': 'Error in profile
creation process'} return render(request,
    'CreateProfile.html', context)
```

## **7. SYSTEM TESTING**

### **7.1 INTRODUCTION**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### **7.2 TESTING STRATEGIES**

In order to perform testing to generate a testcase that should be applied while testing, there have been developed various methods for designing the testcases.

There are generally two categories of test case design techniques. They are,

#### **White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.



## **Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software work.

## **7.3 LEVELS OF TESTING**

### **7.3.1 Unit Testing**

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs.

In this level, we aim to eliminate construction errors in Twitter Sentiment analysis before code is promoted, this strategy is intended to increase the quality of the resulting project as well as the efficiency of the overall development.

### **7.3.2 Integration Testing**

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or altogether. Normally the former is considered a better practice since it allows interface issues to be

located more quickly and fixed in Twitter Sentiment analysis project. Integration testing works to expose defects in interaction between integrated components (modules) like:

### **7.3.3 System Testing**

System Testing is a level of the software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. The process of testing an integrated system to verify that it meets specified requirements. Normally, independent Testers perform System Testing.

### **7.3.4 Acceptance Testing**

Acceptance Testing is a level of the software testing where a system is tested for that an acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

## **7.4 VALIDATIONS**

Introduction to Software Validation?

The concept of validation has been in vogue for centuries. Irrespective of the industry or products, validation ensures various critical aspects of a product and guarantees its success in the market as well as among users. Likewise, **software validation** plays an immensely significant role during

the software development life cycle (SDLC) and helps the testing and development teams to create a quality product. Therefore, in this article, we shall discuss the various aspects of software validation.

What is Software Validation?

Software Validation is a process of evaluating software product, so as to ensure that the software meets the pre-defined and specified business requirements as well as the end users/customers' demands and expectations.

It is basically, performed with the intent to check that whether the developed software is built as per pre-decided software requirement specifications (SRS) and if it caters to fulfil the customers' actual needs in the real environment.

Both, the verification and validation is a software testing activity, and verification is followed by the validation. Validation is usually carried out at the end of the software development.

## **7.5 TEST CASES**

### **USER REQUIREMENTS:**

1. Home
2. Register
3. Login
4. Company
5. User

**Home:**

Use case ID	Convergence of Blockchain with AI for data Security
Use case Name	Home button
Description	Display home page of application
Primary actor	User
Precondition	User must open application
Post condition	Display the Home Page of an application
Frequency of Use case	Many times
Alternative use case	N/A
Use case Diagrams	
Attachments	N/A

**Login Form:**

Use case ID	Convergence of Blockchain with AI for data Security
Use case Name	Login Form
Description	Display Login form to the User
Primary actor	User
Precondition	User must have username & password
