Blockchain Technique for Decentralized Cloud Environments

**ABSTRACT**

Cloud storage is one of the leading options to store massive data, however, the centralized storage approach of cloud computing is not secure. On the other hand, Blockchain is a decentralized cloud storage system that ensures data security. Any computing node connected to the internet can join and form peers network thereby maximizing resource utilization. Blockchain is a distributed peer to peer system where each node in the network stores a copy of blockchain thus making it immutable. In the proposed system, the user’s file is encrypted and stored across multiple peers in the network using the IPFS (InterPlanetary File System) protocol. IPFS creates hash value. The hash value indicates the path of the file and is stored in the blockchain. This paper focuses on decentralized secure data storage, high availability of data, and efficient utilization of storage resources.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| 1. | **CHAPTER 1 : INTRODUCTION**   * 1. GENERAL   2. OBJECTIVE   3. SCOPE OF THE PROJECT   4. EXISTING SYSTEM   1.4.1 EXISTING SYSTEM DISADVANTAGES  1.5 LITERATURE SURVEY  1.6 PROPOSED SYSTEM  1.6.1 PROPOSED SYSTEM ADVANTAGES |  |
| 2. | **CHAPTER 2 : PROJECT DESCRIPTION**  2.1 GENERAL  2.2 PROBLEM DEFINATION  2.3 METHODOLOGIES  2.3.1 MODULES NAME  2.3.2 MODULES EXPLANATION  2.3.3 MODULE DIAGRAM  2.4 GIVEN INPUTAND EXPECTED OUTPUT  2.5 TECHNIQUE OR ALGORITHM |  |
| 3. | **CHAPTER 3 : REQUIREMENTS**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS  3.4 Functional Specification  3.5 Non-Functional SpecificatioN |  |
| 4. | **CHAPTER 4 : SYSTEM DESIGN**  4.1 general  4.2 Uml  4.2.1 USE CASE diagram  4.2.2 CLASS DIAGRAM  4.2.3 OBJECT DIAGRAM  4.2.4 COMPONENT DIAGRAM  4.2.5 DEPLOYMENT DIAGRAM  4.2.6 SEQUENCED DIAGRAM  4.2.7 COLLABARATION DIAGRAM  4.2.8 STATE DIAGRAM  4.2.9 ACTIVITY DIAGRAM  4.3 DATA FLOW DIAGRAM  4..4 ER DIAGRAM  4.5 SYSTEM ARCHETECTURE |  |
| 5. | **CHAPTER 5 :SOFTWARE SPECIFICATION**  5.1 general FRONT END  5.2 features of java  5.2.1 the java framework  5.2.2 objective of java  5.2.3 componets of java frmework  5.2.4 Evolution of collection framework  5.3 features of the common language runtime  5.4 j2ee  5.4.1 EVALUTION OF J2EE  5.4.2 THE J2EE DATA ARCHITECTURE  5.5.3 COMPONENT CLASSES THAT MAKE UP  DATA PROVIDERS |  |
| 6. | **CHAPTER 6 : IMPLEMENTATION**  6.1 GENERAL  6.2 IMPLEMENTATION  6.3 DATA BASE TABLE STRUCTURE |  |
| 7. | **CHAPTER 7 : SNAPSHOTS**  7.1 GENERAL  7.2 VARIOUS SNAPSHOTS |  |
| 8. | **CHAPTER 8 : SOFTWARE TESTING**  8.1 GENERAL  8.2 DEVELOPING METHODOLOGIES  8.3 TYPES OF TESTING |  |
| 9. | **CHAPTER 9**  9.1 FUTURE ENHANCEMENTS |  |
| 10. | **CHAPTER 10 :**  9.1 CONCLUSION  9.2 REFERENCES |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 2.3.2 | Module Diagram |  |
| 4.2.1 | Use case Diagram |  |
| 4.2.2 | Class Diagram |  |
| 4.2.3 | Object Diagram |  |
| 4.2.4 | Component Diagram |  |
| 4.2.5 | Deployment Diagram |  |
| 4.2.6 | Sequence Diagram |  |
| 4.2.7 | Collaboration Diagram |  |
| 4.2.8 | State Diagram |  |
| 4.2.9 | Activity Diagram |  |
| 4.3 | Data Flow Diagram |  |
| 4.4 | E-R Diagram |  |
| 4.5 | System Architecture |  |
| 7.1 | Home Page |  |

**LIST OF SYSMBOLS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **NAME** | **NOTATION** | | **DESCRIPTION** | |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | | Represents a collection of similar entities grouped together. | |
| 2. | Association | name  Class B  Class A  Class A  Class B | | Associations represents static relationships between classes. Roles represents the way the two classes see each other. | |
| 3. | Actor | Class A  Class A  Class B  Class B | | It aggregates several classes into a single classes. | |
| 5. | Aggregation | Interaction between the system and external environment | |
| 5. | Relation  (uses) | | Uses | | Used for additional process communication. | |
| 6. | Relation  (extends) | | extends | | Extends relationship is used when one use case is similar to another use case but does a bit more. | |
| 7. | Communication | |  | | Communication between various use cases. | |
| 8. | State | | State | | State of the process. | |
| 9. | Initial State | |  | | Initial state of the object | |
| 10. | Final state | |  | | Final state of the object | |
| 11. | Control flow | |  | | Represents various control flow between the states. | |
| 12. | Decision box | |  | | Represents decision making process from a constraint | |
| 13. | Usecase | |  | | Interact ion between the system and external environment. | |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which is a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**LIST OF ABBREVATION**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **EXPANSION** |
| 1**.** | DB | DataBase |
| 2. | JVM | Java Virtual Machine |
| 3. | JSP | Java Server Page |
| 4. | PWS | Personalised Web Search |
| 5. | UPS | User Personalised Search |
| 6. | JRE | Java Runtime Environment |

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

As per the Forbes article, 2.5 quintillion bytes of dataare produced each day. Out of the total data in the world over90 percent of data was produced in the last 2 years. With sucha massive increase in the data, cloud storage is required tostore the data. Much of the data currently available through theinternet is quite centralized and is stored with a handful oftechnology companies that have the experience and capital tobuild massive data centers capable of handling this enormousdata. The problem with this approach is the security of data.As this data is stored in a centralized manner, if an attackercan gain access to the server he can easily view and modifythe data. Another problem with this approach is the privacy ofuser data. In many instances, this data is used by third partiesfor data analysis and marketing purposes. Also, the costincurred in storing data in centralized servers is more andmany times users have to pay for the entire plan which theyhave selected even if they have used only a fraction of storageportion thus it does not provide flexibility to the user to payonly for what they are using. Another issue is the scalability ofthe system, it is difficult to scale a centralized storage systemto meet the increasing demand. With zero trust two parties cantransact in Block chain.The proposed system allows two transactionsviz, Taccess and Tdata, Taccess for access control permissionwhich will be set by the respective user who owns the data,and Tdata are used for data storage purposes. The sharedencryption key secures the data from third parties. Cachin,Christian et al discusses the architecture of hyper ledgerblockchain fabric, limitations of electronic coins, working ofhyper ledger fabric, and proof of work consensus algorithm.Hyper ledger Fabric is a permission blockchain network thatallows only limited nodes that have permission to add new

blocks in the blockchain.

In paper the author explains thearchitecture of Ethereum and the working of smart contracts.Bitcoin was used only for sending and receivingcryptocurrency but lacked to add business logic. Ethereumapplications like decentralized file storage and decentralizedautonomous systems are discussed. Ruj, Sushmita proposes BlockStore, a decentralized framework usingblockchain technology to enhance security, transparency intransactions between peers (Host and Renters). The systemuses proof of storage and proof of work to verify that hosts donot meddle with data in Blockchain. The proposed systemdoes not encrypt or decrypts data before uploading it to peerswhich creates a threat to confidentiality and privacy of user’sdata. Juan Benet et alintroduces a new peer to peer filetransfer protocol called IPFS (InterPlanetary File System).IPFS uses a content-based addressing scheme. As per theauthor, IPFS provides a high throughput content-addressedblock storage model along with content-addressed hyperlinks.Li, Dagang, discusses how data sharing in blockchainbasedapplications differs from traditional applications. Theauthor identifies that data-sharing in decentralized architectureis cumbersome. The author proposes Meta-key for secure datasharing in a decentralized storage system based on blockchainalso focuses on the collusion-free property of the proposedcryptographic protocol and proved it strictly. Wohrer et al., explains the solidity used for creating smart contracts inblockchain and its difficulty. All the security issues whichhave been resolved are:

1.Checks-Effects Interaction,

2.Emergency stop,

3.Speed bump,

4. Rate limit,

5. Mutex and

6. Balance limit.

The knowledge related to these issues can found in grey literature and many blog articles. In ,blockchain provides scalability, security, and sustainability, itis also helpful to transform the way of doing business. In thispaper, the author is trying to conduct a comprehensive surveyon the technical and application of blockchain technology bydiscussing its structure to different consensus algorithms. Theauthor has also explained, the structure of blockchain consistsof data, timestamp, and address of the previous block in hashform. The timestamp is recording the time when the block wascreated. A hash function is the one that takes an input of anylength and generates the output with a unique fixed length.Each block contains a hash value of the previous block.therefore, security is increased in Blockchain. It uses proof of

stack (POS), proof of work(POW) consensus algorithm as ameasure to discourage the attacks of Denial of Service andminer can validate transactions in a block depending on the

amount the user holds respectively. Therefore, blockchaintechnology is exceeding recognized and appraised due to itsdecentralized infrastructure and peer-to-peer nature. In paperD. Sivaganesan has suggested the use of blockchain toimprove security and provide transparency in IoT applications.

