Contents

Chapter 1

INTRODUCTION

Cloud computing has recently reached popularity and developed into a major trend in IT. We perform such a systematic review of cloud computing and explain the technical challenges facing in this paper. In Public cloud the “Pay per use” model is used. In private cloud, the computing service is distributed for a single society. In Hybrid cloud, the computing services is consumed both the private cloud service and public cloud service. Cloud computing has three types of services. Software as a Service (SaaS), in which customer prepared one service and run on a single cloud, then multiple consumer can access this service as per on demand .Platform as a Service (PaaS), in which, it provides the platform to create application and maintains the application. Infrastructure as a Service (IaaS), as per term suggest to provides the data storage, Network capacity, rent storage, Data centers etc. It is also known as Hardware as a Service (HaaS). Cloud computing is the latest technology in the field of distributed computing. It provides various online and on-demand services for data storage, network services, platform services and etc.

At present, a large number of organizations are moving towards cloud for storing a large amount of data. Cloud Computing is a technology to provide services over the internet. Cloud act as Data Centre. A customer utilizes clouds resources and services and is charged accordingly. Security is the most important concern in cloud computing. There are many security issues of cloud computing which are related to trust, data confidentiality, authentication, access control etc. The impact of data security and the extent of loss that is suffered due to unauthorized access to cloud data motivates to take the problem as a challenge and come up with feasible solutions that can protect the data from theft, mishandling. The main focus is to study various issues and challenges with the types of cryptographic algorithm implemented in cloud computing environment.

DNA cryptography is used to encrypt message for secure communication on a network. It is a information carrier for transferring message from sender to receiver. For secure communication, it is not only to encrypt message but also necessary to hide encrypted message. DNA cryptography is also used for hiding the data, Hidden message is known by only sender and receiver. DNA computing is used to solve problems in cryptography, cryptanalysis and Steganography. DNA sequences based data encryption seems to be a promising strategy for fulfilling the current information security needs.

This paper focus on the comparative study of some existing works on DNA Cryptography. Message transmission is a process to transmit encrypted data through secure communication channel using DNA cryptography. Text message is encoded in DNA sequence. Message transmission process reduces the time complexity of transferring encrypted message. Bio molecular and one-time- pad technologies is used for secure message encryption.

Today, a large number of organizations are moving towards cloud for storing a large amount of data. Security is the most important concern in cloud. There are many security issues of cloud computing which are related to trust, data confidentiality, authentication, access control etc. These issues breach the CIA triad of cloud. The impact of data security and the extent of loss that is suffered due to unauthorized access to cloud data motivates to take the problem as a challenge and come up with feasible solutions that can protect the data from theft, mishandling. Many encryption techniques are available in cloud computing but some are efficient in terms of time and some are attack resistant. But the need is to propose a technique that is the combination of both.

The security concerns would be still there in cloud systems. Cryptography consists of the encryption-decryption techniques. There are many schemes, algorithms those can be used for security purposes. DNA is now mostly a theoretical concept but if implemented in future would be providing most strong and secure system. In this paper it has been proposed that DNA cryptographic algorithms are adopted for the optimization of data security in cloud security. So that cloud system would be most secure.

Cloud computing has a great capability to boost productivity and minimize costs, hence many companies are embracing it, but at the same time it constitutes many security risks and challenges. Due to possibilities of multiple risks such as service outage, theft of data, data leakage and the chances of malicious insider attack, using single cloud is becoming obnoxious by many companies and new notion of Multi-Clouds usage is becoming perceptible to cope with these security issues. This paper demonstrates powerful security strategy of using DNA based cryptography which ensures secure data storage on Multi-Clouds.

The advancement of technology rises many of new area of computer technology, cloud computing is one of them. It is a new conceptual based service that use by many small and big organization. In a cloud computing data may be stored at varied locations, both physically and geographically. Cloud computing support the client and server technology. Cloud computing have some important feature like cost effectiveness, easy to use and resource sharing, which proof the importance in the field of computer technology. So user wants to use their services to save their cost and expenditures. But one another important things is security of application and services that provided by cloud provider. It is a very big issue for cloud user that which service they are use, how much this is secure. Now days we are using many algorithms and cryptographic techniques for security of data. Some of them is very powerful and secure but some need to modification. In this paper we purpose a new approach of cryptography that is DNA cryptography. The idea behind to implement DNA cryptography is to enforce the other conventional cryptography techniques and algorithms. Our aim is to build a secure and confidential data over a cloud.

Cloud computing has a great potential to enhance productivity and reduce costs , hence many users are preferring it over various traditional technologies, but at the same time it gives rise to many security risks and challenges such as theft of data , data leakage and denial of service. Hence using single cloud is becoming obnoxious and the concept of multi-cloud is gaining popularity. This paper demonstrates use of DNA based cryptography to ensure secure data storage on multi-cloud.

1.1 Problem Statement

Many organizations are unenthusiastic to use cloud services due to data security issues as the data resides on the cloud services provider's servers. To address this issue, there have been several approaches applied by various researchers worldwide to strengthen security of the stored data on cloud computing .The Bi-directional DNA Encryption Algorithm (BDEA) is one such data security techniques. However, the existing technique focuses only on the ASCII character set, ignoring the non-English user of the cloud computing. Thus, this proposed work focuses on enhancing the BDEA to use with the Unicode characters.

1.2 Objective

Objective of our project is to

- Provide security parameter like confidentiality to stored message by using Bi-directional DNA Encryption Algorithm.
- Provide authenticity to system. So we can avoid system access by unauthorized user.

1.3 Applications

- Biomolecular Computation:

This method has been proposed to solve difficult combinatorial search problems such as the Hamiltonian path problem.

- DNA Cryptosystems Using Random One-Time-Pads:

One-time-pad encryption uses a codebook of random data convert plaintext to ciphertext.

- DNA Cryptosystem Using Substitution:

A substitution one-time-pad system uses a plaintext binary message and a table defining a random mapping to cipher text.

- DNA XOR One-time-pad cryptosystem:

A binary input string can be encoded using a single title for each bit. The titles are designed such that they assemble linearly to represent the binary string.

- Encrypting Images With DNA Chips and DNA One-Time-Pads**:**

In this application we outline a system capable of encryption and decryption of input and output data in the form of 2D images recorded using microscopic arrays of DNA on a chip.

- DNA Steganography Analysis:

Steganography using DNA is appealing due to its simplicity.

.

Chapter 2
LITREATURE SURVEY

In cloud computing the major issue is to provide the security of data. In Cloud computing data security is prepared by the Authentication, Encryption & Decryption, Message authentication code, Hash function, and Digital signature and so on. So here we discuss about some security problems and their solutions.

Use of Digital Signature with Diffie Hellman Key Exchange and AES Encryption Algorithm to Enhance Data Security in Cloud Computing [1].

Mr.PrashantRewagad and Ms.YogitaPawar [1]. Here in this paper, the researcher using three way architecture protection schemes. Firstly Diffie-Hellman algorithm is used to generate keys for key exchange step. Then digital signature is used for authentication, thereafter AES encryption algorithm is used to encrypt or decrypt user's data file. Diffie-Hellman key exchange algorithm is vulnerable to main in the middle attack. The most serious limitation is the lack of the authentication.

Union of RSA algorithm, Digital Signature and Kerberos in Cloud Security [2].

Mehdi Hojabri and Mona Heidari [2]. Here in this paper, the researcher first performs the concept of Kerberos authentication services. At the next step the Authenticate Server (AS) of Kerberos do verifies users and created the ticket granting ticket and session key and it sent to the users. The next step users send the ticket granting ticket and session

key to Ticket Granting Server (TGS) for getting the service. Then TGS send ticket and session key for user. In final step the users send the request service to cloud service provider for using the cloud service and also cloud service, provide service to users. After doing this step user can used the cloud service provider. But for more security they performed RSA algorithm for encryption & decryption and then they use Digital Signature for Authentication.

Implementation Digital signature with RSA Encryption algorithm to enhance the Data security of cloud in Cloud Computing [3].

Uma Somani, Kanika Lakhani, and Manish Mundra [3]. In this paper, there are two enterprises A and B. An enterprise A has some data that are public data and enterprise has public cloud. Now B wants some secure data from A's cloud. So RSA algorithm and Digital signature are used for secure communication. In this method, enterprise A takes data from cloud, which B wants. Now the data or document is crushed into little line using Hash code function that is called Message digest. Then A encrypts the message digest within private key the result is in the Digital signature form. Using RSA algorithm, A will encrypt the digital signed signature with B's public key and B will decrypt the cipher text to plain text with his private key and A's public key for verification of signature.

Enhancing security in cloud computing using Bi-Directional DNA Encryption Algorithm [4].

Cloud computing is the latest technology in the field of distributed computing. It provides various online and on-demand services for data storage, network services, platform services, etc. Many organizations are unenthusiastic to use cloud services due to data security issues as the data resides on the cloud services providers' servers. To address this issue, there have been several approaches applied by various researchers worldwide to strengthen security of the stored data on cloud computing. The Bi-directional DNA Encryption Algorithm (BDEA) is one such data security techniques. However, the existing technique focuses only on the ASCII character set, ignoring the non-English user of the cloud computing. Thus, this proposed work focuses on enhancing the BDEA to use with the Unicode characters.

Chapter 3

System Requirement and Specification

Software requirement specification is a fundamental document, which forms the foundation of the software development process. It not only lists the requirements of a system but also has a description of its major feature. An SRS is basically an organization's understanding (in writing) of a customer or potential client's system requirements and dependencies at a particular point in time (usually) prior to any actual design or development work. It's a two way insurance policy that assures that both the client and the organization understand the other's requirements from That perspective at a given point in time. The SRS also functions as a blueprint for completing a project with as little cost growth as possible. The SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specification, testing and validation phase, and documentation plans, are related to it. It is important to note that an SRS contains functional and non functional requirements only. It doesn't offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understand the customer's system requirements to be.

3.1 Functional Requirement

Functional requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. In this system following are the function requirements.

- **Registration:** Registration is required to access the system only by authorized user
- **Login:** After successful registration of the new user he/she should prove his/her authentication to server every time by entering Id and password .
- **Encryption:** Before sending the message to destination the sender will encrypt the message by using DNA cryptography to avoid unauthorized read/ access to the document.
- **Decryption:** At receiver side receiver should decrypt the message to read data.

3.2 Non-Functional Requirement

Non functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviours. They may relate to emergent system properties such as reliability, response time and store occupancy. Non functional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware system or because of external factors such as :-

1. Product Requirements
2. Organizational Requirements
3. User Requirements
4. Basic Operational Requirements.

3.2.1 Product Requirements

Portability: - Since the software is developed in java it can be executed on any platform for which the JVM is available with minor or no modification.

Correctness: - It followed a well-defined set of procedures and rules to compute and also rigorous testing is performed to confirm the correctness of the data.

Ease of Use: - The front end is designed in such a way that it provides an interface which allows the user to interact in an easy manner.

Modularity: - The complete product is broken up into many modules and well-defined interface are developed to explore the benefit of flexibility of the product.

Robustness: - This software is being developed in such a way that the overall performance is optimized and the user can expect the results within a limited time with almost relevancy and correctness. Java itself possesses the feature of robustness, which implies the failure of system is negligible.

Non functional requirements are also called the qualities of a system. These qualities can be divided into execution quality and evolution quality. Execution qualities are security and usability of the system which are observed during run time, whereas evolution quality involves testability, maintainability, extensibility or scalability.

3.2.2 Organizational Requirements

Process Standards: IEEE standards are used to develop the application which is the standard used by the most of the standard software developers all over the world.

Design Methods: Design is one of the important stage in the software engineering process. This stage is the first strep n moving from problem to the solution domain. In other words, starting with what is needed design takes us to work how to satisfy the needs.

The design of the system is perhaps the most critical factor affecting the quality of the software and has a major impact on the later phases, particularly testing and maintenance. We have to design the product with the standards which has been understood by the developers of the team.

3.2.3 User Requirements

- The user must be able to visualize the wireless sensor network.
- The user must be able to configure all the parameters with neat GUI.
- The user must be able to visualize the flow of packets from source to sink.

3.2.4 Basic Operational Requirements

The customers are those that perform the eight primary function of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, will be related to these following points: -

Mission profile or scenario: It describes about the procedures used to accomplish mission objective. It also finds out the effectiveness or efficiency of the system.

Performance and related parameters: It points out the critical system parameters to accomplish the mission.

Utilization environments: It gives a brief outline of system usage. Finds out appropriate environments for effective system operation.

Operational life cycle: It defines the system lifetime.

3.3 Resource Requirement

Netbean IDE 6.9.1: Netbean is a multi-language software development environment comprising an integrated development environment (IDE) and and extensible plug-in system. It is written primarily in java and can be used to develop applications in java and by means of the others. Netbean employs plug-ins in order to provide all of its functionality on top of (and including) the runtime system, in contrast to some other applications where functionality is typically hard coded. The Netbean SDK includes the Netbean java development tool(JDT), offering an IDE with a built-in incremental java compiler and a full model of the java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a workspace, in this case a set of metadata over a flat file space allowing external file modifications as long as the corresponding workspace “resource” is refreshed afterwards.