Post condition	Display the Home Page
Frequency of Use case	Many times
Alternative use case	Forgot password
Use case Diagrams	
Attachments	N/A

**Administrator:**

Use case ID	Convergence of Blockchain with AI for data Security
Use case Name	Company
Description	View details of all users and view graph
Primary actor	User
Precondition	Must open the application home page
Post condition	View Customer Details
Frequency of Use case	Many times
Alternative use case	N/A
Use case Diagrams	
Attachments	N/A

**User:**

Use case ID	Convergence of Blockchain with AI for data Security
Use case Name	User
Description	View details
Primary actor	User
Precondition	User must be login
Post condition	View feedbacks
Frequency of Use case	Many times
Alternative use case	N/A
Use case Diagrams	
Attachments	Photos (if any)

**Customer :**

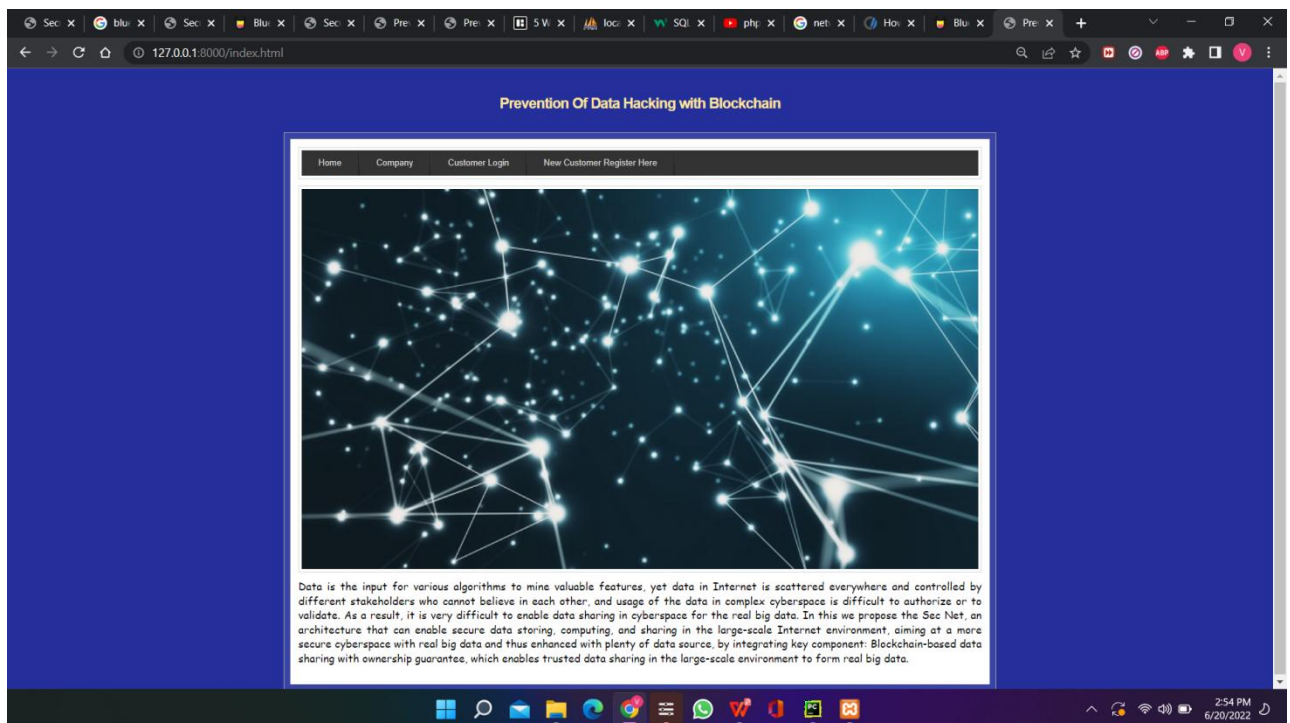
Use case ID	Convergence of Blockchain with AI for data Security
Use case Name	recruiter
Description	View Own Details
Primary actor	User
Precondition	user must be login
Post condition	View profile
Frequency of Use case	Many times
Alternative use case	N/A
Use case Diagrams	
Attachments	N/A

## 8. SCREENS

### 8.1 INPUT OUTPUT SCREENS

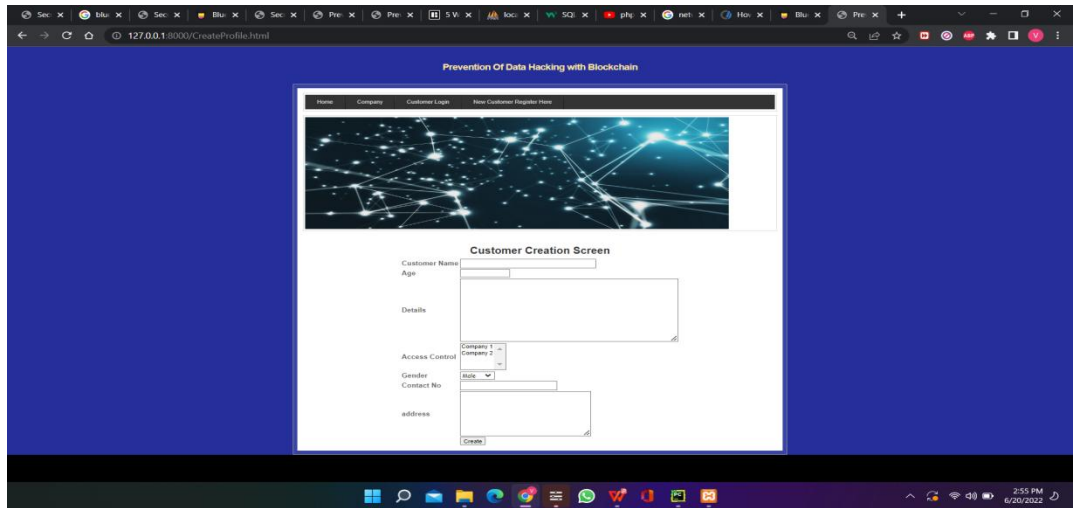
In settings file change port no from 3308 to 3306 and in 'views.py' file also change port no to 3306

Deploy code on DJANGO and start server and run in browser to get below screen

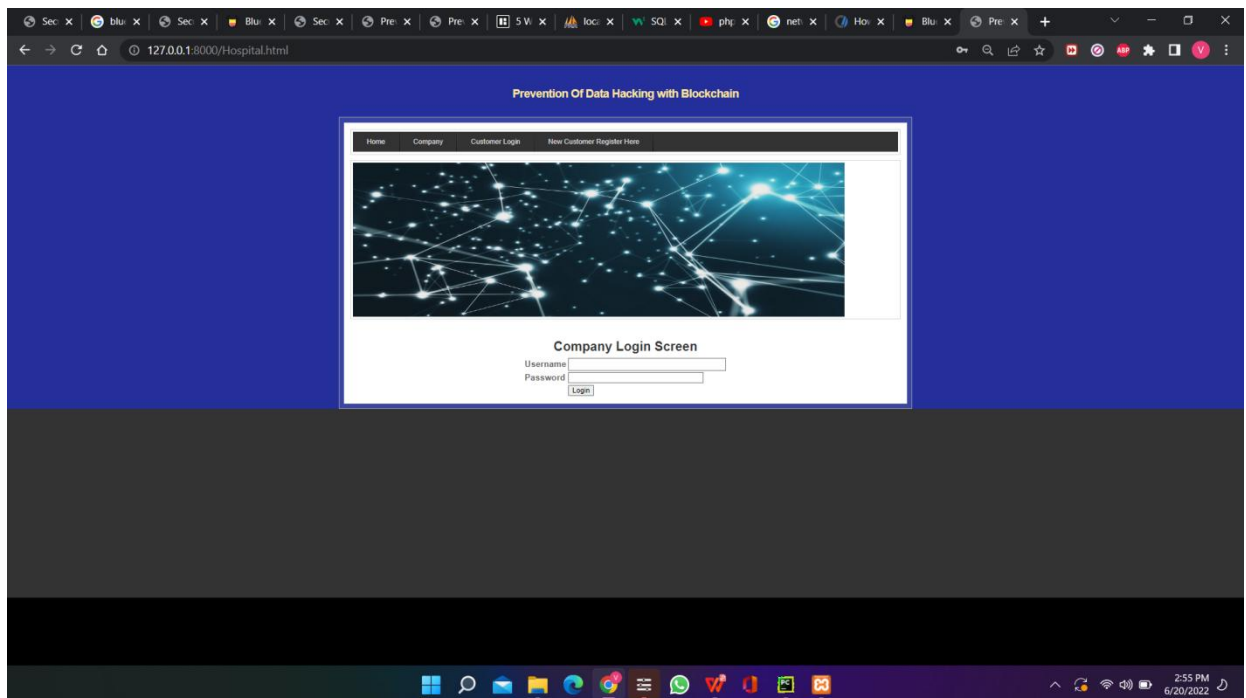


In above screen click on 'New Customer Register Here' link to get below screen

## Prevention Of Data Hacking with Block chain



In above screen I am adding customer details and selecting 'company 1' to share my data and if you want to share with two companies then hold 'CTRL' key and select both companies to give permission. Now press 'Create' button to create profile.