The author has proposed a smart logistic system for thepharmaceutical sector to keep track of the shipment formedicines. This system combines blockchain and IOT, it alsoincludes a smart contract to bond manufactures, distributes,and the dispenser legally. The use of smart contract avoidsthird person intervention as well as provides improvedsecurity and transparency in the transaction.In the proposed system, smart contracts are used to storefile details in the blockchain and also transfer thecrypto currency from the user’s wallet to the peer’s wallet.AES encryption algorithm for enhancing the security of user’sdata stored in cloud storage. The proposed system maps theuser’s wallet address with the user’s file so that only thelegitimate owner can access the file’s data. Users ' informationis stored in Ethereum blockchain. The Ethereum blockchainnetwork allows the use of smart contracts through whichinformation of file uploaded by the user is stored in theblockchain. The proposed system encrypts and decrypts dataevery single time for upload and download operation. Thesystem uses the IPFS Protocol to distribute files efficientlyacross several peers in the network.

**1.2 OBJECTIVES**

The objective of this project is to provide security to user data storage. In this paper we focus on decentralized storage system which should have high availability of data, and efficient utilization of storage resources. Implemented system uses the AES 256bit encryption algorithm to encrypt the data ensuring the confidentiality of the user’s data. Encrypted data is then distributed and stored across peers in the network using the IPFS protocol.

**1.3 SCOPE OF THE PROJECT**

The scope of this research is to resolve the data privacy and data security concerns of decentralized cloud storage but also provides a medium for the peer to rent their under-utilized storage and thereby, maximizing the storage resource utilization.

**1.4EXISTING SYSTEM:**

* In existing system a massive increase in the data, cloud storage is required to store the data. Much of the data currently available through the internet is quite centralized and is stored with a handful of technology companies that have the experience and capital to build massive data centers capable of handling this enormous data.
* The problem with this approach is the security of data. As this data is stored in a centralized manner, if an attacker can gain access to the server he can easily view and modify the data.
* Another problem with this approach is the privacy of user data.

**1.4.1 Existing System Disadvantages:**

* Less Security and privacy for data.
* Takes more cost for storing data.
* Less resources utilization.

**1.5LITERATURE SURVEY**

**TITLE:**Block Chain Enabled Internet Of Things.

**AUTHOR:**Sivaganesan, D

**YEAR:**2019

**DESCRIPTION:**

The internet of things constitutes for more and advanced opportunities paving way for a competitive advantage in numerous industrious in the prevailing and the upcoming markets. The internet of things and the technologies associated with it are not only creating alterations in the internetwork alone but also with the things that are connected to it. As the internet of things includes the generation and the analysis of the data as the important process, it finds difficulties in administering and protecting the data that are flowing from different levels. This in turn causes difficulties in the secure transmission of the information from its source to the authorized destination and the authorized person at the right time. So the above reasons bring in the capability of the block chain into the IOT, to improve the security as well as implant the transparency. The paper put forward a block chain enabled IOT for the pharmaceutical sector to elude the issue of counterfeit in the medicines.

**TITLE:**Study on Data Security Policy Based On Cloud Storage

**AUTHOR:**Zhe, Diao

**YEAR:**2017

**DESCRIPTION:**

Along with the growing popularisation of Cloud Computing. Cloud storage technology has been paid more and more attention as an emerging network storage technology which is extended and developed by cloud computing concepts. Cloud computing environment depends on user services such as high-speed storage and retrieval provided by cloud computing system. Meanwhile, data security is an important problem to solve urgently for cloud storage technology. In recent years, There are more and more malicious attacks on cloud storage systems, and cloud storage system of data leaking also frequently occurred. Cloud storage security concerns the user's data security. The purpose of this paper is to achieve data security of cloud storage and to formulate corresponding cloud storage security policy. Those were combined with the results of existing academic research by analyzing the security risks of user data in cloud storage and approach a subject of the relevant security technology, which based on the structural characteristics of cloud storage system.

**TITLE:**Data security in cloud computing using AES under HEROKU cloud.

**AUTHOR:**Lee, Bih-Hwang, Ervin Kusuma Dewi, and Muhammad Farid Wajdi

**YEAR:**2018

**DESCRIPTION:**

Cloud security is an evolving sub-domain of computer and network security. Cloud platform utilizes third-party data centers model. An example of cloud platform as a service (PaaS) is Heroku. It supports several programming languages that are used for web application deployment model. Heroku is based on a managed container system, with integrated data services and a powerful ecosystem, for deploying and running modern apps. One essential issue in cloud computing is data security, which is handled using cryptography methods. A possible method to encrypt data is Advanced Encryption Standard (AES). In this paper, we implement Heroku as a cloud platform, then we implement AES for data security in Heroku. The performance evaluation shows that AES cryptography can be used for data security. Moreover, delay calculation of data encryption shows that larger size of data increases the data delay time for encrypting data.

**TITLE:**a Smart contracts: security patterns in the ethereum ecosystem and solidity

**AUTHOR:** Wohrer, Maximilian, and Uwe Zdun

**YEAR:**2018

**DESCRIPTION:**

Smart contracts that build up on blockchain technologies are receiving great attention in new business applications and the scientific community, because they allow untrusted parties to manifest contract terms in program code and thus eliminate the need for a trusted third party. The creation process of writing well performing and secure contracts in Ethereum, which is today's most prominent smart contract platform, is a difficult task. Research on this topic has only recently started in industry and science. Based on an analysis of collected data with Grounded Theory techniques, we have elaborated several common security patterns, which we describe in detail on the basis of Solidity, the dominating programming language for Ethereum. The presented patterns describe solutions to typical security issues and can be applied by Solidity developers to mitigate typical attack scenarios.

**TITLE:**Bitcoin: A peer-to-peer electronic cash system

**AUTHOR:**Nakamoto, Satoshi

**YEAR:**2008

**DESCRIPTION:**

A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

**TITLE:**BlockStore: A Secure Decentralized Storage Framework on Blockchain

**AUTHOR:**Ruj, Sushmita, et al

**YEAR:**2018

**DESCRIPTION:**

In order to ensure faster audits, higher transparency and security, many applications are being designed using blockchains. We propose BlockStore, a secure decentralized storage framework using blockchain technology. The primary motivation is efficient utilization of storage resources of users. Users often have un-utilized or underutilized storage in their devices. They can choose to host their storage resources when they are not in use. Users rent storage from the host for a fee for a fixed period of time and release back after the time expires. BlockStore keeps track of un-utilized storage of hosts in Space Wallet, a structure that helps in assigning storage to renters on request. The ownership of storage can be proved by logging all storage transactions in a public ledger (the blockchain), which can be verified by any user. A host cannot host the same storage to two users at the same time, nor can it tamper with the data of the renter. Renters cannot frame a host of cheating. BlockStore uses proofs of storage and data possession to verify that the hosts do not tamper with data and penalizes parties for misbehavior. Users can encrypt data for privacy. Payment and penalty are handled using smart contracts. BlockStore differs from existing solutions, by providing stronger audit that detects and penalizes misbehaving parties earlier than existing schemes.

* 1. **PROPOSED SYSTEM**
* In this paper the use of encryption technique outlines the importance of data security and privacy protection.
* Discusses the increasing demand for cloud storage with associated security and privacy issues in centralized cloud storage. As per the discussion by encrypting the data and scattering the data across multiple nodes, a high level of data security can be achieved.
* Authors have used the AES encryption algorithm to enhance security with speed without impacting the system’s performance.
* The peer-to-peer network uses proof-of-work to record a public history of transactions.

**1.6.1 PROPOSED SYSTEM ADVANTAGE**

* High Security and privacy for data.
* Takes less cost for storing data.
* More resources utilization.

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

Data privacy and security are concerns when data resides thirdparty storage. Storage can be created from the underutilizedresources of peers. Data security, privacy, availability, andresource utilization are the areas handled by the proposedsystem.

**2.2 METHODOLOGIES**

**2.2.1MODULES NAME:**

1. User Interface Design

2. Cloud Storage

3. Data Owner

4. Data User

**1. User Interface Design**

In this module we design the windows for the project. These windows are used for secure login for all users. To connect with server user must give their username and password then only they can able to connect the server. If the user already exits directly can login into the server else user must register their details such as username, password and Email id, into the server. Server will create the account for the entire user to maintain upload and download rate. Name will be set as user id. Logging in is usually used to enter a specific page.

User Login

Server

Database

Home Page

Register &Login Page

**2. Cloud Storage**

This is the first module of this project and he will be the Cloud storage and has control over all the things. In this module Cloud storage can login.Cloud storage can trace the data from the user.Cloud storage will have have block data information.In cloud storage system will also have a data owner information in the database.Cloud storage system will also maintain the data user information from the database.

Data Owner Info

Block Data Info

Trace Data

Cloud storage

Data Base

Data User Info

**3. Data Owner**

This is the second module of this project. In this module data owner should be register then login. Data owner will upload the files.Data owner will be see the files.Data owner should view all the requests from the users.Data owner should send the key to the user.

View File

Data Owner

Data Base

View Requests

Upload File

Send Key

**4. Data User**

This is the third module of this project. In this module user need to register and then login. Then data user can search the files based on the file name. Data user can be see the responses from the user in database.

View Response

Search File

Data User

Database

**2.3 GIVEN INPUT EXPECTED OUTPUT:**

* **User Interface Design**

Input : Enter Login name and Password

Output : If valid user name and password then directly open the home page otherwise show error message and redirect to the registration page.

* **Cloud Storage**

Input : Cloud storage Login name and Password

Output: If valid user name and password then directly open the cloud storagehome page otherwise show error message and redirect to the cloud storage login page.