Swing: The java Foundation Classes (JFC) consists of five major parts: AWT, Swing, and Accessibility, Java 2D and Drag and Drop. Java 2D has become an integral part of

AWT, Swing is built on top of AWT, and Accessibility support is built into Swing. The five parts of JFC are certainly not mutually exclusive, and Swing is expected to merge more deeply with AWT in future versions of Java. Swing is a set of classes that provides more powerful and flexible components than are possible with the AWT. In addition to the familiar components, Swing supplies tabbed panes, scroll panes, tree, and tables. It provides a single API capable of supporting multiple look and feel so that developers and end-users are not locked into a single platform's look and feel. The Swing library makes heavy use of the MVC software design pattern, which conceptually decouples the data being viewed from the user interface controls through which it is viewed. Swing possesses several traits such as: -

1. Platform-independence
2. Extensibility
3. Component-oriented
4. Customizable
5. Configurable
6. Look and Feel

Platform independent both in terms of its expression and its implementation, extensibility which allows for the "plugging" of various custom implementations of specified framework interfaces users can provide their own custom implementation of these components to override the default implementations. Component orientation allows responding to a well-known set of commands specific to the component. Specifically, Swing components are java Beans components, compliant with the Java Beans Component Architecture specifications. Through customizable feature users will programmatically customize a standard Swing component by assigning specific borders, colours, backgrounds, opacities, etc, configurable that allows Swing to respond at runtime to fundamental changes in its setting. Finally look and feel allows one to specialize the look and feel of widgets, by modifying the default via runtime parameters deriving from an existing one, by creating one from scratch or beginning with J2SE 5.0, by using the Look and Feel which is configured with an XML property file.

3.4 Hardware Requirement

- Processor : Pentium 4 or higher.
- Primary memory : 1GB or more.
- Secondary storage : 20GB or more.

3.5 Software Requirement

➢ Operating system	: WINDOWS XP, 7, 8 & HIGHER VERSION.
➢ Programming Language	: JAVA
➢ IDE	: NETBEANS

Chapter 4

System Design

Design is a creative process: a good design is the key to effective system. The system “Design” is defined as “the process of applying various technique and principles for the purpose of defining a process or a system in sufficient details to permit its physical realization”. Various design features are followed to develop the system. The design specification describes the features of the system, the components or elements of the system and their appearance to end-users.

4.1 Fundamental Design Concepts

A set of fundamental design concepts has evolved over the past three decades. Although the degree of interest in each concept has varied over the years, each has stood the test of time. Each provides the software designer with a foundation from which more sophisticated design methods can be applied. The fundamental design concepts provide the necessary framework for “getting it right”. The fundamental design concepts such as abstraction, refinement, modularity, software architecture, control hierarchy, structural partitioning, data structure, software procedure and information hiding are applied in this project to getting it right as per the specification.

4.1.1 Input Design

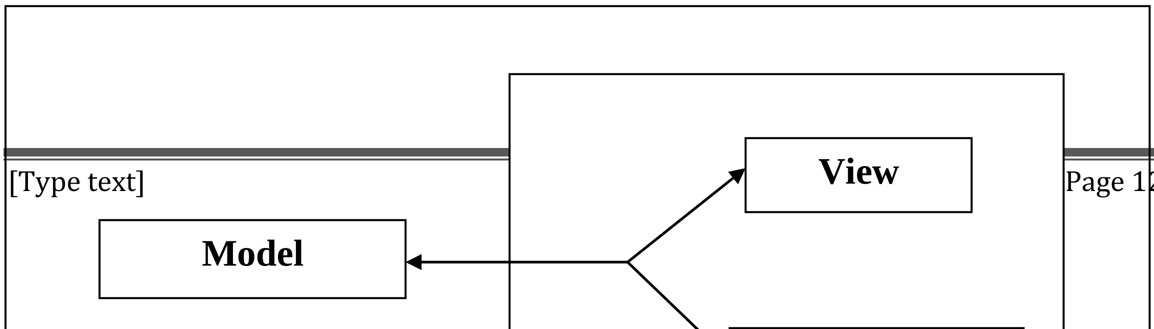
The input design is the process of converting the user-oriented inputs in to the computer based format. The goal of designing input data is to make the automation as easy and free from errors as possible. Providing a good input design for the application easy data input and selection features are adopted. The input design requirements such as user friendliness, consistent format and interactive dialogue for giving the right message and help for the user at right time are also considered for the development of the project. Input design is a part of overall system design which requires very careful attention. Often the collection of input data is the most expensive part of the system, which needs to be route through number of modules. It is the point where the user ready to send the data to the destination machine long with known IP address. If the IP address is unknown then it may prone to error.

4.1.2 Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. It is most important and direct source information to the user. Efficient and intelligent output improves the system relationship with source and destination machine. Outputs from computers are required primarily to get same packet that the user has send instead of corrupted packet and spoofed packets. They are also used to provide to permanent copy of these results for later consultation.

4.1.3 The MVC Design Method

A swing actually makes use of a simplified variant of the MVC design called the model-delegate. This design combines the view and the controller object into a single element that draws the component to the screen and handles GUI events known as the UI delegate. Communication between the model and the UI delegate becomes a two-way street. Each Swing component contains a model and a UI delegate. The model is responsible for maintaining information about the component's state. The UI delegate is responsible for maintain information about how to draw the component on the screen. The UI delegate (in conjunction with AWT) reacts to various events that propagate through the component.



Component

Fig 4.2 Combination of View and Controller into a UI delegate object

The design method that has been followed to design the architecture of the system is MVC design pattern. Swing uses the model-view-controller (MVC) architecture as the fundamental design behind each of its components. Essentially, MVC breaks GUI component into three elements. Each of these elements plays a crucial role in how the component behaves. The MVC design pattern separates a software component into three distinct pieces: a model, a view, and a controller.

Model

The model encompasses the state data for each component. There are different models for different types of components. For example, the model of a scrollbar component might contain information about the current position of its adjustable “thumb,” its minimum and maximum values, and the thumb’s width (relative to the range of values). A menu, on the other hand, may simply contain a list of the menu items the user can select from. Note that this information remains the same no matter how the component is painted on the screen; model data always exists independent of the component’s visual representation.

View

The view refers to how you see the component on the screen. For a good example of how views can differ, look at an application window on two different GUI platforms. Almost all window frames will have a title bar spanning the top of the window. However, the title bar may have a close box on the left side (like the older MacOS platform), or it may have the close box on the right side (as in the Windows 95 platform). These are examples of different types of views for the same window object.

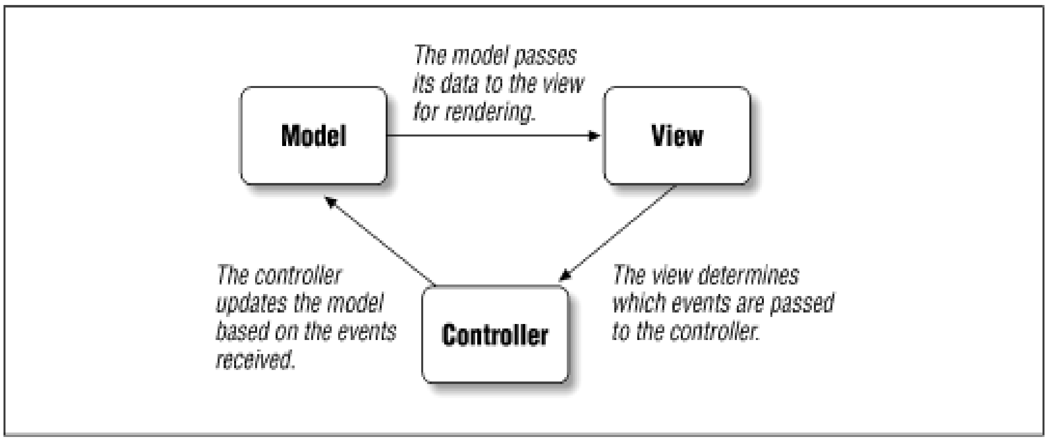

The model passes its data to the view for rendering.
Model
View
The controller updates the model based on the events received.
Controller
The view determines which events are passed to the controller.

Fig 4.2 Communication through the MVC architecture

Controller

The controller is the portion of the user interface that dictates how the component interacts with events. Events come in many forms — a mouse click, gaining or losing focus, a keyboard event that triggers a specific menu command, or even a directive to repaint part of the screen. The controller decides how each component will react to the event—if it reacts at all.

The view cannot render the scrollbar correctly without obtaining information from the model first. In this case the scrollbar will not know where to draw its “thumb” unless it can obtain its current position and width relative to the minimum and maximum. Likewise the view passes these events on to the controller, which decides how to handle them best. Based on the controller’s decision the values in the model may need to be altered. If the user drags the scrollbar thumb, the controller will react by incrementing the thumb’s position in the model. At that point the whole cycle can repeat.

The JFC user interface component can be broken down into a model, view and controller. The view and controller and combined into one piece, a common adaptation of the basic MVC pattern. They form the user interface for the component.

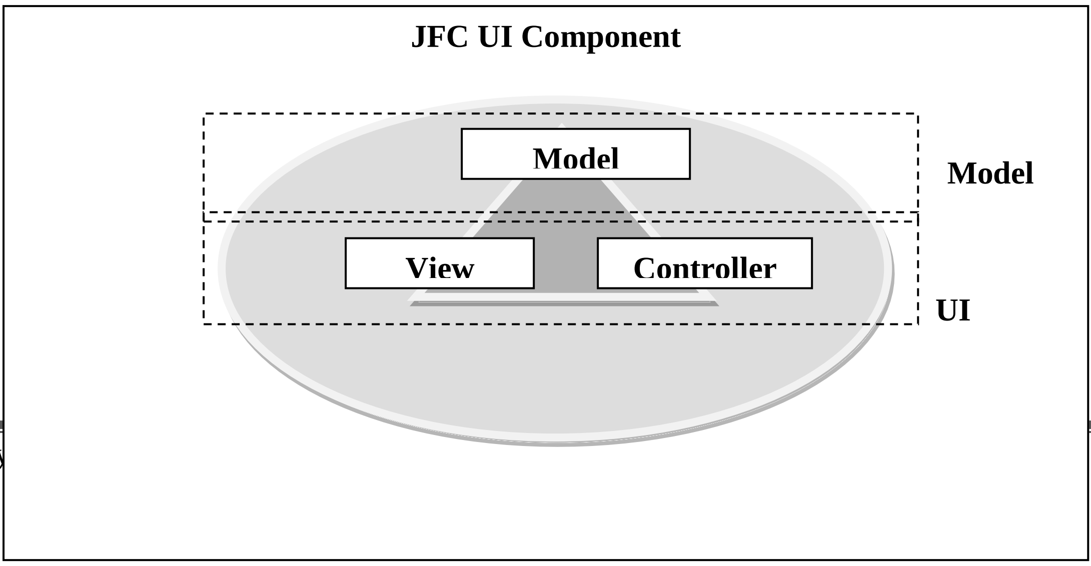


Fig 4.3 JFC user interface component

4.2 Software process model

Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.

4.2.1 Waterfall Model design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially. The below figure shows different phases of waterfall model.

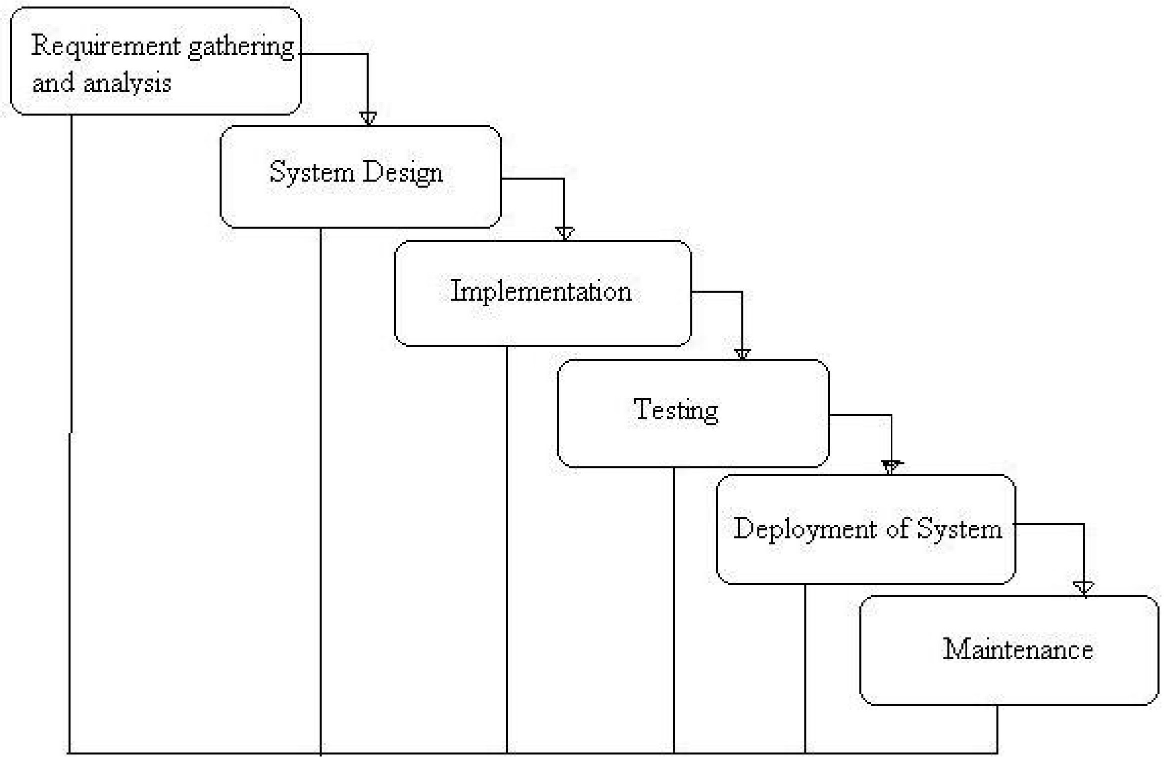


Fig 4.4: Waterfall model

The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- **Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system:** Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model phases do not overlap.