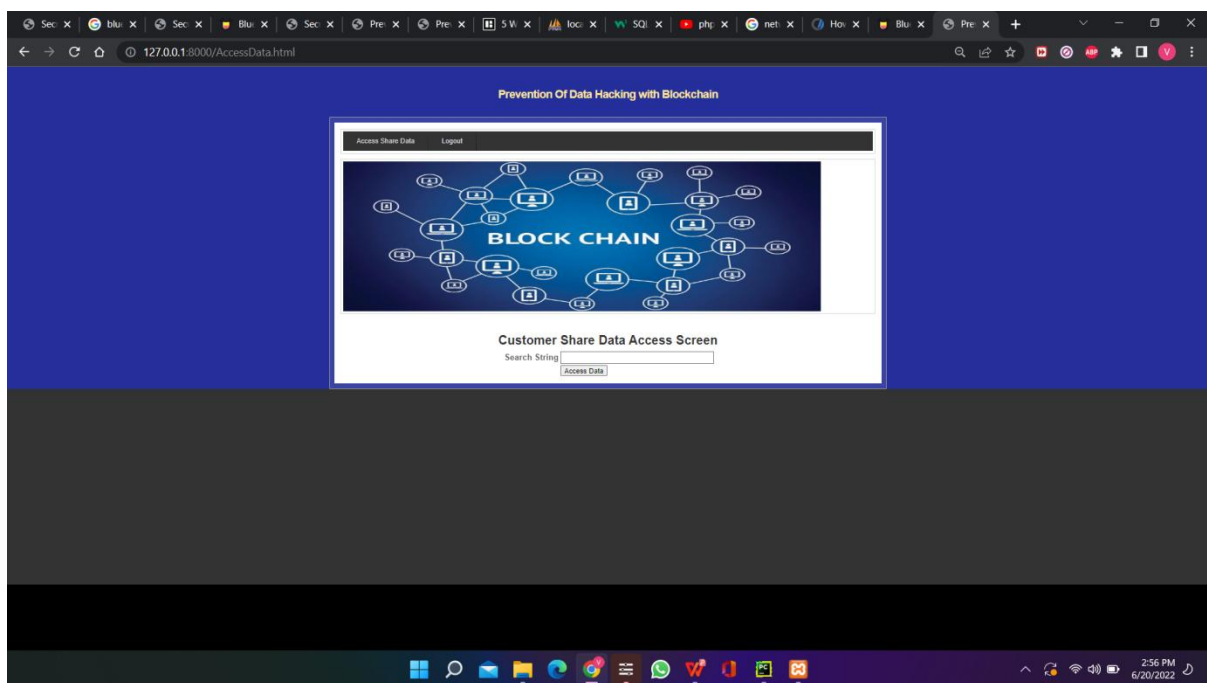


In above screen one customer is created with customer ID 1 and now company 1 can login and search and access this customer data as customer has given permission to company 1.

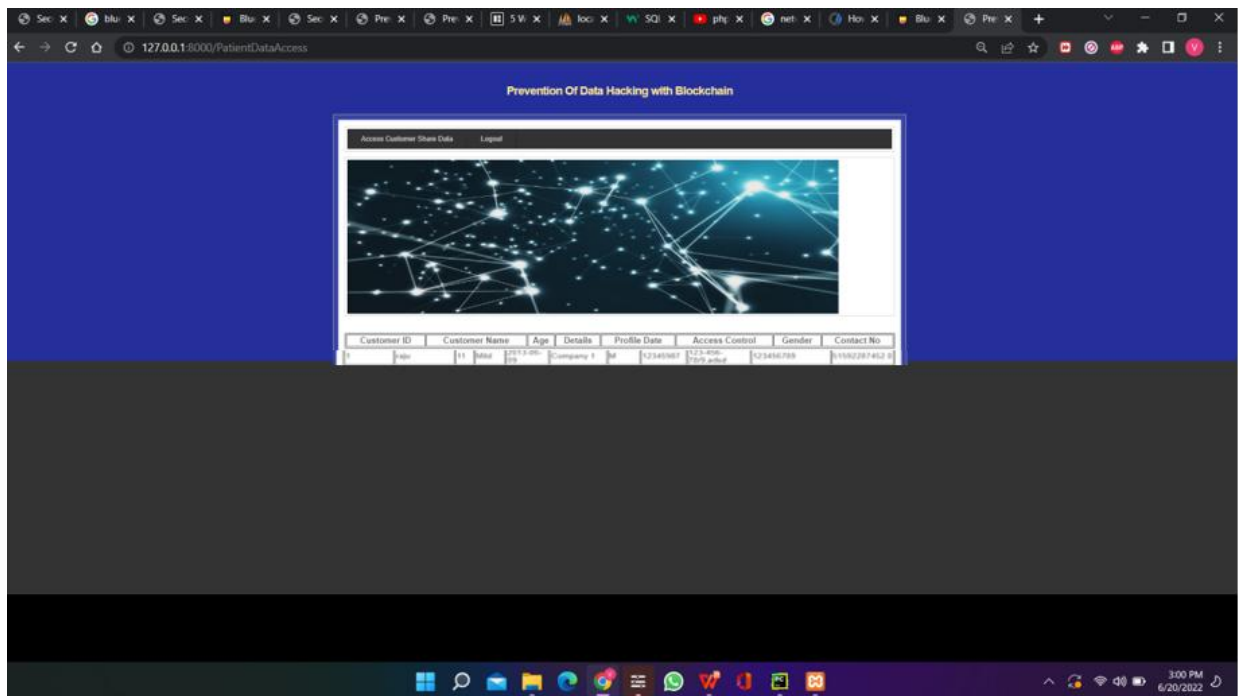


## Prevention Of Data Hacking with Block chain

In above screen to login as company 1 click on 'company' link to get above screen. Use 'company 1' as username and 'company1' as password to login as company1 and use company2 to login as company2. After login will get below screen click on 'Access Patient Share Data' link to search for patient details

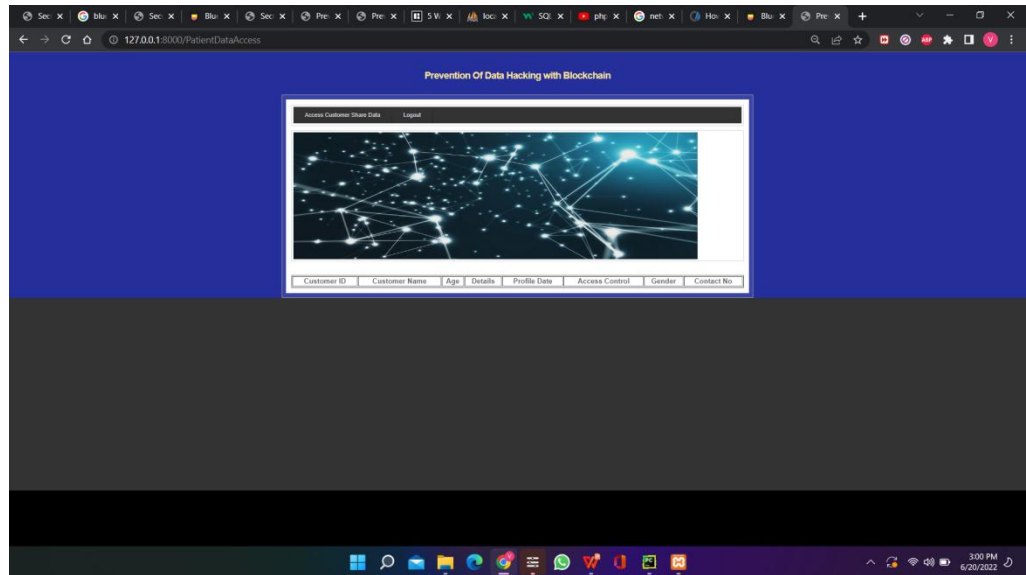


In above screen I want to search for all customers who are having identical address and then click on 'Access data' button to get below screen

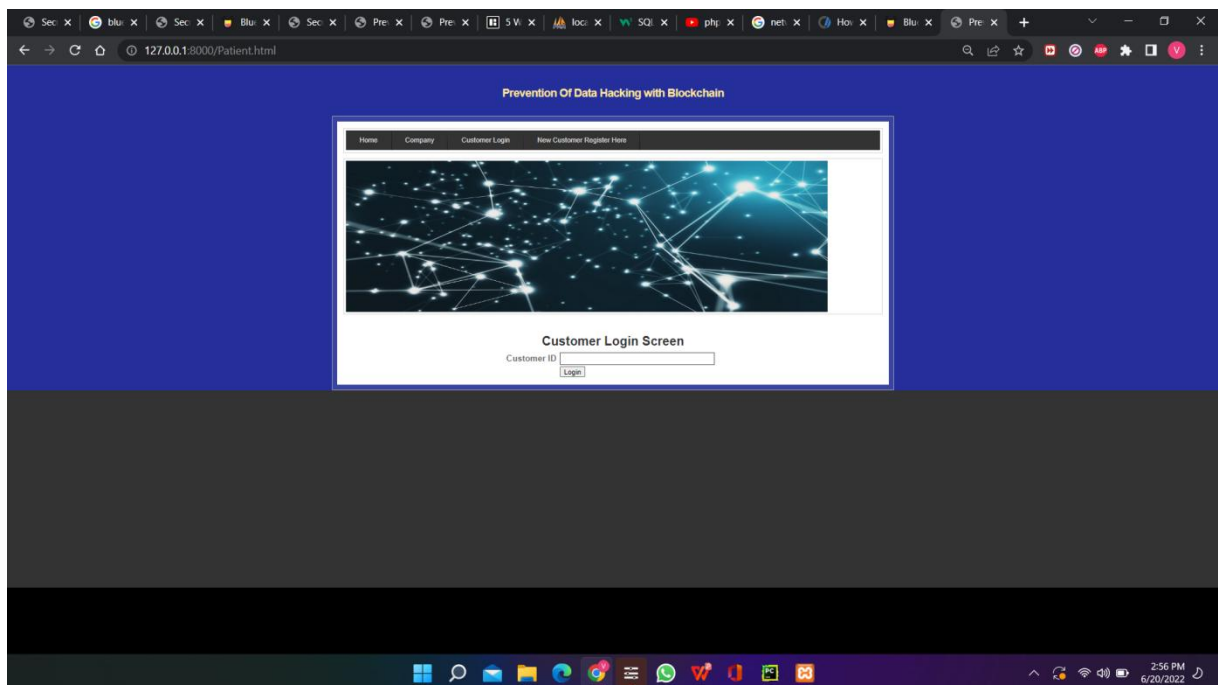


Now click on 'Access Patient Share Data' link and search for same address from company2. For above query will get below result

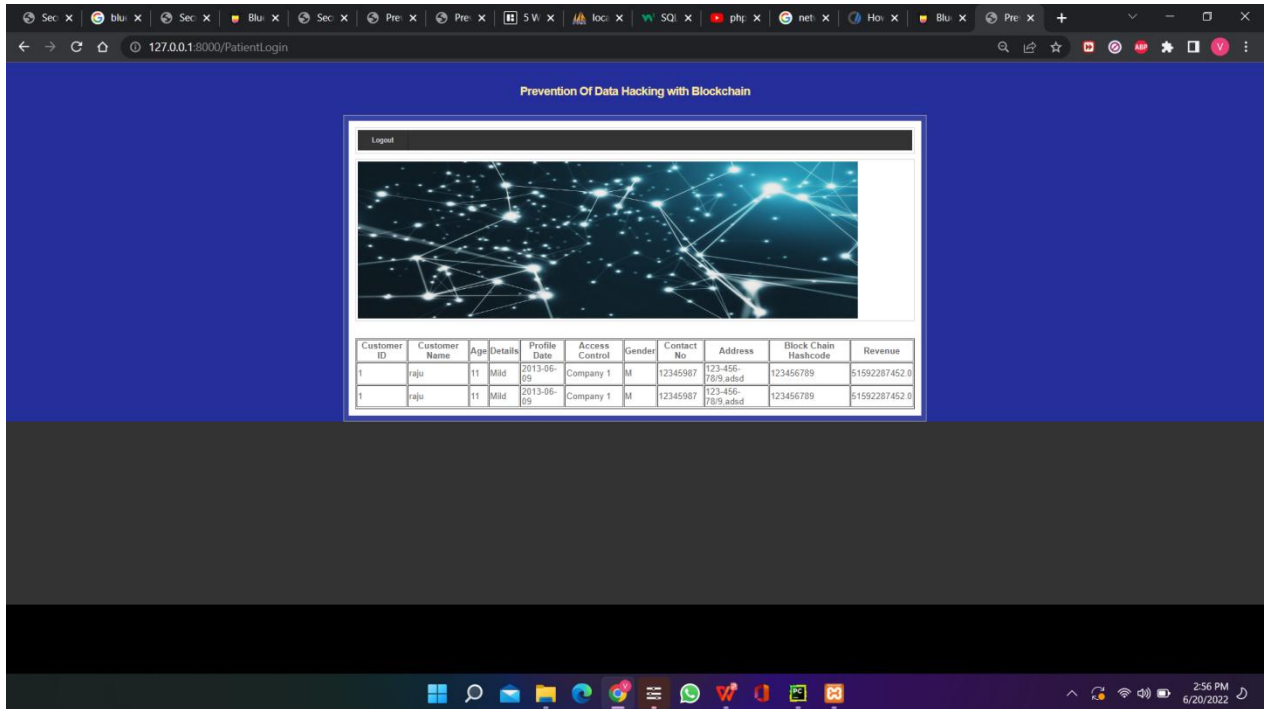