* **Data Owner**

Input : Enter the login name and password

Output : If valid user name and password then directly open the data owner home page otherwise show error message and redirect to the data owner login page.

* **Data User**

Input : Enter the user name and password

Output: If valid user name and password then directly open the data user home page otherwise show error message and redirect to the data user login page.

**2.4 TECHNIQUE USED OR ALGORITHM USED**

**2.4.1 EXISTING ALGORITHM**

* **Centralized Storage Mechanism**

In many instances, this data is used by third parties for data analysis and marketing purposes. Also, the cost incurred in storing data in centralized servers is more and many times users have to pay for the entire plan which they have selected even if they have used only a fraction of storage portion thus it does not provide flexibility to the user to pay only for what they are using. Another issue is the scalability of the system, it is difficult to scale a centralized storage system to meet the increasing demand.

**2.4.2 PROPOSED ALGORITHM**

* **AES Algorithm**

The Advanced Encryption Standard (AES) is a symmetric [block cipher](https://searchsecurity.techtarget.com/definition/block-cipher). AES is implemented in software and hardware throughout the world to [encrypt](https://searchsecurity.techtarget.com/definition/encryption) sensitive data. It is essential for government computer security, cybersecurity and electronic data protection.AES-128 uses a 128-[bit](https://whatis.techtarget.com/definition/bit-binary-digit) key length to encrypt and decrypt a block of messages, while AES-192 uses a 192-bit key length and AES-256 a 256-bit key length to encrypt and decrypt messages. Each cipher encrypts and decrypts data in blocks of 128 bits using [cryptographic](https://searchsecurity.techtarget.com/definition/cryptography) keys of 128, 192 and 256 bits, respectively. Ciphers use the same key for encrypting and decrypting.

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

We have conducted experiments on our collected dataset and extensive results have demonstrated that our model outperforms all other existing models. In the future, we will investigate more tasks under this framework, such as event summarization and event attribute mining in social media.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shouls what the system do and not how it should be implemented.

* PROCESSOR : DUAL CORE 2 DUOS.
* RAM : 2GB DD RAM
* HARD DISK : 250 GB

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

**SOFTWARE REQUIREMENTS**

* FRONT END : J2EE (JSP, SERVLET)
* BACK END : MY SQL 5.5
* OPERATING SYSTEM : WINDOWS 7
* IDE : ECLIPSE

**3.4 FUNCTIONAL REQUIREMENTS**

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior,Firstly, the system is the first that achieves the standardnotion of semantic security for data confidentialityin attribute-based deduplication systems byresorting to the hybrid cloud architecture.

**3.5 NON-FUNCTIONAL REQUIREMENTS**

**EFFICIENCY**

Our multi-modal event tracking and evolution frameworkis suitable for multimedia documents from various socialmedia platforms, which can not only effectively capturetheir multi-modal topics, but also obtain the evolutionarytrends of social events and generate effective event summarydetails over time. Our proposed mmETM model can exploit the multi-modalproperty of social event, which can effectively model socialmedia documents including long text with related imagesand learn the correlations between textual and visualmodalities to separate the visual-representative topics andnon-visual-representative topics.

**CHAPTER 4**

**DESIGN ENGINEERING**

**4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

**4.2 UML DIAGRAMS**

**4.2.1 USE CASE DIAGRAM**



**EXPLANATION:**

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

**4.2.2 CLASS DIAGRAM**

****

**EXPLANATION**

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

**4.2.3 OBJECT DIAGRAM**



**EXPLANATION:**

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

**4.2.4 COMPONENT DIAGRAM**



**EXPLANATION**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicates dependencies.

**4.2.5 DEPLOYMENT DIAGRAM**



**EXPLANATION:**

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.

**4.2.6 SEQUENCE DIAGRAM**



**EXPLANATION:**

Sequence diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, Sequence diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An sequence diagram shows the overall flow of control.

**4.2.7 COLLABORATION DIAGRAM**



**EXPLANATION:**

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

**4.2.8 STATE DIAGRAM**



**EXPLANATION:**

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

**4.2.9 ACTIVITY DIAGRAM**



**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**4.3 DATA FLOW DIAGRAM**

**Level 0**

Register

Home Page

Login

Verify Details

Data base

Error page

**Level 1**

View Response

Search File

View File

Trace Data

Data base

Upload File

Data Owner Info

Block Data Info

Send keys

View Requests

Data User Info

**EXPLANATION:**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

**4.4 E-R DIAGRAM**

View Response

Block Data Info

Data base

Send Keys

Data User Info

Data OwnerInfo

View Requests

Search File

Cloud Storage

Data Owner

Data User

Verify

Details

Trace Data

View File

Upload File

**EXPLANATION:**

Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database.

**4.5 SYSTEM ARCHITECTURE**

Cloud Storage

Data Owner

Upload File

View File

View Requests

Send Key

Data User

Search File

View Response

Trace Data

Block Data Info

Data Owner Info

Data User Info

Data

Base

**EXPLANATION**

In this project cloud storage has a login the page.Cloud storage has a trace a data in the database.Cloud storage has also block data information for the user in the cloud computing.Cloud storage has also a data owner information from the database.Cloud storage has also a data user information from the cloud computingData owner has also register all the details and login the page.Data owner has a upload a file in the database.Data owner has also a view file. Data owner can view all the requests from the database. Data owner has a send key to the cloud computing. Data user can also a register a file after they can login in the database. Data user can have a search a file. Data user can also have a view response from the database in the cloud computing.

**CHAPTER 5**

**DEVELOPMENT TOOLS**

* 1. **GENERAL**

This chapter is about the software language and the tools used in the development of the project. The platform used here is JAVA. The Primary languages are JAVA,J2EE and J2ME. In this project J2EE is chosen for implementation.

**5.2 FEATURES OF JAVA**

**5.2.1 THE JAVA FRAMEWORK**

Java is a programming language originally developed by James Gosling at Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere".

Java is considered by many as one of the most influential programming languages of the 20th century, and is widely used from application software to web applications the java framework is a new platform independent that simplifies application development internet. Java technology's versatility, efficiency, platform portability, and security make it the ideal technology for network computing. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

**5.2.2 OBJECTIVES OF JAVA**

To see places of Java in Action in our daily life, explore java.com.

## WHY SOFTWARE DEVELOPERS CHOOSE JAVA

Java has been tested, refined, extended, and proven by a dedicated community. And numbering more than 6.5 million developers, it's the largest and most active on the planet. With its versatility, efficiency, and portability, Java has become invaluable to developers by enabling them to:

* Write software on one platform and run it on virtually any other platform
* Create programs to run within a Web browser and Web services
* Develop server-side applications for online forums, stores, polls, HTML forms processing, and more
* Combine applications or services using the Java language to create highly customized applications or services
* Write powerful and efficient applications for mobile phones, remote processors, low-cost consumer products, and practically any other device with a digital heartbeat

## SOME WAYS SOFTWARE DEVELOPERS LEARN JAVA

Today, many colleges and universities offer courses in programming for the Java platform. In addition, developers can also enhance their Java programming skills by reading Sun's java.sun.com Web site, subscribing to Java technology-focused newsletters, using the Java Tutorial and the New to Java Programming Center, and signing up for Web, virtual, or instructor-led courses.

**OBJECTORIENTED**

To be an Object Oriented language, any language must follow at least the four characteristics.

1. Inheritance   :It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse  the existing code and adding addition a features as needed.

2. Encapsulation: It is the mechanism of combining the information and providing the abstraction.

3. Polymorphism: As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the  methods.

4. Dynamic binding: Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

**5.2.3 JAVA SWING OVERVIEW**

**ABSTRACT WINDOW TOOLKIT (AWT) IS CROSS-PLATFORM**

Swing provides many controls and widgets to build user interfaces with. Swing class names typically begin with a J such as JButton, JList, JFrame. This is mainly to differentiate them from their AWT counterparts and in general is one-to-one replacements. Swing is built on the concept of Lightweight components vs AWT and SWT's concept of Heavyweight components. The difference between the two is that the Lightweight components are rendered (drawn) using purely Java code, such as drawLine and drawImage, whereas Heavyweight components use the native operating system to render the components.Some components in Swing are actually heavyweight components. The top-level classes and any derived from them are heavyweight as they extend the AWT versions. This is needed because at the root of the UI, the parent windows need to be provided by the OS. These top-level classes include JWindow, JFrame, JDialog and  JApplet. All Swing components to be rendered to the screen must be able to trace their way to a root window of one of those classes.

**NOTE**: It generally it is not a good idea to mix heavyweight components with lightweight components (other than as previously mentioned) as you will encounter layering issues, e.g., a lightweight component that should appear "on top" ends up being obscured by a heavyweight component. The few exceptions to this include using heavyweight components as the root pane and for popup windows. Generally speaking, heavyweight components will render on top of lightweight components and will not be consistent with the look and feel being used in Swing. There are exceptions, but that is an advanced topic. The truly adventurous may want to consider reading this [article](http://java.sun.com/products/jfc/tsc/articles/mixing/) from Sun on mixing heavyweight and lightweight components.

**5.2.4 EVOLUTION OF COLLECTION FRAMEWORK:**

Almost all collections in Java are derived from the [java.util.Collection](http://download.oracle.com/javase/7/docs/api/java/util/Collection.html) interface. Collection defines the basic parts of all collections. The interface states the add() and remove() methods for adding to and removing from a collection respectively. Also required is the toArray() method, which converts the collection into a simple array of all the elements in the collection. Finally, the contains() method checks if a specified element is in the collection. The Collection interface is a subinterface of [java.util.Iterable](http://download.oracle.com/javase/7/docs/api/java/util/Iterable.html), so the iterator() method is also provided. All collections have an iterator that goes through all of the elements in the collection. Additionally, Collection is a generic. Any collection can be written to store any class. For example, Collection<String> can hold strings, and the elements from the collection can be used as strings without any casting required.

