

**A mini project report on**

**SECURING CLOUD DATA UNDER THE KEY EXPOSURE**

**Submitted in partial fulfilment of the requirements for the**

**award of degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**CSE (CYBER SECURITY)**

**By**

<b>SAMALA HARIKRISHNA</b>	<b>-225U1A6252</b>
<b>VASURI ANAND KUMAR</b>	<b>-225U1A6263</b>
<b>T.GUNA SHEKAR</b>	<b>-225U1A6257</b>
<b>MOHAMMED ABDUL ZABI</b>	<b>-225U1A6241</b>

**UNDER THE ESTEEMED GUIDANCE OF**

**Mrs.CH.Ithvika**

**M. Tech**

**(Asst.Professor)**



**DEPARTMENT OF CSE (CYBER SECURITY)**

**AVN INSTITUTE AND ENGINEERING AND TECHNOLOGY**

**(AFFILIATED TO JNTU UNIVERSITY, HYDERABAD)**

**PATELGUDA, KOHEDA ROAD, IBRAHIMPATNAM, 501510**

**2024-2025**



## **CERTIFICATE**

This is a certificate for a project work entitled "SECURING CLOUD DATA UNDER KEYEXPOSURE" submitted by **SAMALA HARIKRISHNA** (225U1A6252), **VASURI ANAND KUMAR** (225U1A6263), **T.GUNA SHEKER** (225U1A6257), and **MOHAMMED ABDUL ZABI** (225U1A6241) in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering (CYBER SECURITY) to the Jawaharlal Nehru Technological University. This certificate states that the project was carried out under the guidance and supervision during the academic year 2024-2025 and that the results have not been submitted to any other University or Institute for the award of degree or diploma.

### **Internal Guide**

**Mrs.CH.Ithvika**

**M. Tech**

**(Asst.Professor)**

### **Project Coordinator**

**Mr. P. SATHISH**

**M. Tech**

**(Asst. Professor)**

### **HOD**

**Dr. P. INDIRA PRIYADARSINI**

**M. Tech, Ph. D**

**(Professor)**

**External Examiner**

# ACKNOWLEDGEMENT

First of all, we would like to thank everyone who has guided us in successfully completing our project, **Securing Cloud Data Under The Key Exposure**

We would like to express our deep sense of gratitude to the **AVN Institute of Engineering Technology** for giving us the opportunity to take up this project work. We sincerely thank **Our Principal, Dr. P. NAGESWARA REDDY Sir**, for his administration, which provided us with a wonderful educational environment.

We are deeply grateful for the inspiring guidance, encouragement, and continuous support of **Dr. P. INDIRA PRIYADARSINI, Head of the Department of Computer Science and Engineering (Cyber Security)**. Her helpful suggestions and constant encouragement have been invaluable in completing this dissertation. It was a pleasure working under her alert, human, and technical supervision.

We also express our deep gratitude towards our internal guide, **Mrs. CH. ITHVIKA** , Asst.Professor, for his guidance, comments, and encouragement during the course of this work. We are equally thankful to the staff of the Computer Science and Engineering (Cyber Security) Department and our friends, who directly or indirectly helped us in completing this project work

# **DECLARATION**

We declare that the work reported in the entitled **SECURING CLOUD DATA UNDER THE KEY EXPOSURE** is a record of the work done by us in the Department of Computer Science and Engineering (Cyber Security), AVN Institute of Engineering and Technology, Hyderabad.

No part of the thesis is copied from books/journals/Internet and wherever referred, the same has been duly acknowledged in the text. The reported data are based on the project work done entirely by us and not copied from any other source