## Prevention Of Data Hacking with Block chain



In above screen no customer details are showing as company2 not having permission. So block chain allow only those users to access data who has permission. Now logout and login as customer by entering customer id in below screen



After login will get below details for customer1



Prevention Of Data Hacking with Blockchain

Logout

Customer ID	Customer Name	Age	Details	Profile Date	Access Control	Gender	Contact No	Address	Block Chain Hashcode	Revenue
1	nju	11	Mild	2013-06-09	Company 1	M	123456789	123456789	123456789	51592287452.0
1	nju	11	Mild	2013-06-09	Company 1	M	123456789	123456789	123456789	51592287452.0

In above screen we can see customer all details and hash code generated by block chain and in last column we can see patient reward revenue as 0.5 and it will get update upon every access from company user.

## **9. CONCLUSION**

In order to average Blockchain to fit the problems of abusing data and with the help of blockchain for trusted data management SecNet which is a new networking paradigm focusing on secure data storing, sharing and computing instead of communicating SecNet provide ownership guaranteeing with the help of block chain for better network security.

## **10. FUTURE ENHANCEMENT**

In future work, we will explore how to leverage blockchain for the access authorization on data requests, and design secure and detailed smart contracts for data sharing and computing service in SecNet. In addition, we will model SecNet and analyze its performance through extensive experiments based on advanced platforms (e.g., integrating IPFS [27] and Ethereum [28] to form a SecNet-like architecture).

## 11. BIBLIOGRAPHY

- [1]H. Yin, D. Guo, K. Wang, Z. Jiang, Y. Lyu, and J. Xing, “Hyperconnected network: A decentralized trusted computing and networking paradigm,” *IEEE Netw.*, vol. 32, no. 1, pp. 112–117, Jan./Feb. 2018.
- [2]K. Fan, W. Jiang, H. Li, and Y. Yang, “Lightweight RFID protocol for medical privacy protection in IoT,” *IEEE Trans Ind. Informat.*, vol. 14, no. 4, pp. 1656– 1665, Apr. 2018.