There are three main types of collections:

* Lists: always ordered, may contain duplicates and can be handled the same way as usual arrays
* Sets: cannot contain duplicates and provide random access to their elements
* Maps: connect unique keys with values, provide random access to its keys and may host duplicate values

**LIST**

Lists are implemented in the JCF via the java.util.List interface. It defines a list as essentially a more flexible version of an array. Elements have a specific order, and duplicate elements are allowed. Elements can be placed in a specific position. They can also be searched for within the list. Two concrete classes implement List. The first is java.util.ArrayList, which implements the list as an array. Whenever functions specific to a list are required, the class moves the elements around within the array in order to do it. The other implementation is java.util.LinkedList. This class stores the elements in nodes that each have a pointer to the previous and next nodes in the list. The list can be traversed by following the pointers, and elements can be added or removed simply by changing the pointers around to place the node in its proper place.

**SET:**

Java's [java.util.Set](http://download.oracle.com/javase/7/docs/api/java/util/Set.html) interface defines the set. A set can't have any duplicate elements in it. Additionally, the set has no set order. As such, elements can't be found by index. Set is implemented by java.util.HashSet,java.util.LinkedHashSet, and java.util.TreeSet. HashSet uses a hash table. More specifically, it uses a [java.util.HashMap](http://download.oracle.com/javase/7/docs/api/java/util/HashMap.html) to store the hashes and elements and to prevent duplicates. Java.util.LinkedHashSet extends this by creating a doubly linked list that links all of the elements by their insertion order. This ensures that the iteration order over the set is predictable. [java.util.TreeSet](http://download.oracle.com/javase/7/docs/api/java/util/TreeSet.html) uses a red-black tree implemented by a [java.util.TreeMap](http://download.oracle.com/javase/7/docs/api/java/util/TreeMap.html). The red-black tree makes sure that there are no duplicates. Additionally, it allows Tree Set to implement java.util.SortedSet.

The [java.util.Set](http://download.oracle.com/javase/7/docs/api/java/util/Set.html) interface is extended by the java.util.SortedSet interface. Unlike a regular set, the elements in a sorted set are sorted, either by the element's compareTo() method, or a method provided to the constructor of the sorted set. The first and last elements of the sorted set can be retrieved, and subsets can be created via minimum and maximum values, as well as beginning or ending at the beginning or ending of the sorted set. The SortedSet interface is implemented by java.util.TreeSet

[java.util.SortedSet](http://download.oracle.com/javase/7/docs/api/java/util/SortedSet.html) is extended further via the java.util.NavigableSet interface. It's similar to SortedSet, but there are a few additional methods. The floor(), ceiling(), lower(), and higher() methods find an element in the set that's close to the parameter. Additionally, a descending iterator over the items in the set is provided. As with SortedSet, java.util.TreeSet implements NavigableSet.

**MAP:**

Maps are defined by the java.util.Map interface in Java. Maps are simple data structures that associate a key with a value. The element is the value. This lets the map be very flexible. If the key is the hash code of the element, the map is essentially a set. If it's just an increasing number, it becomes a list. Maps are implemented by java.util.HashMap, java.util.LinkedHashMap, and java.util.TreeMap. HashMap uses a hash table. The hashes of the keys are used to find the values in various buckets. LinkedHashMap extends this by creating a doubly linked list between the elements. This allows the elements to be accessed in the order in which they were inserted into the map. TreeMap, in contrast to HashMap and LinkedHashMap, uses a red-black tree. The keys are used as the values for the nodes in the tree, and the nodes point to the values in the map

**THREAD:**

Simply put, a threadis a program's path of execution. Most programs written today run as a single thread, causing problems when multiple events or actions need to occur at the same time. Let's say, for example, a program is not capable of drawing pictures while reading keystrokes. The program must give its full attention to the keyboard input lacking the ability to handle more than one event at a time. The ideal solution to this problem is the seamless execution of two or more sections of a program at the same time.

## CREATING THREADS

Java's creators have graciously designed two ways of creating threads: implementing an interface and extending a class. Extending a class is the way Java inherits methods and variables from a parent class. In this case, one can only extend or inherit from a single parent class. This limitation within Java can be overcome by implementing interfaces, which is the most common way to create threads. (Note that the act of inheriting merely allows the class to be run as a thread. It is up to the class to start() execution, etc.)

Interfaces provide a way for programmers to lay the groundwork of a class. They are used to design the requirements for a set of classes to implement. The interface sets everything up, and the class or classes that implement the interface do all the work. The different set of classes that implement the interface have to follow the same rules.

**5.5 CONCLUSION**

Swing's high level of flexibility is reflected in its inherent ability to override the native host [operating system](http://en.wikipedia.org/wiki/Operating_system) (OS)'s GUI controls for displaying itself. Swing "paints" its controls using the Java 2D APIs, rather than calling a native user interface toolkit. The Java thread scheduler is very simple. All threads have a priority value which can be changed dynamically by calls to the threads setPriority() method . Implementing the above concepts in our project to do the efficient work among the Server.

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 GENERAL**

**1. Block.java**

package com.dao;

import java.util.Date;

public class Block {

public String hash;

public String previousHash;

public String data; //our data will be a simple message.

private long timeStamp; //as number of milliseconds since 1/1/1970.

private int nonce;

//Block Constructor.

public Block(String data,String previousHash ) {

this.data = data;

this.previousHash = previousHash;

this.timeStamp = new Date().getTime();

this.hash = calculateHash(); //Making sure we do this after we set the other values.

}

//Calculate new hash based on blocks contents

public String calculateHash() {

String calculatedhash = StringUtil.*applySha256*(

previousHash +

Long.*toString*(timeStamp) +

Integer.*toString*(nonce) +

data

);

return calculatedhash;

}

public void mineBlock(int difficulty) {

String target = new String(new char[difficulty]).replace('\0', '0'); //Create a string with difficulty \* "0"

while(!hash.substring( 0, difficulty).equals(target)) {

nonce ++;

hash = calculateHash();

}

System.*out*.println("Block Mined!!! : " + hash);

}

public String toString()

{

return data;

}

}

**2. FileUpload.java**

package com.servlets;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.IOException;

import java.io.InputStream;

import java.io.PrintWriter;

import java.security.Key;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.util.Date;

import javax.crypto.KeyGenerator;

import javax.servlet.ServletException;

import javax.servlet.annotation.MultipartConfig;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.Part;

import com.dao.DBConnection;

import com.dao.ImageEncrypt;

import com.dao.NoobChain;

import com.dao.RandomeString;

/\*\*

\* Servlet implementation class FileUpload

\*/

@MultipartConfig(maxFileSize = 16177215)

@WebServlet("/FileUpload")

public class FileUpload extends HttpServlet {

private static final long serialVersionUID = 1L;

private static final int BUFFER\_SIZE = 4096;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public FileUpload() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

Connection conn = null;

InputStream inputStream = null;

String filename;

PrintWriter out=response.getWriter();

File file = null;

String id=RandomeString.getID();

Part filePart = request.getPart("file");

String ctype=filePart.getContentType();

String email=(String)request.getSession().getAttribute("email");

filename=request.getParameter("filename");

String content=request.getParameter("content");

if (filePart != null)

{

System.out.println(filePart.getName());

System.out.println(filePart.getSize());

System.out.println(filePart.getContentType());

try {

System.out.println(email);

inputStream = filePart.getInputStream();

ByteArrayOutputStream bs = new ByteArrayOutputStream();

int bytesRead = -1;

byte[] buffer = new byte[BUFFER\_SIZE];

while ((bytesRead = inputStream.read(buffer)) != -1)

{

bs.write(buffer, 0, bytesRead);

}

KeyGenerator keyGenerator;

keyGenerator = KeyGenerator.getInstance("AES");

keyGenerator.init(128);

Key key = keyGenerator.generateKey();

System.out.println(key);

byte[] keybit=key.getEncoded();

byte[] encrypted = ImageEncrypt.encryptPdfFile(key, bs.toByteArray() );

System.out.println(encrypted);

String k=key.getEncoded().toString();

String hash=NoobChain.getBlock(content);

conn=DBConnection.connect();

String sql="insert into upload values(?,?,?,?,?,?,?,?)";

PreparedStatement statement = conn.prepareStatement(sql);

statement.setString(1, id);

statement.setString(2, hash);

statement.setString(3, email);

statement.setString(4, filename);

statement.setString(5, ctype);

statement.setBytes(6, encrypted);

statement.setString(7, k);

statement.setBytes(8,keybit);

int row = statement.executeUpdate();

if (row > 0)

{

DBConnection.addActivity((String)request.getSession().getAttribute("email"), "Uploaded file "+filename+" successfully", new Date().toLocaleString());

//DBConnection.addActivity(cspid, email);

out.println("<script type=\"text/javascript\">");

out.println("alert('Uploaded Successfully');");

out.println("window.location='UploadFile.jsp'</script>");

}else{

out.println("<script type=\"text/javascript\">");

out.println("alert('File Uploading Failed');");

out.println("window.location='UploadFile.jsp'</script>");

}

} catch (SQLException ex)

{

ex.printStackTrace();

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

} finally {

if (conn != null) {

try {

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

**3. SendKey.java**

package com.servlets;

import java.io.IOException;

import java.sql.Connection;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Date;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.dao.DBConnection;

/\*\*

\* Servlet implementation class SendKey

\*/

@WebServlet("/SendKey")

public class SendKey extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public SendKey() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