Advantages of waterfall model:

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are very well understood.

When to use the waterfall model:

- This model is used only when the requirements are very well known, clear and fixed.
- Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements
- Ample resources with required expertise are available freely

4.3 Methodology

Previous section describes the study about the cloud computing, basics of cloud computing and security problems occurs in cloud. Then study some papers to solve these security problems. Here in this paper, the Bi-serial DNA encryption algorithm is performing, that providing the two level of security.

4.3.1 DNA Digital Coding

In information science, the binary digital coding encoded by two state 0 or 1 and a combination of 0 and 1. But DNA digital coding can be encoded by four kind of base as shown in table 1. That is ADENINE (A), THYMINE (T), CYTOSINE (C) and GUANINE (G). There are possibly 4! = 24 pattern by encoding format like (0123/ATGC).

Table 1: DNA Digital Coding

Binary value	**DNA digital code**
00	A
01	T
10	G

11	C

Key Combination

Here in this work, we are using ATGC as a key. Every bit have 2 bits like A=00, T=01, G=10, and C=11 and by using ATGC, key combinations is generated and give numbering respectively that is given into table. From the table 2, we can generate 64 bit key values and adding ATGC, we can generate 72-bit key (64 bits of key combination and 8 bits of ATGC). ATGC key is sending to the receiver side by using Diffie Hellman key sharing algorithm. In this work, every time the key value will be randomly changed.

Table 2: Key combination

KEY COMBINATION	**PATTERNS**	**VALUES**
AA	0000	0
AT	0001	1
AG	0010	2
AC	0011	3
TA	0100	4
TT	0101	5
TG	0110	6
TC	0111	7
GA	1000	8
GT	1001	9
GG	1010	10
GC	1011	11
CA	1100	12
CT	1101	13
CG	1110	14
CC	1111	15

Encryption Process

Fig 4.5: encryption process

Decryption Process

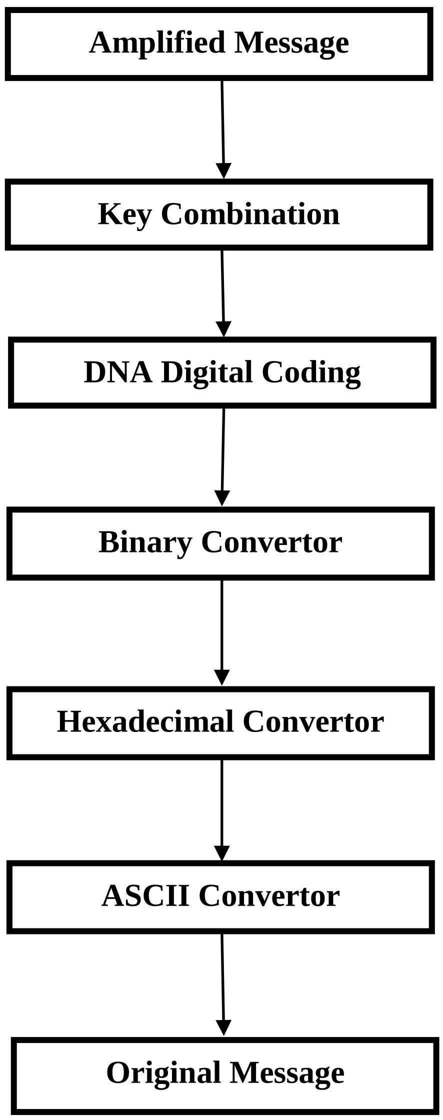
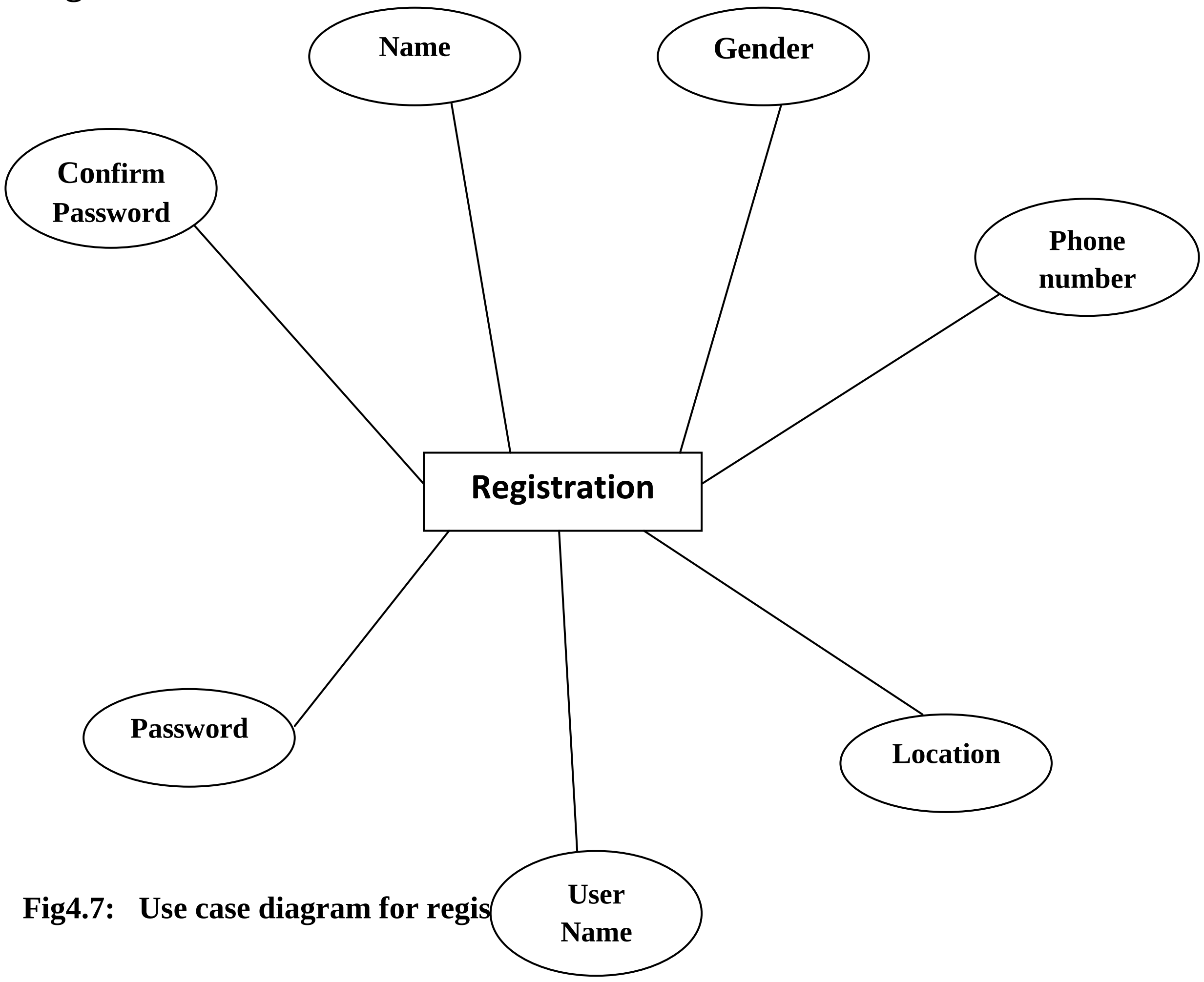


Fig4.6: Decryption process

4.4 Use case diagrams

Use case diagram is a simplest representation of a user's interaction with the system which also shows the relationship between the user and the other use cases where user is also involved. Use case diagram helps in identifying the different users of a particular system and different use cases.

1. **Registration**



2.

Fig4.7: Use case diagram for regis

3. **Login**

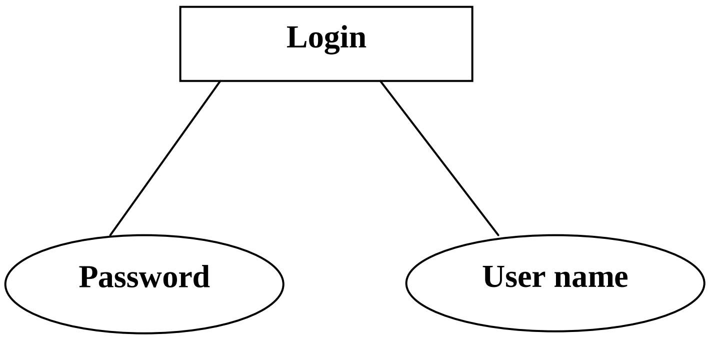
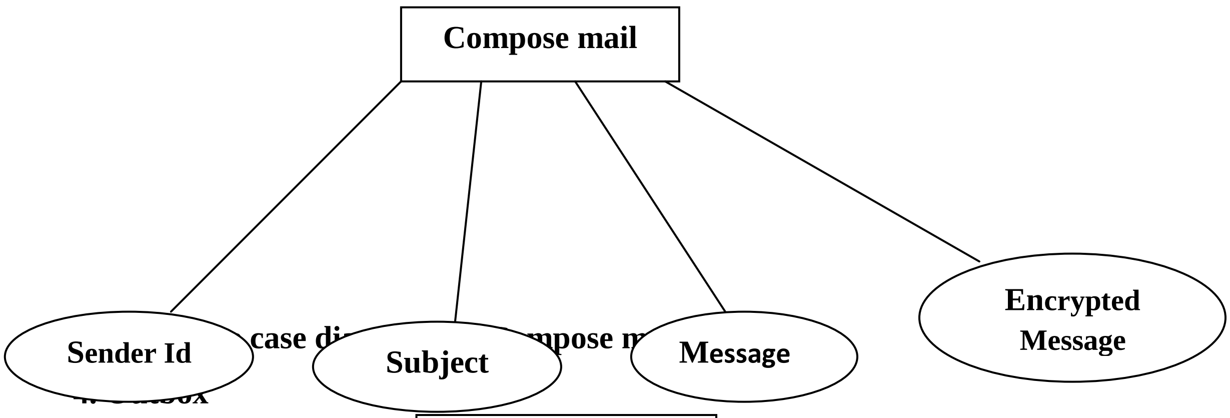


Fig4.8: Use case diagram for Login.

4. **Compose Mail**



case di mpose m

4. Outbox

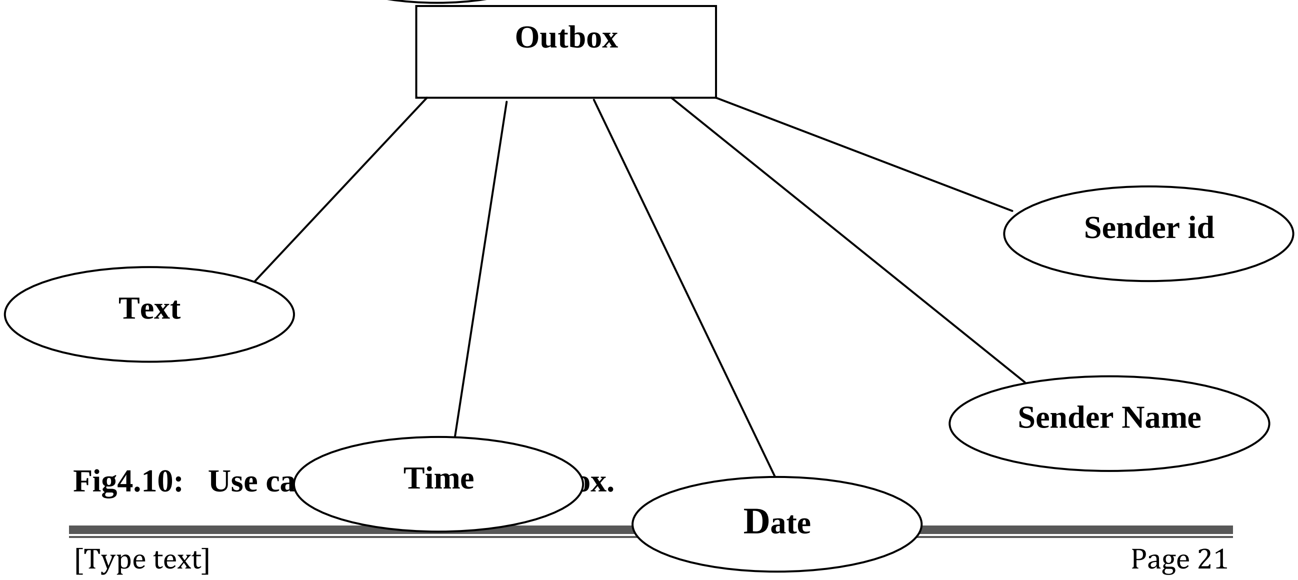


Fig4.10: Use ca ox.