- [3]T. Chajed, J. Gjengset, J. Van Den Hooff, M. F. Kaashoek, J. Mickens,
- [4]R. Morris, and N. Zeldovich, “Amber: Decoupling user data from Web applications,” in *Proc. 15th Workshop Hot Topics Oper. Syst. (HotOS XV)*, Warth-Weiningen, Switzerland, 2015, pp. 1–6.
- [5]M. Lecuyer, R. Spahn, R. Geambasu, T.-K. Huang, and S. Sen, “Enhancing selectivity in big data,” *IEEE Security Privacy*, vol. 16, no. 1, pp. 34–42, Jan./Feb. 2018.
- [6]Y.-A. de Montjoye, E. Shmueli, S. S. Wang, and A. S. Pentland, “openPDS: Protecting the privacy of metadata through SafeAnswers,” *PLoS ONE*, vol. 9, no. 7, 2014, Art. no. e98790.
- [7]C. Perera, R. Ranjan, and L. Wang, “End-to-end privacy for open big data markets,” *IEEE Cloud Computes.*, vol. 2, no. 4, pp. 44–53, Apr. 2015.
- [8]X. Zheng, Z. Cai, and Y. Li, “Data linkage in smart Internet of Things systems: A consideration from a privacy perspective,” *IEEE Communes. Mag.*, vol. 56, no. 9, pp. 55–61, Sep. 2018.