String fid=request.getParameter("fid");

String to=request.getParameter("to");

String from=(String)request.getSession().getAttribute("email");

String key=DBConnection.getkey(fid);

Connection con=DBConnection.connect();

try {

Statement st=con.createStatement();

int i=st.executeUpdate("insert into ukeys values(0,'"+from+"','"+to+"','"+fid+"','"+key+"')");

DBConnection.addActivity((String)request.getSession().getAttribute("email"), "Sended key responce to "+to+" successfully", new Date().toLocaleString());

response.sendRedirect("SendKey.jsp");

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

}

}

**4.ViewData.java**

package com.servlets;

import java.io.IOException;

import java.io.OutputStream;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Date;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import com.dao.DBConnection;

import com.dao.ImageEncrypt;

/\*\*

\* Servlet implementation class ViewData

\*/

@WebServlet("/ViewData")

public class ViewData extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public ViewData() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

String iid=request.getParameter("fid");

String file\_key=request.getParameter("key");

byte[] image = null;

Connection con = null;

byte[] imgData = null;

byte[] key=null;

String ctype="";

Statement stmt = null;

String filename="";

ResultSet rs = null;

try {

con = DBConnection.connect();

stmt = con.createStatement();

rs = stmt.executeQuery("select \* from upload where fid="+iid+" and filekey='"+file\_key+"'");

System.out.println("select \* from upload where fid="+iid+" and filekey='"+file\_key+"'");

if (rs.next()) {

System.out.println("ok");

image = rs.getBytes("data");

key=rs.getBytes("en\_key");

filename=rs.getString("filename");

ctype=rs.getString("contenttype");

DBConnection.addActivity((String)request.getSession().getAttribute("email"), "Downloaded file "+filename+" successfully", new Date().toLocaleString());

//imgData = image.getBytes(1, (int) image.length());

} else {

}

// display the image

response.setContentType(ctype);

//response.setHeader("Content-disposition", "attachment; filename="+ filename);

SecretKey key2 = new SecretKeySpec(key, 0, key.length, "AES");

imgData=ImageEncrypt.decryptPdfFile(key2, image);

System.out.println(key2);

OutputStream o = response.getOutputStream();

o.write(imgData);

o.flush();

o.close();

} catch (Exception e) {

e.printStackTrace();

/\* out.println("Unable To Display image");

out.println("Image Display Error=" + e.getMessage());

\*/return;

}

try {

rs.close();

stmt.close();

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

finally{

}

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

}}

**5. Owner.java:**

package com.servlets;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import com.dao.DBConnection;

/\*\*

\* Servlet implementation class Owner

\*/

@WebServlet("/Owner")

public class Owner extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public Owner() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

PrintWriter o = response.getWriter();

String uid = request.getParameter("email");

String pwd = request.getParameter("password");

String sql = "select \* from owner where email='"+uid+"' and password='"+pwd+"'";

boolean b = DBConnection.getData(sql);

HttpSession session = request.getSession();

if(b == true){

session.setAttribute("email", uid);

sql = "select name from owner where email='"+uid+"'";

String name = DBConnection.getName(sql);

session.setAttribute("name", name);

response.sendRedirect("DataOwnerHome.jsp");

}else{

o.println("<script type=\"text/javascript\">");

o.println("alert('Please enter valid Details');");

o.println("window.location='DataOwnerLogin.jsp';</script>");