4. Inbox

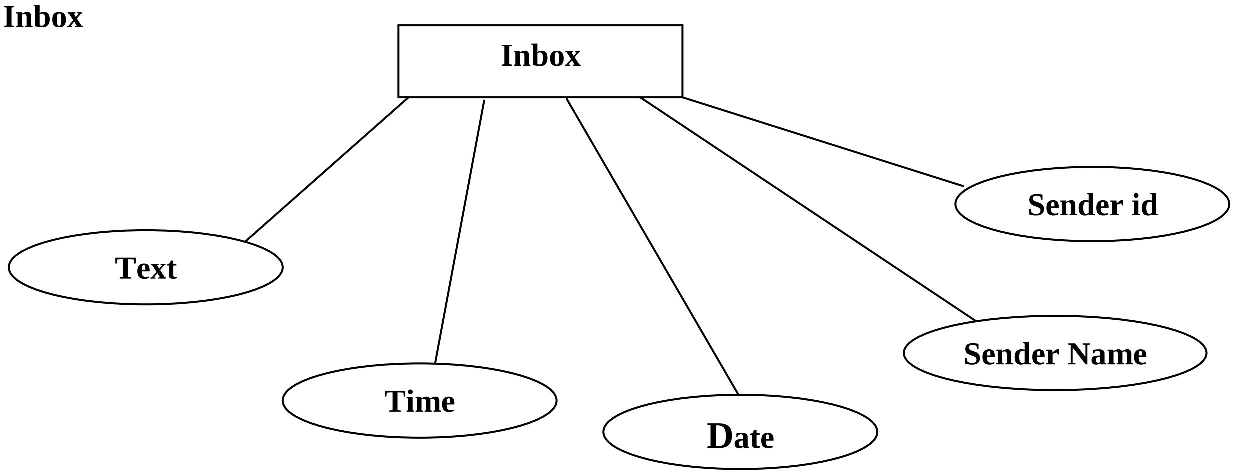


Fig4.11: Use case diagram for Inbox.

4.5 Sequence Diagram

Sequence diagram are very easy to understand. Sequence diagram describes the interaction between system and environment. A sequence diagram shows interactions arranged in a time sequence. The below figure shows completing working of the Efficient ID-Based Aggregate Signature Scheme for Wireless Sensor Networks.

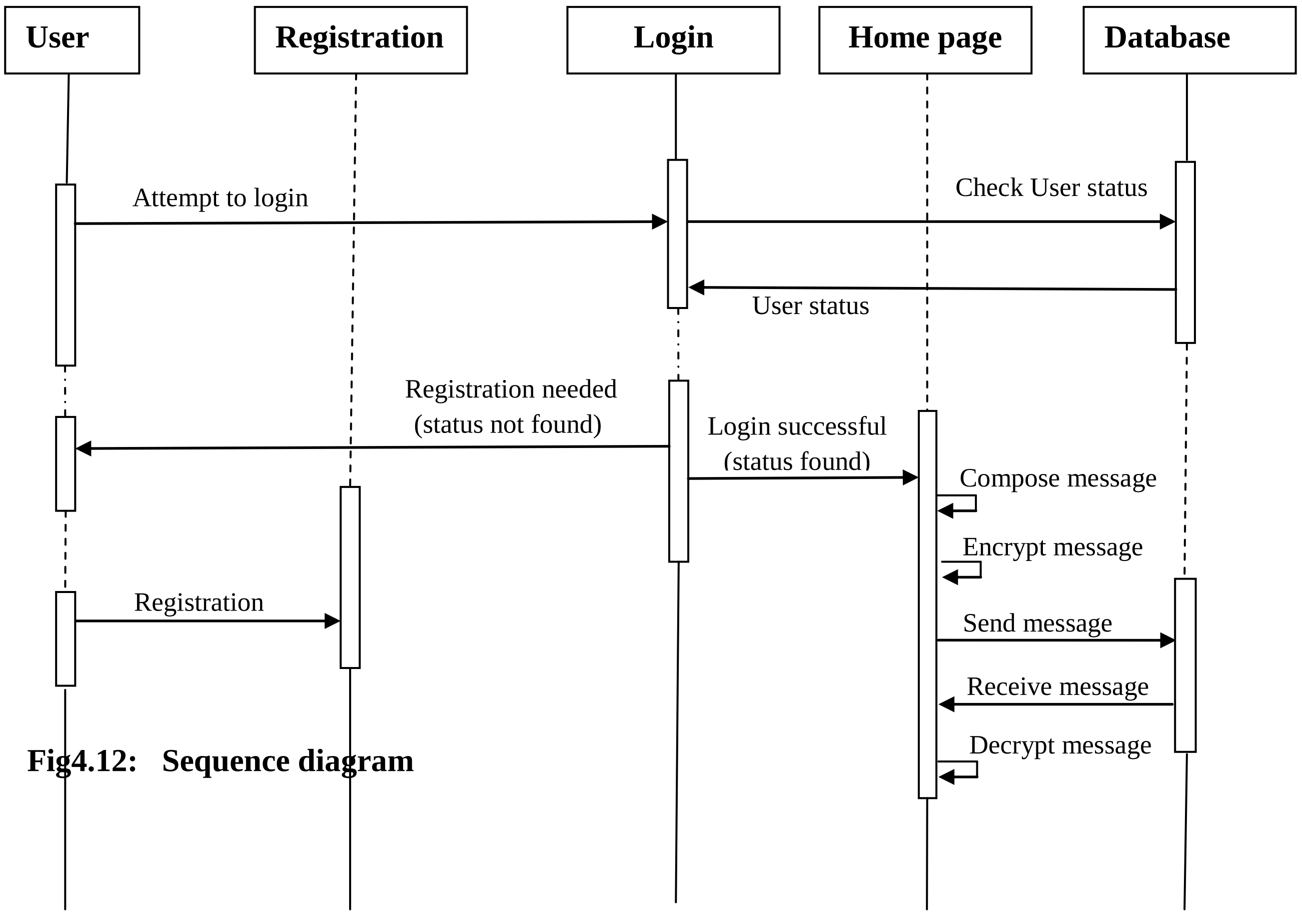


Fig4.12: Sequence diagram

4.6 Data Flow Diagram of the System

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams.

The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose.

The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

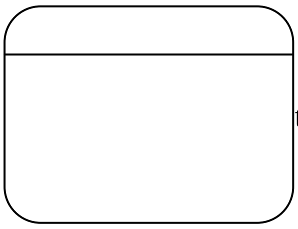
A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

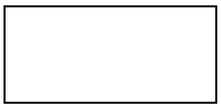
DFD SYMBOLS:

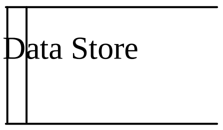
In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.

4. An open rectangle is a data store, data at rest or a temporary repository of data


transforms data flow.


Source or Destination of data



CONSTRUCTING A DFD:

Several rules of thumb are used in drawing DFD'S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

SAILENT FEATURES OF DFD'S

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

TYPES OF DATA FLOW DIAGRAMS

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

CURRENT PHYSICAL:

In Current Physical DFD proecess label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

CURRENT LOGICAL:

The physical aspects at the system are removed as mush as possible so that the current system is reduced to its essence to the data and the processors that transform them regardless of actual physical form.

NEW LOGICAL:

This is exactly like a current logical model if the user were completely happy with he user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

NEW PHYSICAL:

The new physical represents only the physical implementation of the new system.

RULES GOVERNING THE DFD'S

PROCESS

1) No process can have only outputs.
2) No process can have only inputs. If an object has only inputs than it must be a sink.
3) A process has a verb phrase label.

DATA STORE

1) Data cannot move directly from one data store to another data store, a process must move data.
2) Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store

3) A data store has a noun phrase label.

SOURCE OR SINK

The origin and /or destination of data.

1) Data cannot move direly from a source to sink it must be moved by a process
2) A source and /or sink has a noun phrase land

DATA FLOW

1) A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The later is usually indicated however by two separate arrows since these happen at different type.
2) A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
3) A data flow cannot go directly back to the same process it leads. There must be at least one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
4) A Data flow to a data store means update (delete or change).
5) A data Flow from a data store means retrieve or use.

Level 0 Data flow diagram

A context-level or level 0 data flow diagram shows the interaction between the system and external agents which act as data sources and data sinks. On the context diagram (also known as the Level 0 DFD) the system's interactions with the outside world are modeled purely in terms of data flows across the system boundary. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization.