}

}

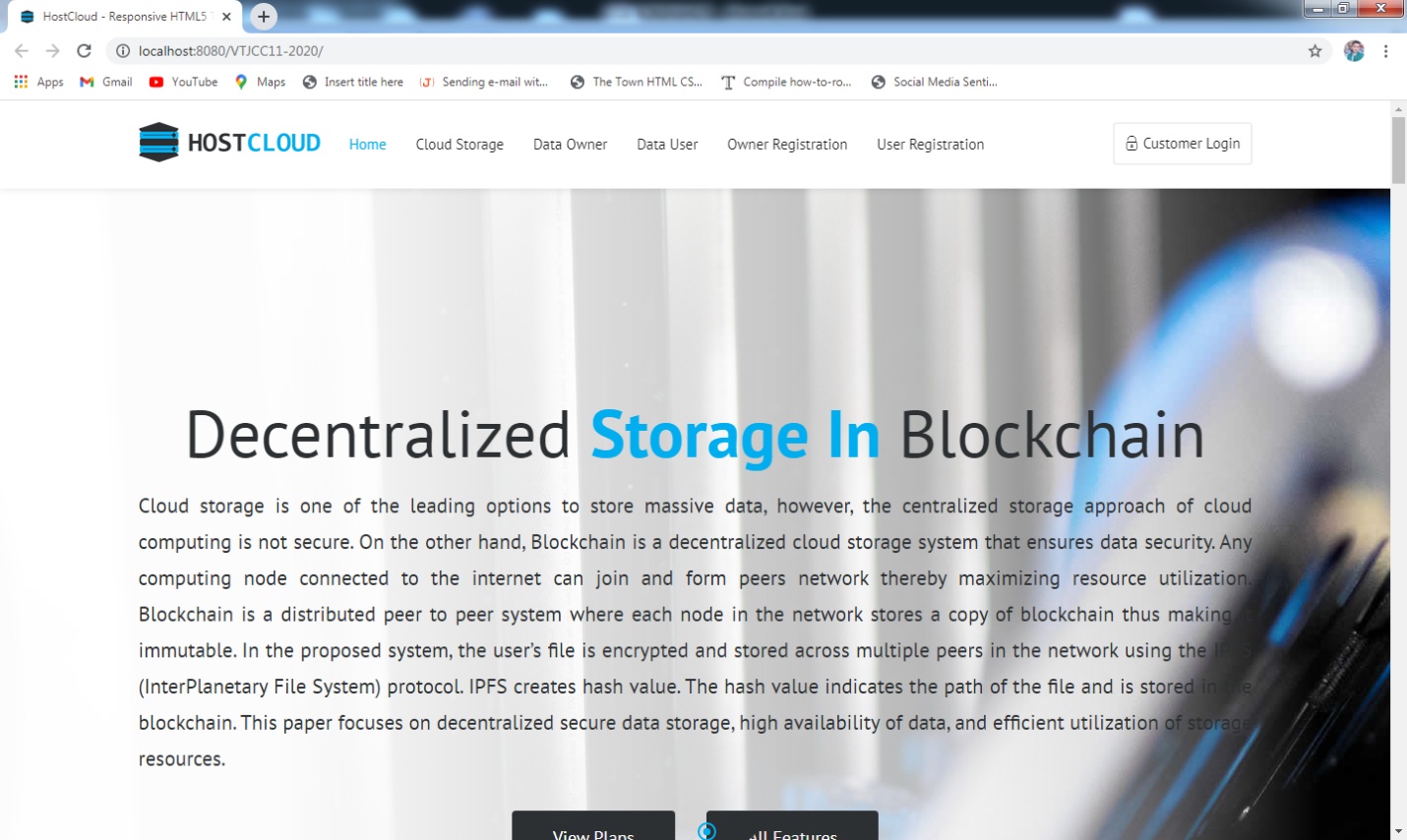
}

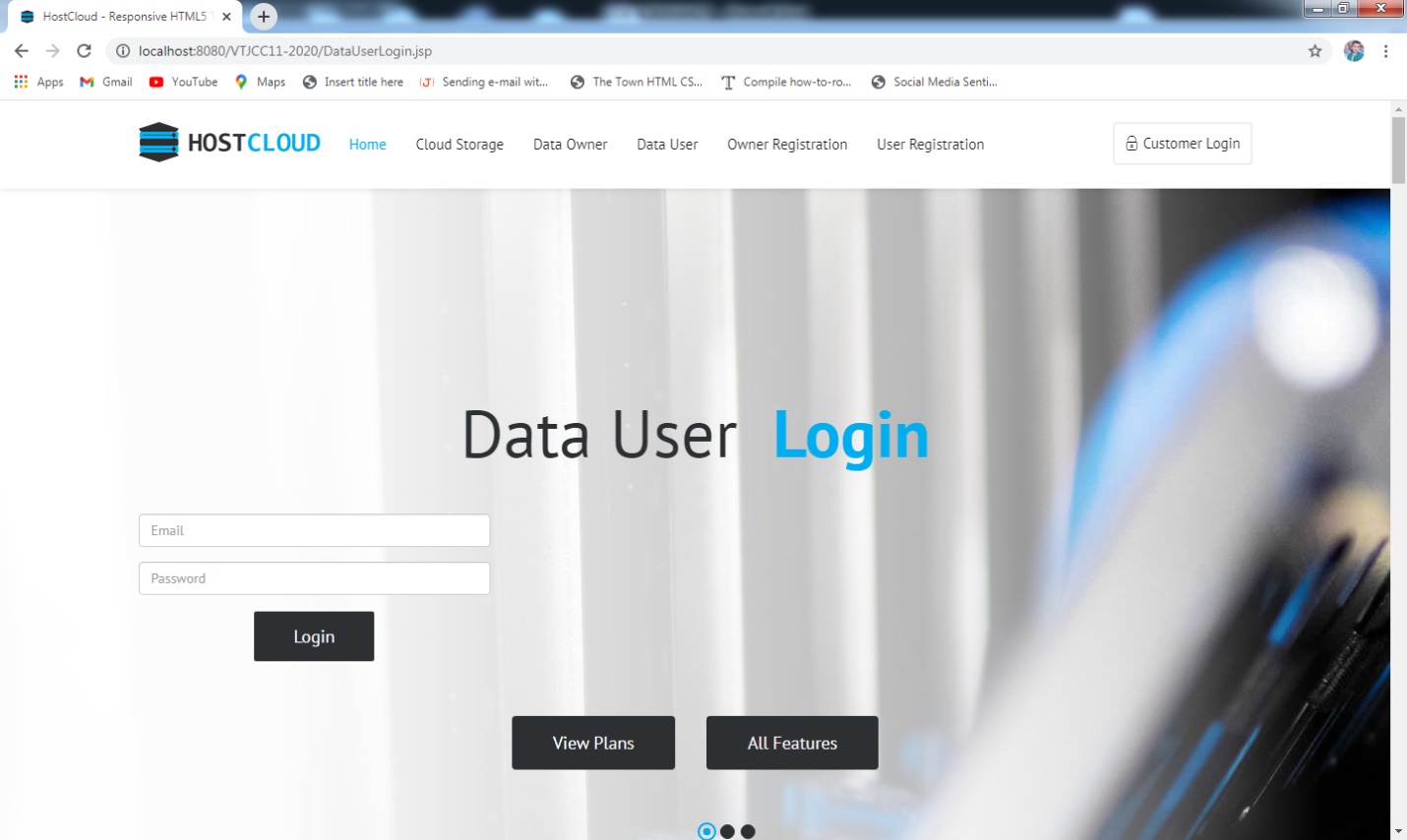
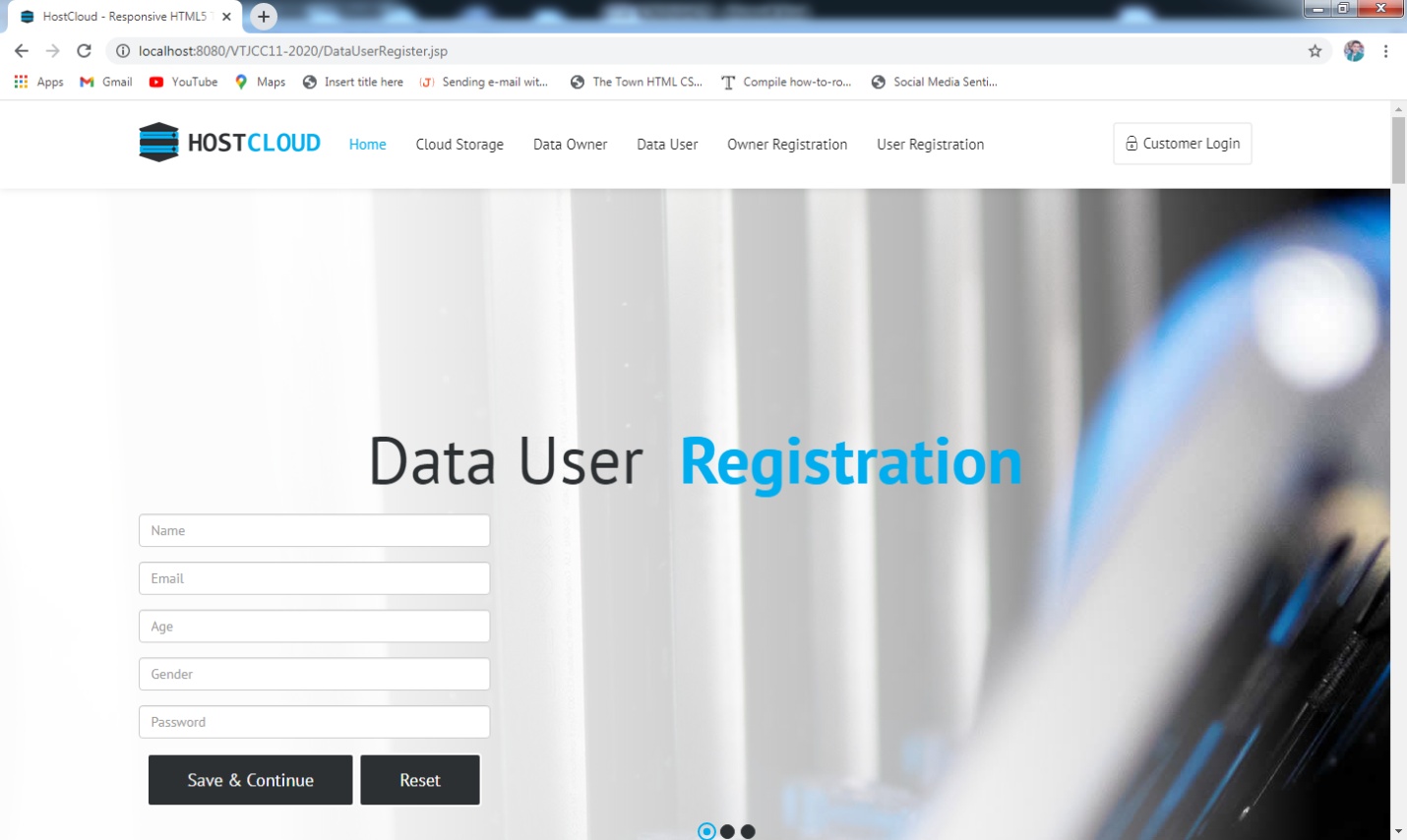
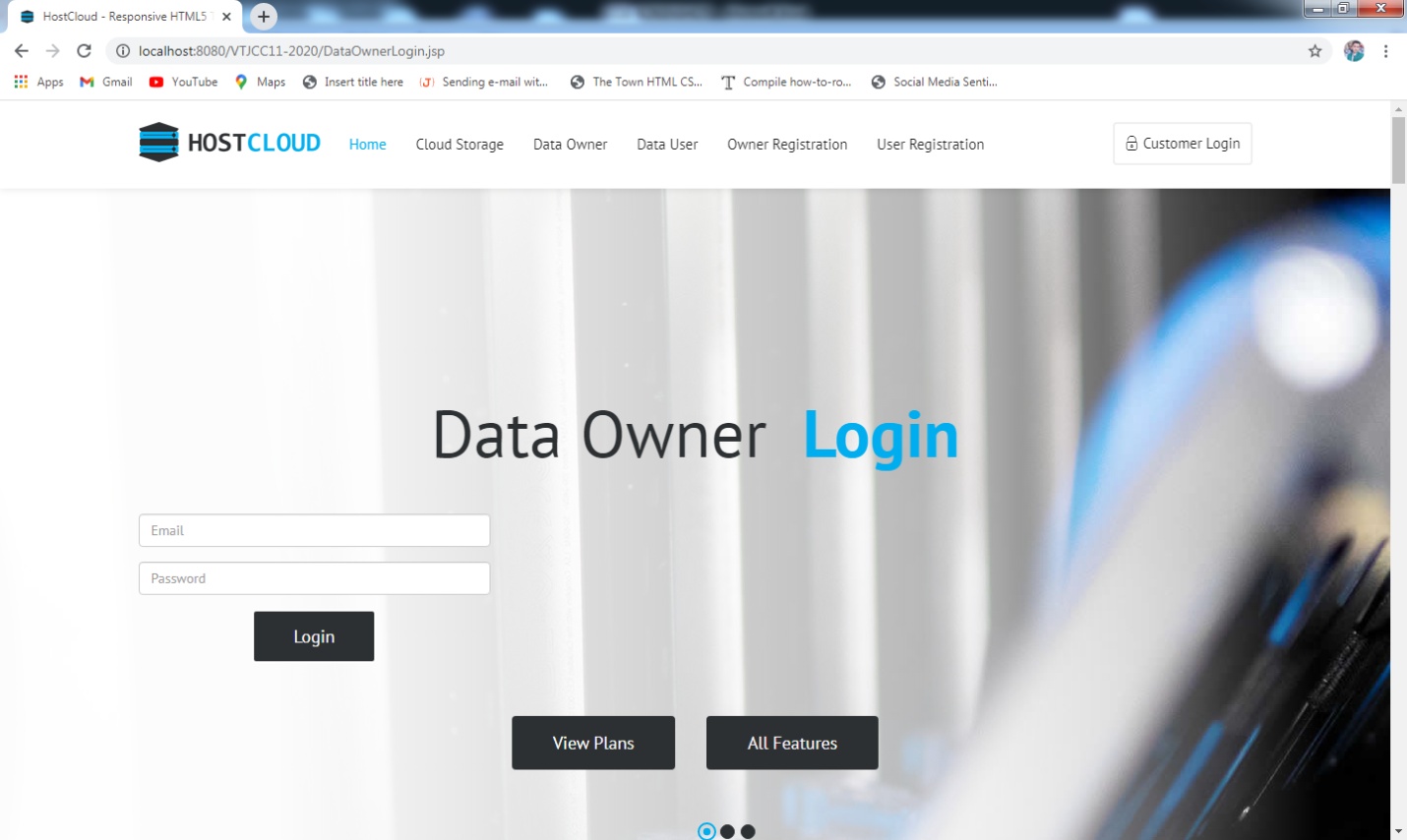