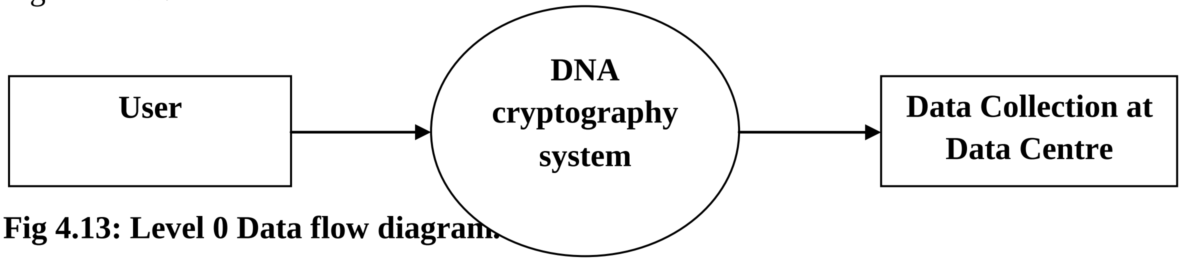


Fig 4.13: Level 0 Data flow diagram

Level 1 Data flow diagram

The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system**.**

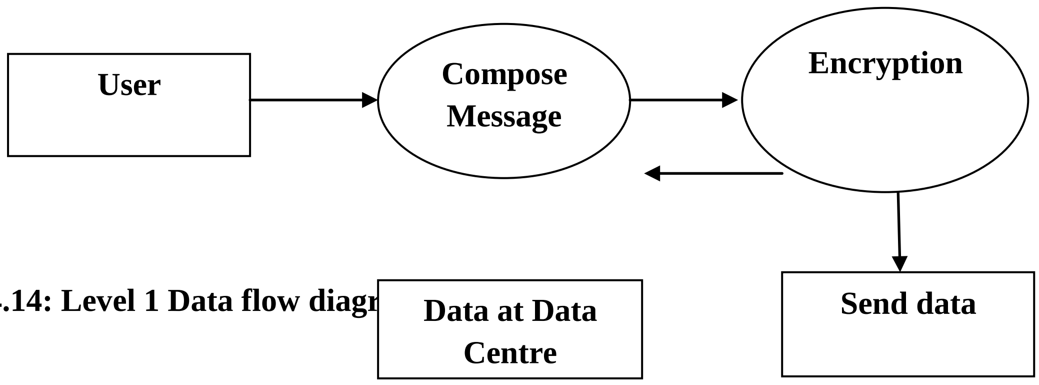


Fig 4.14: Level 1 Data flow diagr

4.7 ER Diagram of the System

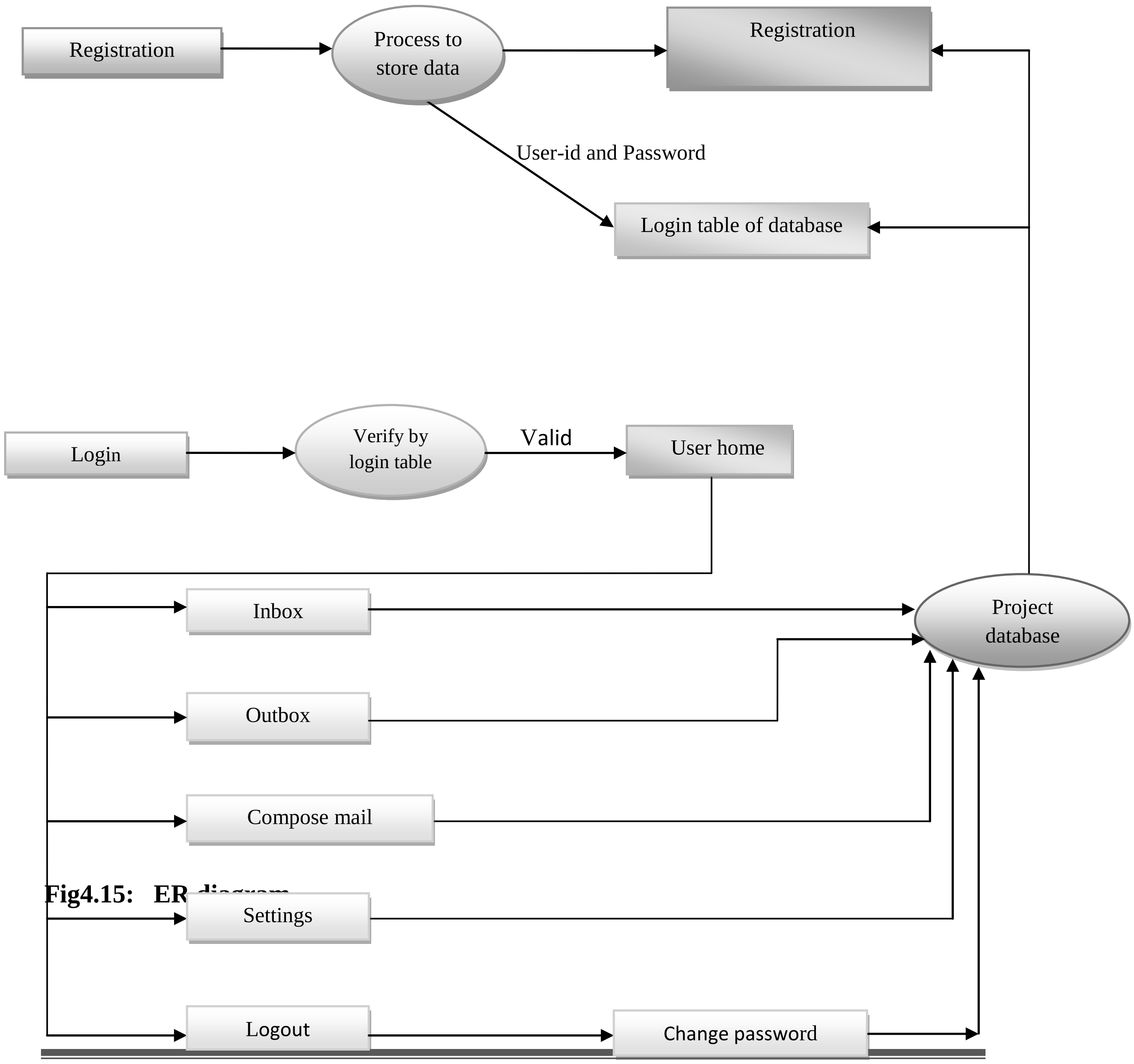


Fig4.15: ER diagram

Chapter 5
System Implementation

Implementation is a process of realizing a Technical specification or an algorithm as program, software elements using computer programming and deployment. Implementation process is considered as a very crucial stage as this stage of the proposed project gives the final solution for solving the issues associated with the domain. It can be observed that many processes of implementation can also be understood as materializing the ideas, which includes the detail analysis of the document for the proper design process in terms of perfect mapping to achieve the desire output. More often the project is ruined due to the improper selection of programming tools and using inappropriate programming methods. In this stage the programming language which uses the concept of object oriented technology can be applied for linking up the coding phase with the design phase. The object oriented concept is preferable for designing various phases of the proposed system. The below sections highlights the concerns pertaining to the choice of operating system, selection of tools, and selection of programming language.

5. 1 Introduction

Implementation of the proposed application is preceded by significant stages such as selection of platform, choice of coding language, etc. It is found that several parameters can be able to encourage the functionality of the system with respect to required speed, various security specifications and implementation details. The three major things that influence the implementation stage the most is:

- Choosing the platform.
- Choosing the programming Language to code the application.
- Selection of programming tool and following programming guidelines

5.2 Choosing the Platform

The operating system windows7 is a personal OS designed by Microsoft Company. It belongs to windows family of OS. It was initiate early as 2006 under the code name “Black bomb”. Microsoft Corporation released windows 7 for manufacturing on July 22 2009, and was available for general public on October 2009. This was basically planned to upgrade to previous version of windows OS, with a purpose of addressing problems in windows vista, while retaining software and hardware facility. The software can be applicable for updating the windows with the use of Aero using additional taskbar. Various requests are handled by the task bar which includes a new window management feature. In windows 7 various additional attributes or plug-ins were installed for updating which consists of many libraries, sharing of system home groups and a support which can be applicable to multi touch input scheme. A new "Action Centre" interface was also additionally provided, an additional over viewing of scheme safety and care particulars, and squeezes have been extended for managing the User Version Controller scheme to create it less susceptible to obtrusive. In difference to Vista, Windows 7 was well acclaimed by critics, who suggested the operating system showed a major advancement compared to its prior versions due to its enhanced performance, its intuitive interface (with specific praises devoted to newly designed taskbar), reduced User Account Control popup, and other advancement prepared diagonally the policy.

- It provides large number of new tools that highlights improved productivity by using enhanced usability.
- Provides the advantage of reduced boot time.
- Windows 7 provides better compatibility to run almost every software that is windows compatible.
- This operating system is designed to run almost all the software’s that are compatible with earlier version of windows such as windows xp and windows vista.
- Windows 7 provides the user with an advantage of much organized way and ease to locate documents.
- Windows 7 has reduced Application launch time.
- Windows 7 has better organizing feature of wireless connections.

5.3 Choosing the programming language to code application

By Definition a software design is representation for script plans. Which provide specification for computation or algorithms. A programming language governs a structural method for defining a code or data, operations or transformation or computations to be carried out automatically on the data. The programmer utilizes the abstraction present in the language to represent the concept or the computation. These are represented as the collection of simplest element called primitives. Programming is a procedure done which the packages uses the primitives in order to develop a new code for application or to adapt an existing code to new application or changing environment. Selection of the programming language depends on various factors like the application being developed, ease of coding and debugging, available resources and its compatibility.

5.3.1 Java

Java was at first grown by James Gosling at Sun Micro system. Java is a universally useful object oriented computer programming language particularly intended to have few implementation conditions as would be prudent. It permits the application designers "write once, run anywhere" (WORA) suggesting that complied java code can keep running in any platform that backings the java without being re-compiled. Java codes are assembled to byte code that keeps running on the Java virtual Machine (JVM) immaterial to underlying system architecture. Java language can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compiles interpret a program so that you can run it on your computer. The java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called java byte codes – the platform independent codes interpreted by the interpreter on the java platform. The interpreter parses and runs each java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed.

5.3.2 JDK Contents

There are various fundamental components of JDK which are a collection of various software design tools which includes the following:

- jdb-the debugger
- jhat- Java Heap Analysis Tool(Experimental)
- jinfo – This utility gets configuration information from a running java process or crash dump.(experimental).
- jmap – This utility outputs the memory map for java and can print shared object memory maps or heap memory details o a given process or core dump (experimental).
- Jmc-Java Misson Control.
- Jps – Java virtual machine process status tool lists the instrumented HotSpot java virtual machines (JVMs) on the target system (experimental).
- jrunscript- Java command line script shell.
- Jstack - utility which prints java stack traces of java threads (experimental).
- Jstat – Java virtual machines statics monitoring tool (experimental).
- Jstatd – jstat daemon (experimental).
- Keytool – tool fir manipulating the keystone.
- Pack200 – JAR compression tool
- Policytool – the policy cration and managemnet tool, which can determine policy for a java runtime, specifying which permission are available for code from various sources.
- visualVM – visual tool integrating several command-line JDK tools and lightweight performance and memory profiling capabilities.
- W simport – generates portable JAX-WS artifacts for invoking a web service.
- Xjc – Part of the java API for XML binding (JAXB) API. It accepts an XML schema and generates java classes.
- Applet viewer – this tool can be used to run and debug java applets without a web browser.
- Apt –the annotation processing tool.
- Extcheck – a utility which can detect JAR file confilcts.
- Idlj – the IDL-to-java complier. This generates javaa binding from a giving java IDL file.

- Jab switch – the java access bridge. Exposes assistive technologies on microsoft windows systems.
- Java – the loader for java application. This tool is an interpreter and can interpret the class files generated by the javac compiler. Now a single launcher is used for both development and deployment. The old launcher, jre, no loner comes with Sun JDK, and instead it has been replaced by this new java loader.
- Javac – the java complier, which converts source code into java byte code.
- Javadoc – the documentation generator, which automatically generates documnetation from source code comments.
- Jar – the archive, which packages related class libraries into a single JAR file. This tool also helps manage JAR files.
- Javafxpackager – tool to package and sign JavaFX applications.
- Jarsigner – the jar signing and verification tool
- Javah – the C header and stub generator, used to write native methods.
- Javap – the class file disassembler.
- Javaws – the java web start launcher for JNLP applications
- Jconsole – Java monitoring and management console.
- Jdb – the debugger
- Jhat – java heap analysis tool .
- Jinfo – This utility gets configuration information from a rumming java process or crach dump.
- Jamp- This utility gets configuration information from a running java process or crash dump.
- Jmap – This utility outputs the memory map for java and can print shared object memory maps or heap memory details of a given process or core dump.
- Jmc- java Mission Control
- Jps – Java virtual machine process status tool lists he instrumented HotSpot java virtual machines (JVMs) on the target system.
- Jrunscript – java command line script shell.
- Jstack – utility which prins java stack traces of java threads
- Jstat- java virtual machine statistics monitoring tool
- Jstatd – jstat daemon
- Keytool – tool for manipulating the keystone.

- Pack200 – JAR compression tool
- Plicytool – the policy cration and managenent tool, which can determine policy for a java runtime, specifying which permission are available for code from various source.
- VisualVM – visual tool integrating several command-line JDK tools and lightweight performance and memory profiling capabilities.
- W simport- generates portable JAX-WS artifacts for invoking a web services.
- Xjc – part of the java API for XML binding (JAXB) API. It accepts an XML schema and genrates java classes.

The Future version of JDK may not contain Experimental tool. The JDK is also available with a Total Java Runtime Environment, commonly known as Private Runtime, since it is isolated from the "regular" JRE and has additional contents. It includes JVM and all the class library files that are present in the development environment, also includes extra libraries that are only used by developers like internationalization libraries and IDL libraries. Copy of the JDK also consists of a wide class of example programs illustrating the use of almost every feature of Java API.

5.3.3 Jar Files

In java based applications JAR (Java Archive) is considered as a wrapped file format which is mainly utilized for combining different class files which are also connected with metadata where it can be also used for storing properties (text, images, etc.) into single file, in order to distribute application software or library on the Java platform. JAR files are known as Basic archive files which are developed on the basis of ZIP file format and also contains .jar file extension where the JAR files can be created or take out by the user with the use of jar command which is available in JDK. It can also be achieved using zip tool .The order of zip file entries are important during compression, since manifest needs to be first. File name of JAR is Unicode.

5.3.4 Design and Enable Jar Files

JAR Files empower the Java runtime for deploying set of classes and their related assets effectively. The substance in the JAR files can be compressed ,which gives the capacity of whole application being downloaded in a solitary step, and is much helpful contrasted with the separately downloading the diverse uncompressed files that can make a single Java oriented Application. The package termed as java.util.zip likewise comprises of classes that can be used for reading and writing the JAR files. JAR files consists of an optional manifest file that is located in path META-INF/MANIFEST.MF. The usability of the Jar file is defined in the entry of the jar file.

For example a class path entry can be utilized in order to specify any different JAR files in order to load the respective JAR file. This entry also consists of a list which contains absolute path or relative paths of the other JAR. Executable Java Program is packaged within file which is in JAR format, with respect to any library the program will use. Executable JAR files contain the manifest which specifies the entry point class. Some operating systems run these processes directly on clicking. Native launcher is possible to be created on most of the platforms. For instance in Microsoft Windows users preferring to have Windows EXE files are allowed to use tools like JSmooth as well as Launch4J etc.

5.4 Module Implementation

Encryption Module

```
public String convertStringToHex(char[] str){
              StringBuffer hex = new StringBuffer();
              for(int i = 0; i < str.length; i++){
                hex.append(Integer.toHexString((int)str[i]));
              }
              return hex.toString();
  }
public String convertHexToString(String hex){
  StringBuilder sb = new StringBuilder();
StringBuilder temp = new StringBuilder();
  for( int i=0; i<hex.length()-1; i+=2 ){
String output = hex.substring(i, (i + 2));
int decimal = Integer.parseInt(output, 16);
sb.append((char)decimal);
temp.append(decimal);
              }
  System.out.println("Decimal : " + temp.toString());
return sb.toString();
  }
```

```
     public static String hexToBinary(String hex) {
         StringBuilder binStrBuilder = new StringBuilder();
                int c = 1;
                for (int i = 0; i < hex.length() - 1; i += 2) {
            String output = hex.substring(i, (i + 2));
            int decimal = Integer.parseInt(output, 16);
            String binStr = Integer.toBinaryString(decimal);
                 int len = binStr.length();
                   StringBuilder sbf = new StringBuilder();
                        if (len < 8) {

                                for (int k = 0; k < (8 - len); k++) {
                                        sbf.append("0");
                                }
                                sbf.append(binStr);
                        } else {
                                sbf.append(binStr);
                        }
            c++;
                        binStrBuilder.append(sbf.toString());
                }
return binStrBuilder.toString();
}
public static String binaryToHex(String bin) {
      String s = bin;
BigInteger b = new BigInteger(s, 2);
System.out.println(b.toString(16));
String value=b.toString(16);
return value;
}
    private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
        String mail=txtMail.getText();
        String sub=txtSub.getText();
        String text=txtData.getText();
         if(Pattern.matches(patternEmail, mail))
         {
             int checKemail=DataBase.checkIfEmail(mail);
             if(checKemail>0)
             {
                   btnSend.setEnabled(true);
                   pnl1.setEnabled(true);
                    btnEncrypt.setEnabled(false);
                  char[] ascii1 = text.toCharArray();

    for(char ch:ascii1){
        System.out.println((int)ch+"  ");
     txtAsciiValue.append(""+(int)ch);
    }
    String str=null;
        for(int i: ascii1){
         str = Character.toString((char)i);
            System.out.println(str);
        }
            String x=convertStringToHex(ascii1); //ascii to hex
        System.out.println("Hexadecimal Value:"+x);
        txtHexadecimal.setText(x);
       String b=hexToBinary(x); //hexa to binary
       System.out.println("Hex to Binary : "+b);
       txtBinaryValue.setText(b);
        String bin1;
        if (b.length() % 2 == 0)
            {
                bin1 = b;
            }
            else
            {
                bin1 = b + "0";
            }
        System.out.println("Binary With Append: "+bin1);
                System.out.println(b+"\n"+bin1);
       String a;
       String sum="";
   for (int i = 0; i < bin1.length(); i = i + 2)
            {
                a = bin1.substring(i, i+2);
                if ("01".equals(a))
                {
                    a = "T";
                }
                else
                    if ("00".equals(a))
                {
                    a = "A";
                }
                else if ("10".equals(a))
                {
                    a = "G";
                }
                else if ("11".equals(a))
                {
                    a = "C";
                }

                sum += a;
                //System.out.println(sum);
             }
   System.out.println("sum value: "+sum);
   txtDNADigitalCode.setText(sum);
   String sum1="";
   String c;
    for (int i = 0; i < sum.length(); i = i + 2)
            {
```

```
c = sum.substring(i, i+2);
if ("AA".equals(c))
{
    c = "0101";
}
else if ("AC".equals(c))
{
    c = "0010";
}
else if ("AT".equals(c))
{
    c = "0011";
}
else if ("AG".equals(c))
{
    c = "0001";
}
else if ("TA".equals(c))
{
    c = "0110";
}
else if ("TT".equals(c))
{
    c = "1111";
}
else if ("TG".equals(c))
{
    c = "0111";
}
else if ("TC".equals(c))
{
    c = "1001";
}
else if ("GA".equals(c))
```

```
            {
               c = "1010";
            }
            else if ("GT".equals(c))
            {
               c = "0100";
            }
            else if ("GG".equals(c))
            {
               c = "1000";
            }
            else if ("GC".equals(c))
            {
               c = "1100";
            }
            else if ("CA".equals(c))
            {
               c = "1110";
            }
            else if ("CT".equals(c))
            {
               c = "1011";
            }
            else if ("CG".equals(c))
            {
               c = "0000";
            }
            else if ("CC".equals(c))
            {
               c = "1101";
            }
            sum1 += c;
         }
```