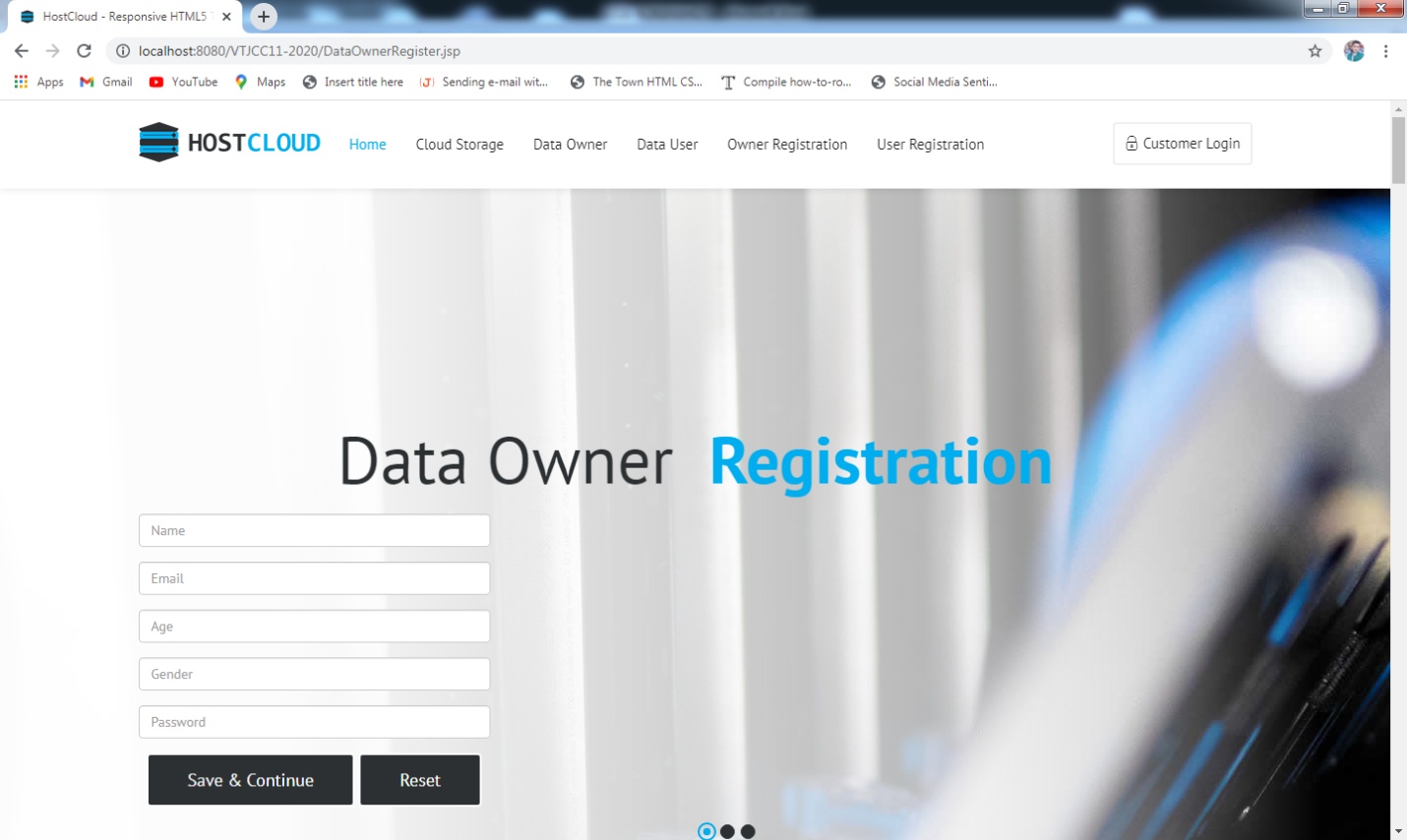
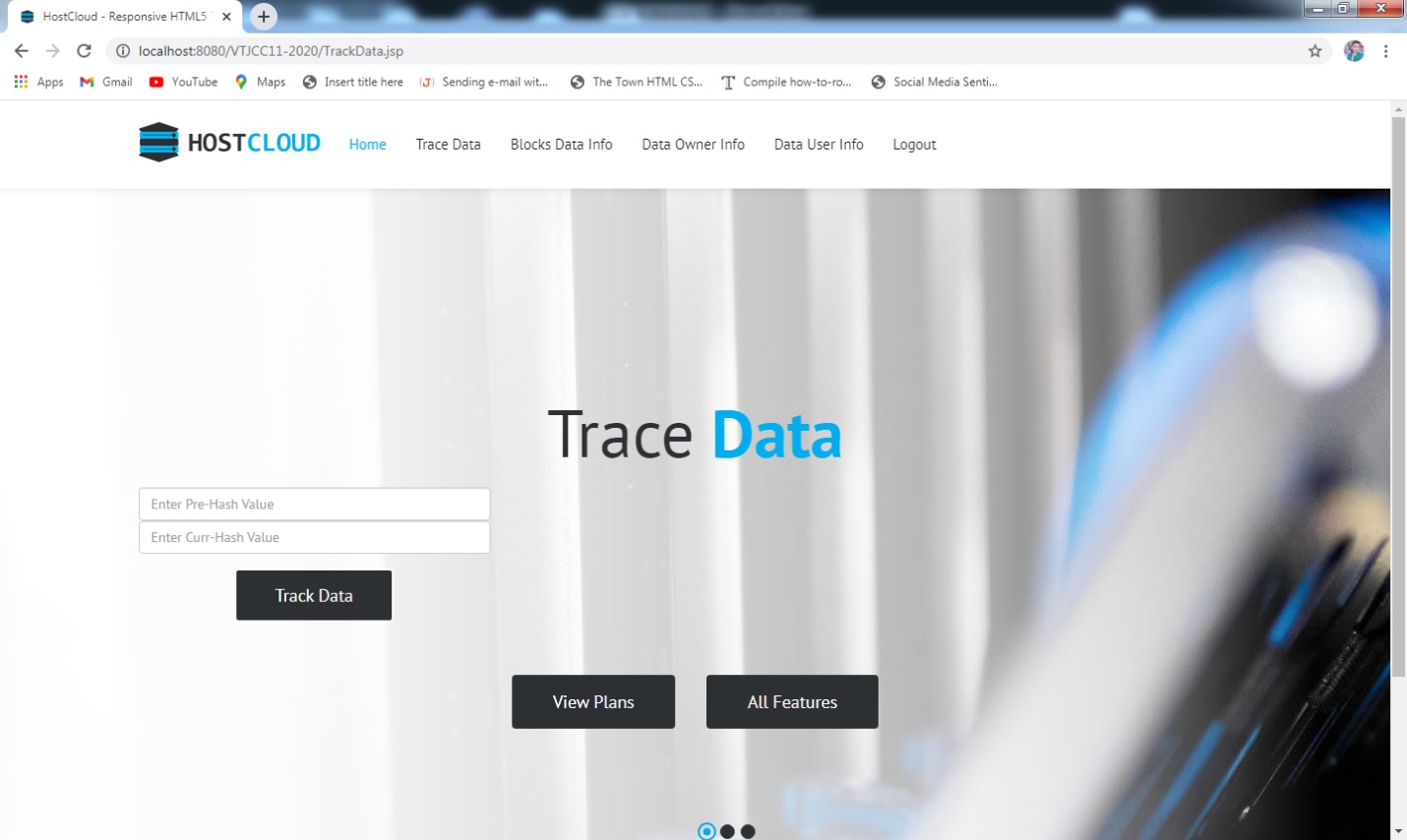
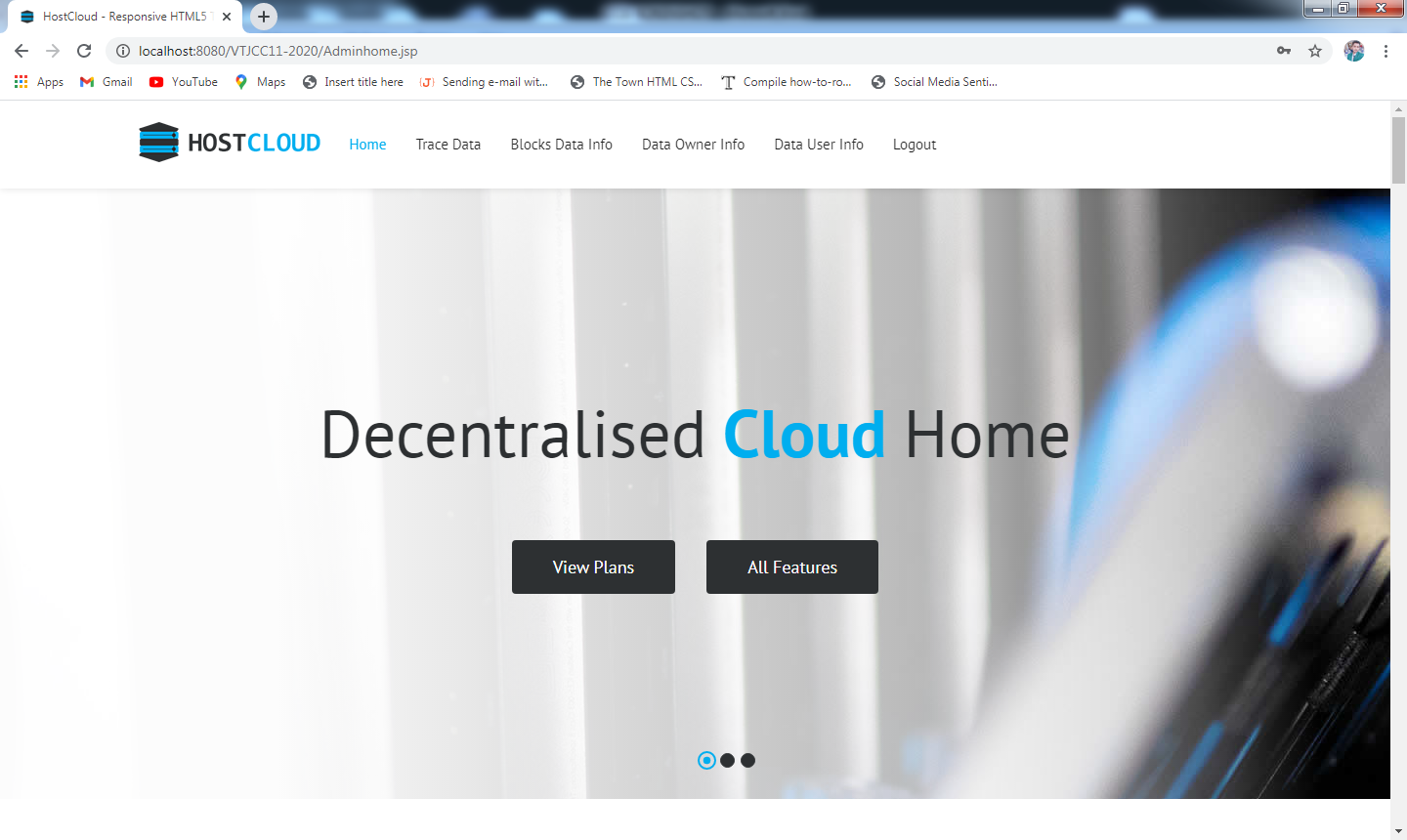
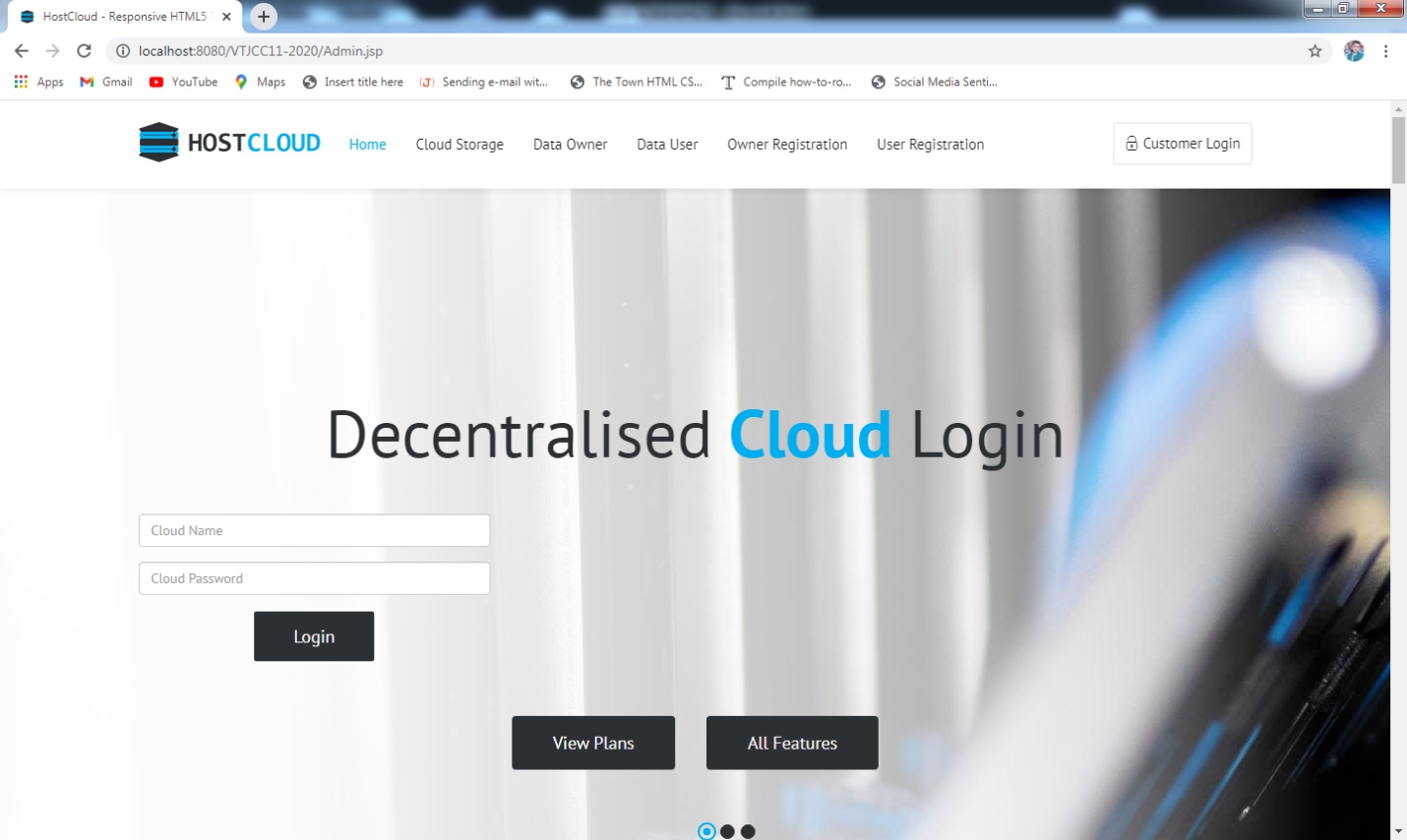
**CHAPTER 7**