Decryption Process

```
public static String binaryToHex(String bin) {
   String s = bin;
BigInteger b = new BigInteger(s, 2);
System.out.println(b.toString(16));
String value=b.toString(16);
return value;
}
public String convertHexToString(String hex){
              StringBuilder sb = new StringBuilder();
              StringBuilder temp = new StringBuilder();

              //49204c6f7665204a617661 split into two characters 49, 20, 4c...
              for( int i=0; i<hex.length()-1; i+=2 ){
                  String output = hex.substring(i, (i + 2));
                  //convert hex to decimal
                  int decimal = Integer.parseInt(output, 16);
                  //convert the decimal to character
                  sb.append((char)decimal);
                  temp.append(decimal);
              }
              System.out.println("ASCII : " + temp.toString());
txtAsciiValue2.setText(temp.toString());
              return sb.toString();
  }
    private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String msg=txtMsg.getText().trim();
       btnDecrypt.setEnabled(false);
        pnl3.setVisible(true);
        String b;
        String sum="";
         for (int i = 0; i < msg.length(); i = i + 4)
            {
```

```
// mList.Add(binary.Substring(i, 2));
b = msg.substring(i, i+4);
if ("0101".equals(b))
{
   b = "AA";
}
else if ("0010".equals(b))
{
   b = "AC";
}

else if ("0011".equals(b))
{
   b = "AT";
}
else if ("0001".equals(b))
{
   b = "AG";
}
else if ("0110".equals(b))
{
   b = "TA";
}
else if ("1111".equals(b))
{
   b = "TT";
}
else if ("0111".equals(b))
{
   b = "TG";
}
else if ("1001".equals(b))
{
   b = "TC";
}
else if ("1010".equals(b))
{
   b = "GA";
}
else if ("0100".equals(b))
{
   b = "GT";
}
else if ("1000".equals(b))
{
   b = "GG";
}
else if ("1100".equals(b))
{
   b = "GC";
}
else if ("1110".equals(b))
{
   b = "CA";
}
else if ("1011".equals(b))
{
   b = "CT";
}
else if ("0000".equals(b))
{
   b = "CG";
}
else if ("1101".equals(b))
{
   b = "CC";
}
sum += b;
```

```
    }
   System.out.println("sum Value: "+sum);
txtDNADigitalCode2.setText(sum);
   String a;
   String sum1="";
    System.out.println("dna");
   for (int i = 0; i < sum.length(); i = i + 1)
    {
       // mList.Add(binary.Substring(i, 2));
       a = sum.substring(i, i+1);

       if ("T".equals(a))
       {
         a = "01";
       }
       else if ("A".equals(a))
       {
         a = "00";
       }
       else if ("G".equals(a))
       {
         a = "10";
       }
       else if ("C".equals(a))
       {
         a = "11";
       }
       sum1 += a;
    }
    System.out.println("Sum1 value: "+sum1);
    txtBinaryValue2.setText(sum1);
     String bin2;
      String hex=binaryToHex(sum1);
      txtHexadecimal2.setText(hex);
```

```
        System.out.println("hex value: "+hex);
         String mssg=convertHexToString(hex.trim());
         System.out.println("msg"+mssg);
         jTextArea1.setText(mssg);
  }
```

5.5 Database

Table Name: "reg"

Sl.no	**Column Name**	**Data Type**	**Constraints**
1	Id	integer	PRIMARY KEY
2	name	varchar(100)	NOT NULL
3	gender	varchar(100)	NOT NULL
4	phonenumber	varchar(100)	NOT NULL
5	location	varchar(100)	NOT NULL
6	username	varchar(100)	NOT NULL
7	password	varchar(100)	NOT NULL

Table Name: "userdetails"

Sl.no	**Column Name**	**Data Type**	**Constraints**
1	usid	integer	PRIMARY KEY
2	sender	varchar(100)	NOT NULL
3	receiver	varchar(100)	NOT NULL
4	subject	varchar(500)	NOT NULL
5	senddate	varchar(100)	NOT NULL
6	sendtime	varchar(100)	NOT NULL
7	message	varchar(100)	NOT NULL
8	r_id	varchar(100)	NOT NULL

Chapter 6

Software Testing and Results

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all the system elements have been properly integrated and perform allocated functions. The testing process is actually carried out to make sure that the product exactly does the same thing what is supposed to do. Testing is the final verification and validation activity within the organization itself.

6.1 Testing Objectives

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. There are different types of tests.

6.2 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of an individual software unit of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relays 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.

6.3 Integration

After successful completion of unit testing or module testing, individual functions are integrated into classes. Again integration of different classes takes place and finally integration of front-end with back-end occurs.

- **Integration of functions into classes**

  At the start of coding phase only the functions required in different parts of the program are developed. Each of the functions is coded and tested independently. After verification of correctness of the different functions, they are integrated into their respective classes.

- **Integration of different classes**

  Here the different classes are tested independently for their functionality. After verification of correctness of outputs after testing each class, they are integrated together and tested again.

➢ **Integration of front-end with back-end**

The front-end of the project is developed in java swing environment. The user interface is designed to facilitate the user to input various commands to the system and view the system’s normal and faculty behaviour and its output. The back-end code is then integrated with the GUI and tested.

6.4 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfied, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.5 Functional test

Functional tests provide systematic demonstration that functions tested are available as specified by 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.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

Classes integrated	**Functions integrated in each class**	**Test done**	**Remarks**
		Class tested to check whether all commands that were applied are working correctly or not.	Success

		Class tested to check whether all commands that were applied are working correctly or not.	Success
		Class tested to check whether all commands that were applied are working correctly or not.	Success
		Class tested to check whether all commands that were applied are working correctly or not.	Success

Table 6.1: Integration Testing Table.

6.6 Validation Testing

At the culmination of integration testing, software is completed and assembles as a package. Interfacing errors are uncovered and corrected. Validation testing can be defined in many ways. Here the testing validates the software function in a manner that is reasonably expected by the customer.

Functionality to be tested	**Input**	**Test done**	**Remarks**
Working of Front-End	User interaction with help of a mouse and keyboard	Appropriate forms open when buttons are clicked	Success
Working of create network	User click to create network	Sensor node with specified number of nodes is displayed.	Success
Working of run simulation	User clicks to send packet to data from source to destination	Data aggregation and routing of packet is shown in simulation	Success

Table 6.2: Validation Testing Table

6.7 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 description and flows, emphasizing pre-driven process links and integration points.

6.8 White Box Testing

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

6.9 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 documents. 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 works.

6.10 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.

6.11 Quality Assurance

Quality assurance consists of the auditing and reporting functions of management. The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confident that the product quality is meeting its goals. This is an “umbrella activity” that is applied throughout the engineering process. Software quality assurance encompasses: -

- Analysis, design, coding and testing methods and tools.
- Formal technical reviews that are applied during each software engineering.
- Control of software documentation and the change made to it.
- A procedure to ensure compliance with software development standards.
- Measurement and reporting mechanism.

6.11.1 Quality Factor

An important objective of quality assurance is to track the software quality and assess the impact of methodological and procedural changes on improved software quality. The factors that affect the quality can be categorized into two broad groups:

- Factor that can be directly measured.
- Factors that can be indirectly measured.

These factors focus on three important aspects of a software product

- Its operational characteristics.
- Its ability to undergo changes.
- Its adaptability to a new environment.
- Effectiveness or efficiency in performing its mission.
- Duration of its use by its customer.

6.11.2 Generic Risks

A risk is an unwanted event that has negative consequences. We can distinguish risks from other project events by looking for three things:

- A loss associated with the event.
- The likelihood that the event will occur.
- The degree to which we can change the outcome.

The generic risks such as the product size risk, business impact risks, customer related risks, process risks, technology risks, development environment risks, security risks etc. This project is developed by considering all these important issues.

6.11.3 Security Technologies and Policies

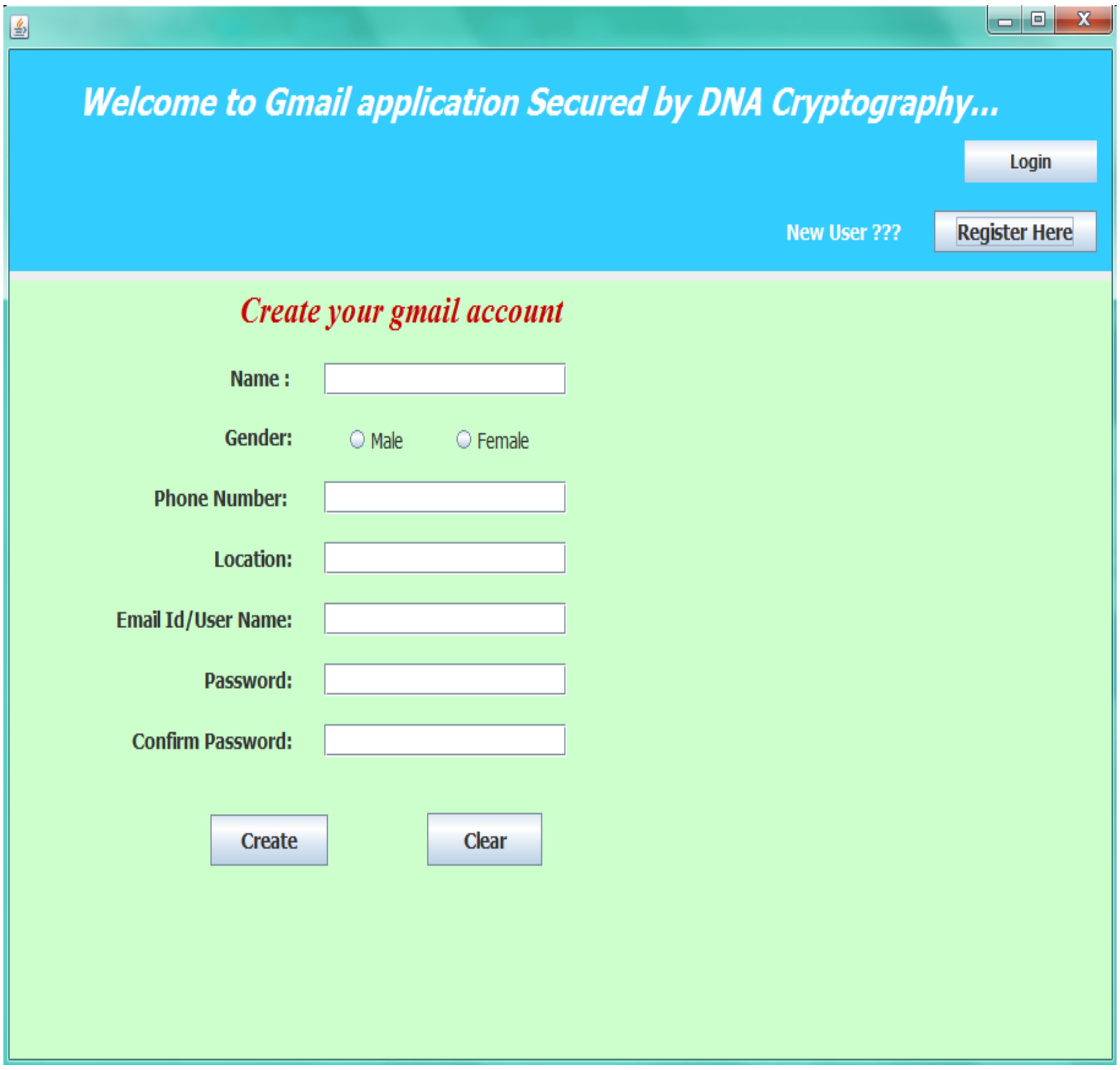
The software quality assurance is comprised of a variety of tasks associated with seven major activities: -

- Application of technical methods.
- Conduct of formal technical reviews.
- Software testing.
- Enforcement of standard.
- Control of change.
- Measurement.
- Record keeping and reporting.