**SNAPSHOTS**

**GENERAL:**

This project is implements like web application using COREJAVA and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

**SNAPSHOTS**

****

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 GENERAL**

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.

**8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3TYPES OF TESTS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl. No** | **Test scenario** | **User action** | **Expected result** | **Actual Result** | **Remarks** |
| 1. | Registration | Users registering into the system. | Register into the system. | Successfully alert registered message. | Pass |
| 2. | Login | 1. Entered correct password.  2. Three failed attempts. | 1. Log into the system.  2. Alert generated. | 1. Successfully logged in.  2. Successfully generated the alert. | Pass |
| 3. | Cloud Storage | Trace a data,Block data info,Data owner info,Data user info | Massages sending cloud storage alert is generated. | Successfully generated the alert and massages sending | Successful |
| 4. | Data Owner | Upload a file, View File, View Requests and Send Keys | Data Owner has to actions | Successfully generated the alert to data owner message | Successful |
| 4. | Data User | Search File and View Responses | Massages Alert is generated | Successfully generated the alert for data user massages | Successful |

**8.3.1 UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**8.3.2 FUNCTIONAL TEST**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

**8.3.3 SYSTEM TEST**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**8.3.4 PERFORMANCE TEST**

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5 INTEGRATION TESTING**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**8.3.6 ACCEPTANCE TESTING**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**ACCEPTANCE TESTING FOR DATA SYNCHRONIZATION:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

**8.2.7 BUILD THE TEST PLAN**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**CHAPTER 9**

**APPLICATION**

**9.1 FUTURE ENHANCEMENT**

In the future, an adaptive scheduling algorithm can beincorporated with which files can be accessed multiple timesby the user as compared to the one which is accessed rarely.This will help to ensure that frequently accessed files areavailable easily to the user whenever required. Also, a creditsystem can be added with which each peer will be assigned adefault 100 credit, based on their system uptime, and severalsuccessfully served file access that requests their credits willbe either deducted or added. Peer’s with more credits will begiven higher priority for data storage.

**CHAPTER 10**

**10.1 CONCLUSION**

The proposed system enhances the security of data by encrypting and distributing the data across multiple peers inthe system. Implemented system uses the AES 256bitencryption algorithm to encrypt the data ensuring theconfidentiality of the user’s data. Encrypted data is thendistributed and stored across peers in the network using the IPFS protocol. Our system not only solves the privacy andsecurity concerns of centralized cloud storage but alsoprovides a medium for the peer to rent their underutilizedstorage and earn cryptocurrency in return thereby,maximizing the storage resource utilization.

**10.2 REFERENCES**

[1] Bernard Marr, ”How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read.” Forbes, 2018.

[2] Zhe, Diao, ”Study on Data Security Policy Based On Cloud Storage”2017 ieee 3rd international conference on big data security on cloud(bigdatasecurity), ieee international conference on high performance andsmart computing (hpsc), and ieee international conference on intelligentdata and security (ids) IEEE, 2017.

[3] Lee, Bih-Hwang, Ervin Kusuma Dewi, and Muhammad Farid Wajdi.”Data security in cloud computing using AES under HEROKU cloud.”2018 27th Wireless and Optical Communication Conference (WOCC).IEEE, 2018.

[4] Nakamoto, Satoshi,”Bitcoin: A peer-to-peer electronic cash system”,(2008).

[5] Zyskind, Guy, and Oz Nathan,” privacy: Using blockchain to protectpersonal data”,IEEE Security and Privacy Workshops. IEEE, 2015.

[6] Cachin, Christian,” Architecture of the hyperledger blockchain fabric”,Workshop on distributed cryptocurrencies and consensus ledgers. Vol.310. 2016.

[7] Buterin, Vitalik, ”A next-generation smart contract and decentralizedapplication platform”, white paper (2014).

[8] Ruj, Sushmita, et al, ”BlockStore: A Secure Decentralized StorageFramework on Blockchain” 2018 IEEE 32nd International Conferenceon Advanced Information Networking and Applications (AINA). IEEE,2018.

[9] Benet, Juan, “IPFS - Content Addressed, Versioned, P2P File System.”2014.

[10] Li, Dagang, et al. Meta-Key: ”A Secure Data-Sharing Protocol UnderBlockchain-Based Decentralized Storage Architecture”, IEEE NetworkingLetters 1.1 (2019): 30-33.

[11] Wohrer, Maximilian, and Uwe Zdun,”a Smart contracts: securitypatterns in the ethereum ecosystem and solidity”, InternationalWorkshop on Blockchain Oriented Software Engineering (IWBOSE).IEEE, 2018.

[12] Sum, V. “SECURIT Y AND PRIVACY MECHANISM USINGBLOCKCHAIN.” Journal of Ubiquitous Computing andCommunication T echnologies (UCCT ) 1.01 (2019): 45 -54

[13] Sivaganesan, D. “BLOCK CHAIN ENABLED INT ERNET OFnT HINGS.” Journal of Information T echnology 1.01 (2019): 1-8.