Chapter 7

Snapshots

1. Registration Page


Welcome to Gmail application Secured by DNA Cryptography...

Login

New User ???
Register Here

Create your gmail account

Name :	
Gender:	○ Male ○ Female
Phone Number:	
Location:	
Email Id/User Name:	
Password:	
Confirm Password:	

Create
Clear

Fig 7.1: Registration page

2. **Login Page**

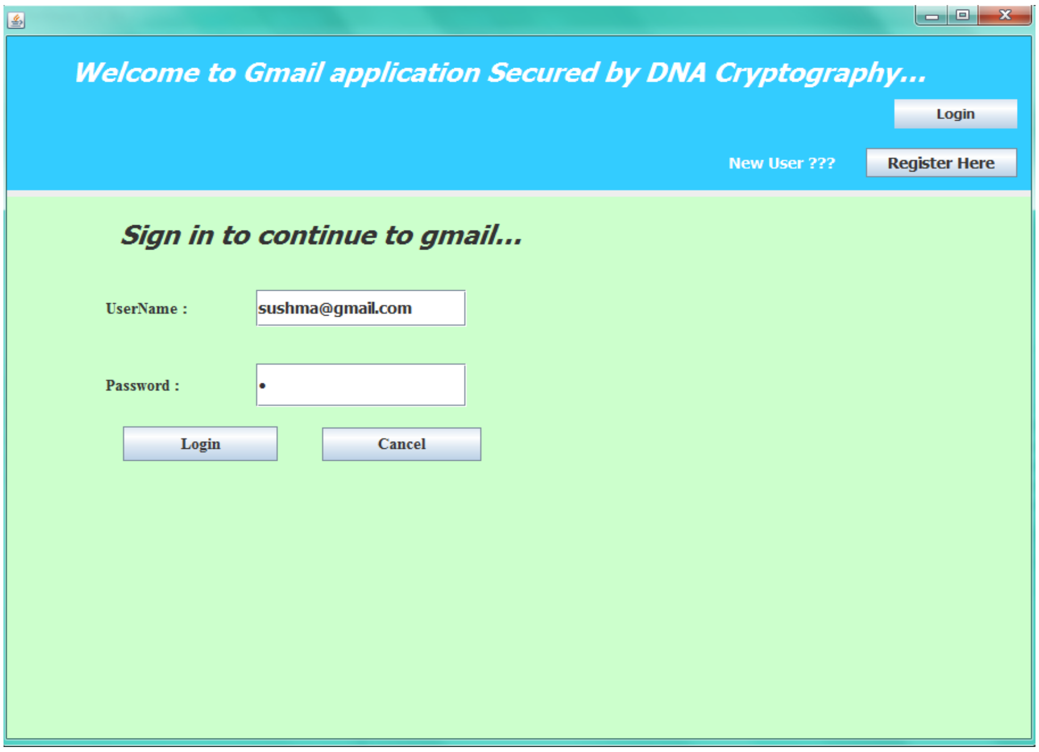

Welcome to Gmail application Secured by DNA Cryptography...
Login
New User ???
Register Here
Sign in to continue to gmail...
UserName :
sushma@gmail.com
Password :
•
Login
Cancel

Fig 7.2: Login Page

3. **User Home Page**

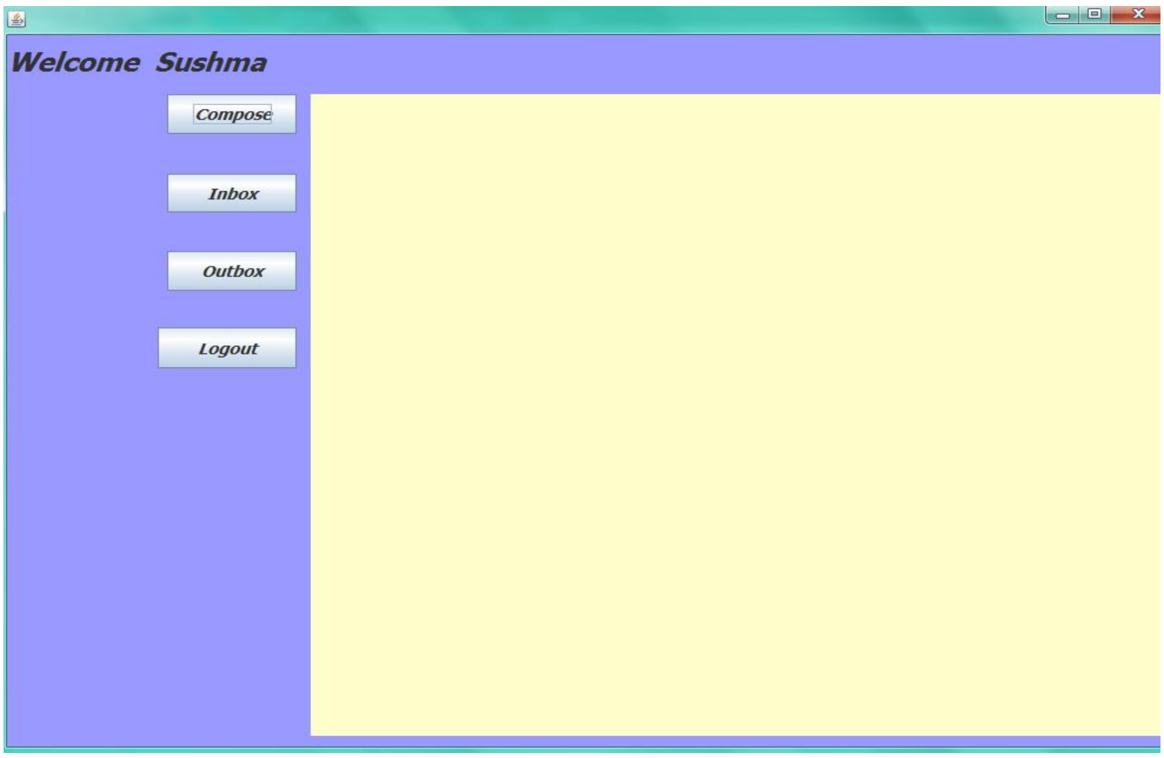
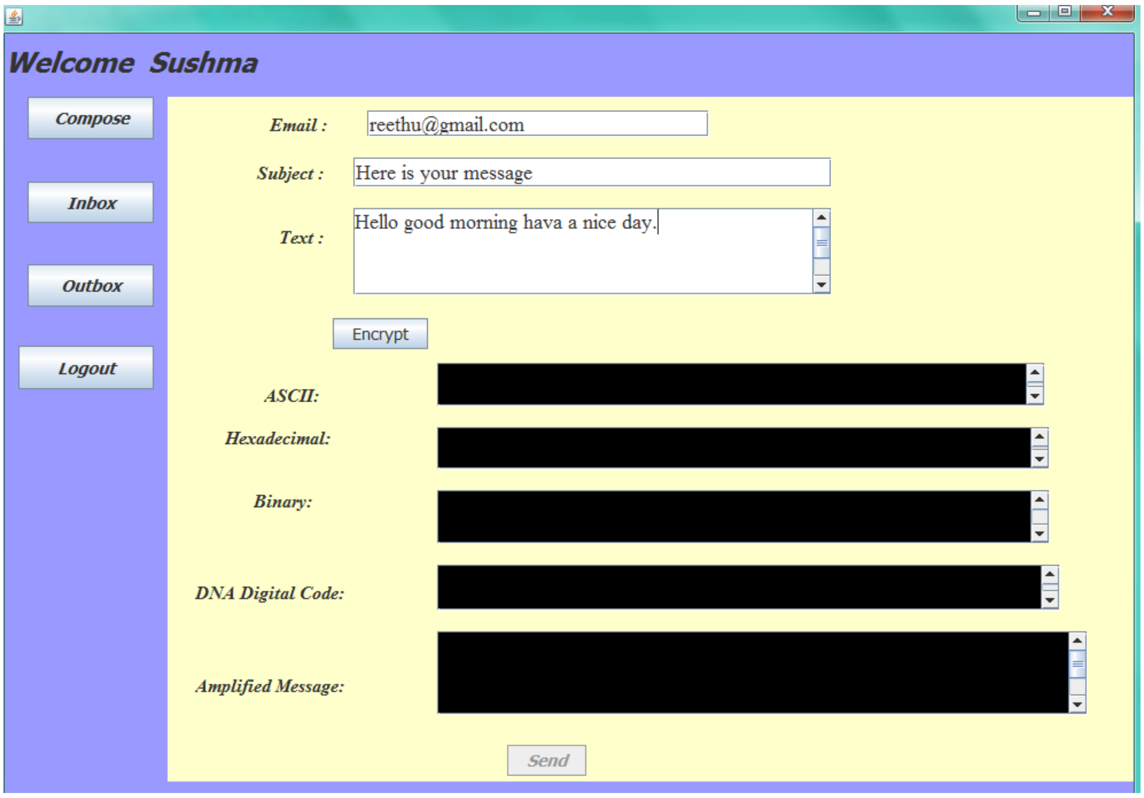


Fig 7.3: Home Page

4. Compose Form



Email : reethu@gmail.com

Subject : Here is your message

Text : Hello good morning hava a nice day.

ASCII:

Hexadecimal:

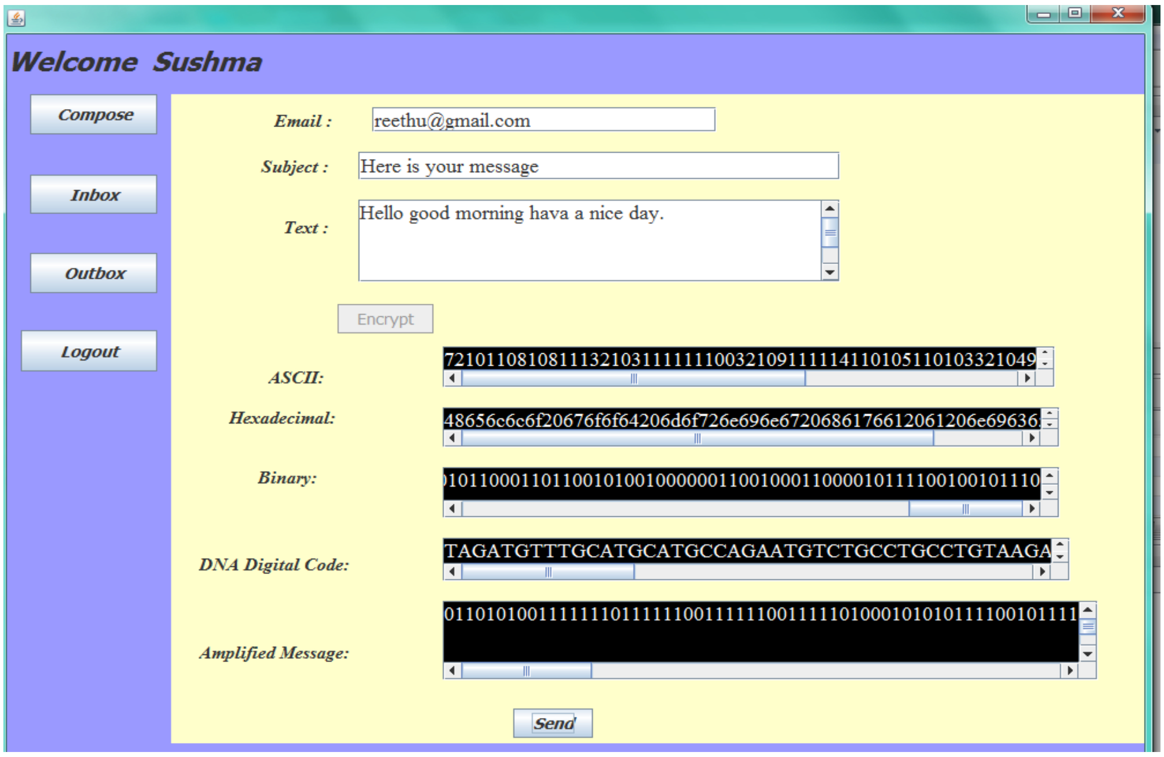
Binary:

DNA Digital Code:

Amplified Message:

Fig 7.4: Compose Page

5. Result of Encryption



Email : reethu@gmail.com

Subject : Here is your message

Text : Hello good morning hava a nice day.

ASCII: 72101108108111321031111111003210911111141101051101033210491

Hexadecimal: 48656c6c6f20676f6f64206d6f726e696e6720686176612061206e69636

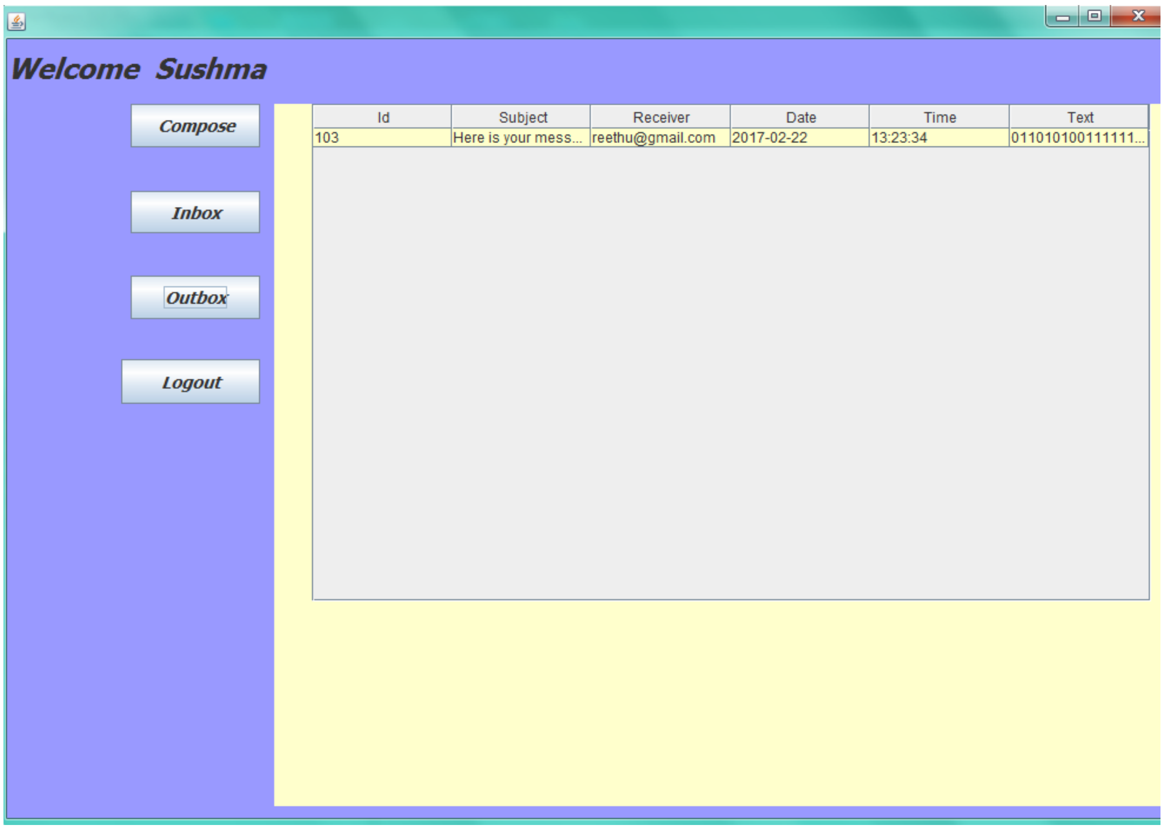
Binary: 101100011011001010010000001100100011000010111100100101110

DNA Digital Code: TAGATGTTTGCATGCATGCCAGAATGTCTGCCTGCCTGTAAGA

Amplified Message: 011010100111111101111110011111100111110100010101011110010111

Fig 7.5: Encrypted Text

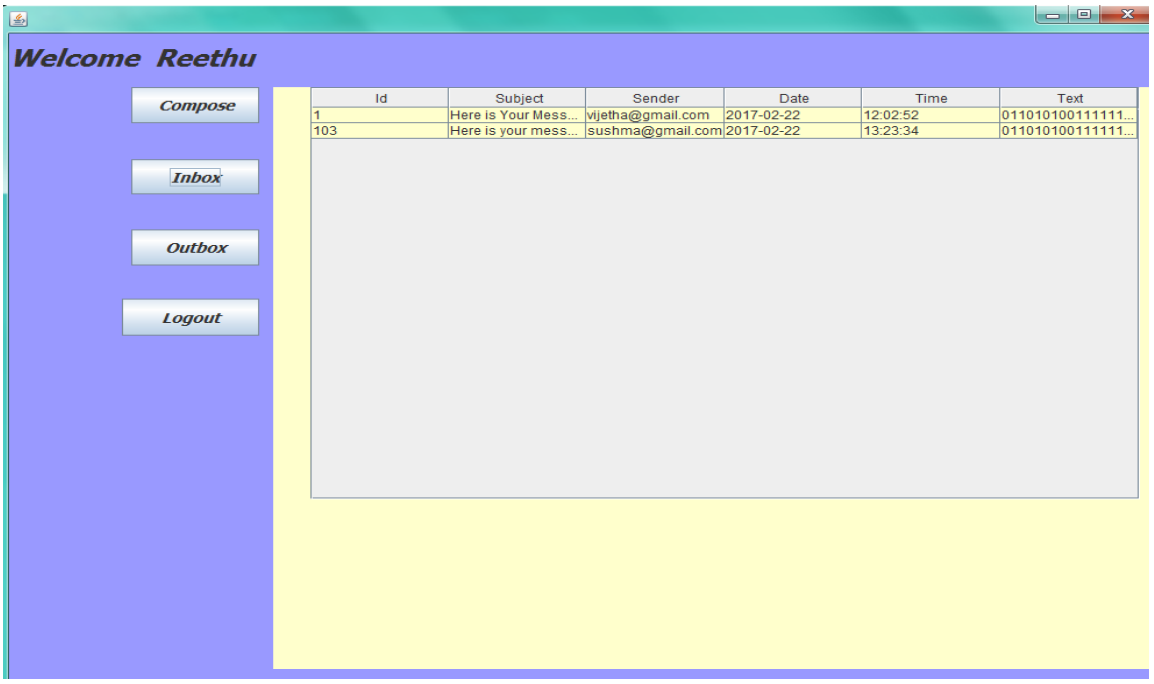
6. Out Box



Id	Subject	Receiver	Date	Time	Text
103	Here is your mess...	reethu@gmail.com	2017-02-22	13:23:34	011010100111111...

Fig 7.6: Out box Form

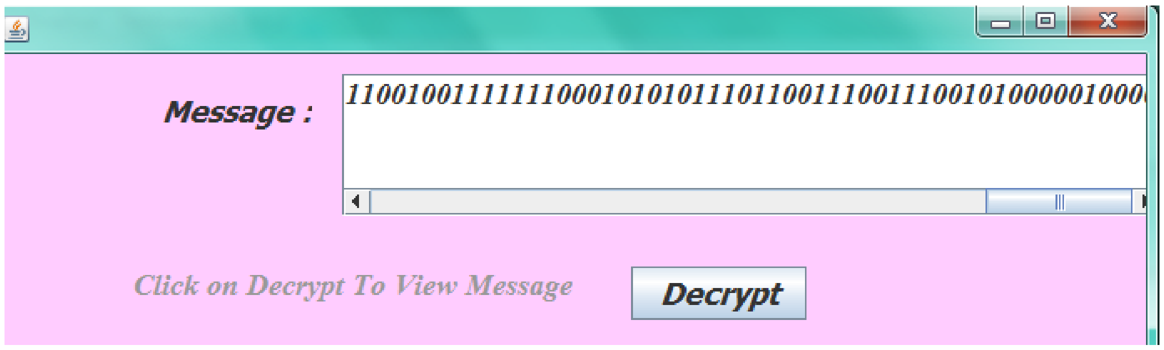
7. Inbox Page



Id	Subject	Sender	Date	Time	Text
1	Here is Your Mess...	vijetha@gmail.com	2017-02-22	12:02:52	011010100111111...
103	Here is your mess...	sushma@gmail.com	2017-02-22	13:23:34	011010100111111...

Fig 7.7: Inbox Page.

8. View Received Message

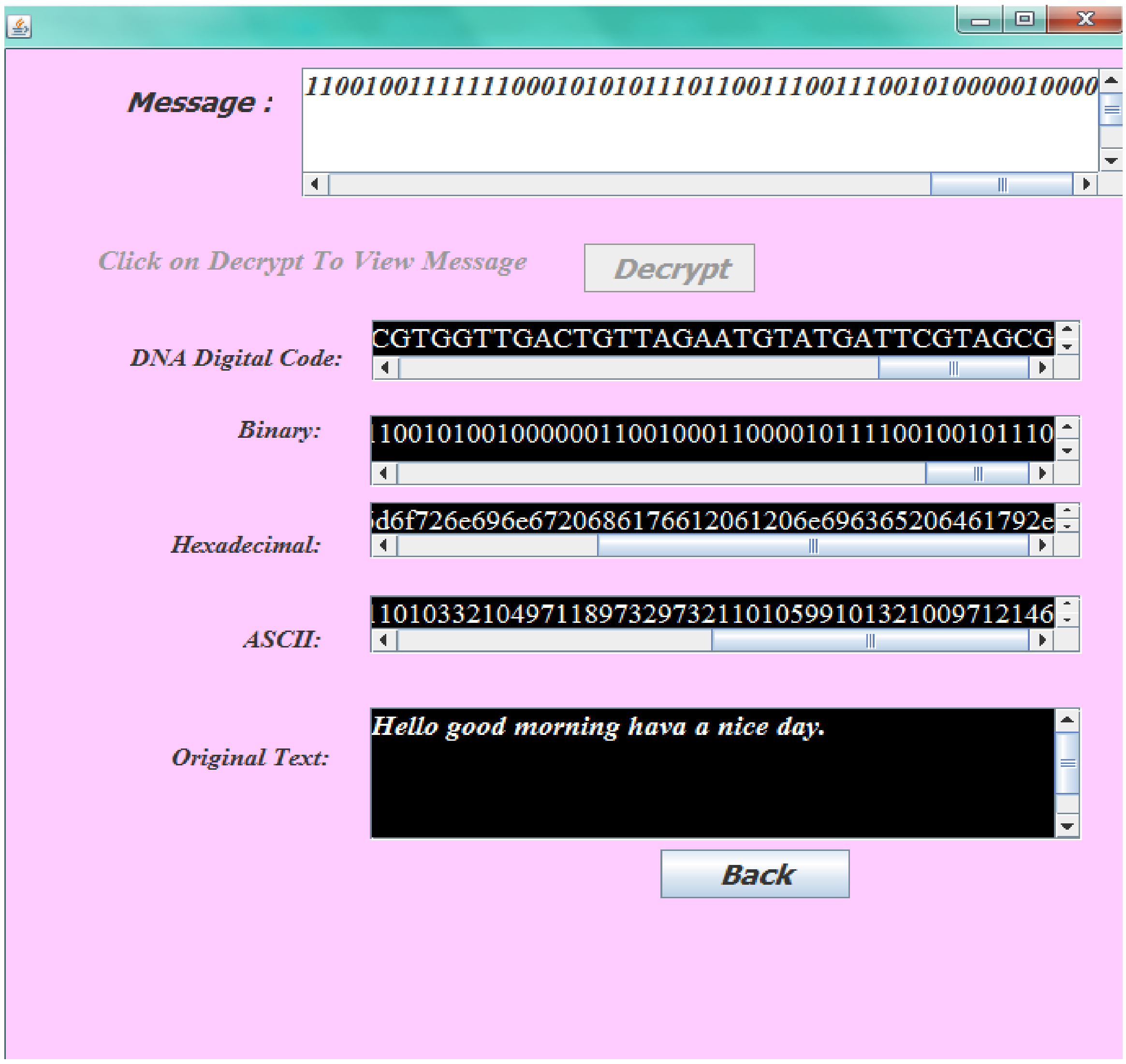


Message : 11001001111111000101010111011001110011100101000001000

Click on Decrypt To View Message

Fig 7.8: View Message

9. Decryption of Message



Message : 11001001111111000101010111011001110011100101000001000

Click on Decrypt To View Message

DNA Digital Code: CGTGGTTGACTGTTAGAATGTATGATTCGTAGCG

Binary: 1001010010000001100100011000010111100100101110

Hexadecimal: d6f726e696e6720686176612061206e696365206461792e

ASCII: 1010332104971189732973211010599101321009712146

Original Text: Hello good morning hava a nice day.

Fig 7.9: Decrypted Text.

Chapter 8

Conclusion

Data security is the main challenge for data storage. Various algorithms like RSA, Diffie-Hellman, DNA encryption etc. are available to provide data security for the data stored on servers. Digital signatures, Extensible Authentication Protocols are used for authentications. Using BDEA algorithm, we achieve 2-layer security for ASCII character sets. The proposed system focuses on extending the BDEA algorithm to be used with Unicode character set. This can help reach to the wider community of the cloud users. The future work will focus on the possible attacks and cryptanalysis of the cipher text and measure its strength.

REFERENCES

[1] PrashantRewagad, YogitaPawar, "Use of Digital Signature with Diffie-Hellman Key Exchange and AES Encryption Algorithm to Enhance Data Security in Cloud Computing" 2013 International Conference on Communication System and Network Technologies (IEEE Computer Society).

[2] Uma Somani, Kanika Lakhani, ManishaMundra, "Implementing Digital Signature with RSA Encryption Algorithm to Enhance the Data Security of Cloud in Cloud

Computing"-2010 IEEE 1st International Conference on Parallel, Distributed and Grid Computing (PDGC-2010).

[3] Mehdi Hojabri& Mona Heidari"Union of RSA algorithm, Digital Signature and KERBEROS in Cloud Computing" International Conference on Software Technology and Computer Engineering (STACE-2012).

[4] Ashish Prajapati, Amit Rathod "Enhancing security in cloud computing using Bi-Directional DNA Encryption Algorithm", International Conference on Intelligent Computing, Communication & Devices. (ICCD-2014), Springer.