**SAMALA HARIKRISHNA (225U1A6252)**

**VASURI ANAND KUMAR (225U1A6263)**

**T GUNA SHEKAR REDDY (225U1A6257)**

**MOHAMMED ABDUL ZABI (225U1A6241)**

# Table of Contents

<b>Table of Contents</b>	<b>v</b>
<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>viii</b>
<b>Abstract</b>	<b>x</b>
<b>1 PREAMBLE</b>	<b>1</b>
1.1 INTRODUCTION.....	1
<b>2 LITERATURE SURVEY</b>	<b>3</b>
2.1 INTRODUCTION.....	3
2.2 LITERATURE SURVEY.....	3
2.3 PAPER 1.....	3
2.4 PAPER 2.....	4
2.5 PAPER 3.....	4
2.6 PAPER 4.....	4
2.7 PAPER 5.....	5
<b>3 METHODOLOGY</b>	<b>6</b>
3.1 INTRODUCTION.....	6
<b>4 SYSTEM REQUIREMENT SPECIFICATION</b>	<b>10</b>
4.1 INTRODUCTION.....	10
4.2 FUNCTIONAL REQUIREMENTS .....	11
4.3 NON-FUNCTIONAL REQUIREMENTS .....	11
4.4 HARDWARE REQUIREMENTS .....	14
4.5 SOFTWARE REQUIREMENTS.....	15
4.6 SOFTWARE QUALITY ATTRIBUTES .....	15
<b>5 SYSTEM ANALYSIS</b>	<b>16</b>
5.1 INTRODUCTION.....	16
5.2 FEASIBILITY STUDY .....	16
5.3 ECONOMICAL FEASIBILITY .....	17

5.4	TECHNICAL FEASIBILITY .....	17
5.5	SOCIAL FEASIBILITY .....	17
<b>6</b>	<b>SYSTEM DESIGN</b>	<b>18</b>
6.1	INTRODUCTION.....	18
6.2	SYSTEM DEVELOPMENT METHODOLOGY.....	18
6.3	DESIGN USING UML .....	20
6.4	DATA FLOW DIAGRAM.....	20
6.5	CLASS DIAGRAM.....	22
6.6	USE CASE DIAGRAM .....	23
6.7	ACTIVITY DIAGRAM .....	23
6.8	SEQUENCE DIAGRAM.....	24
<b>7</b>	<b>IMPLEMENTATION</b>	<b>29</b>
7.1	INTRODUCTION.....	29
7.2	FILE UPLOAD CODE .....	30
7.3	FILE SPLIT CODE.....	33
<b>8</b>	<b>TESTING AND RESULTS</b>	<b>39</b>
8.1	INTRODUCTION.....	39
8.2	TESTING METHODOLOGIES.....	39
8.3	TEST CASE 1.....	41
8.4	TEST CASE 2.....	41
8.5	TEST CASE 3.....	41
8.6	TEST CASE 4.....	42
8.7	TEST CASE 5.....	42
8.8	TEST CASE 6.....	42
8.9	IMAGE OF HOME PAGE .....	43
8.10	IMAGE OF ADMIN, OWNER AND USER LOGIN PAGE.....	43
8.11	IMAGE OF FILE UPLOAD .....	44
8.12	IMAGE OF FILE DOWNLOAD.....	44
<b>9</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	<b>45</b>
9.1	CONCLUSION .....	45
9.2	FUTURE SCOPE.....	45
	<b>References</b>	<b>46</b>

# List of Figures

3.1	Structure of cloud computing.....	6
3.2	Structure of cloud computing.....	8
3.3	Structure of service models.....	8
4.1	Software Quality Attributes .....	15
6.1	Waterfall Model .....	20
6.2	Data flow diagram for Owner login.....	22
6.3	Data flow diagram for User login.....	22
6.4	Data flow diagram for Cloud login .....	23
6.5	Class Diagram.....	24
6.6	Use Case Diagram.....	26
6.7	Activity Diagram .....	27
6.8	Sequence Diagram.....	28
8.1	Home page.....	43
8.2	Admin, Owner and User login page .....	43
8.3	File upload.....	44
8.4	File Download.....	44

# List of Tables

6.1	Symbols used in UML.....	21
8.1	Unit Test Cases .....	41



# Abstract

Recent news reveal a powerful attacker which breaks data confidentiality by acquiring cryptographic keys, by means of coercion or backdoors in cryptographic software. Once the encryption key is exposed, the only viable measure to preserve data confidentiality is to limit the attackers access to the ciphertext. This may be achieved, for example, by spreading ciphertext blocks across servers in multiple administrative domains thus assuming that the adversary cannot compromise all of them. Nevertheless, if data is encrypted with existing schemes, an adversary equipped with the encryption key, can still compromise a single server and decrypt the ciphertext blocks stored therein.

we study data confidentiality against an adversary which knows the encryption key and has access to a large fraction of the ciphertext blocks. To this end, we propose Bastion, a novel and efficient scheme that guarantees data confidentiality even if the encryption key is leaked and the adversary has access to almost all ciphertext blocks. We analyze the security of Bastion, and we evaluate its performance by means of a prototype implementation. We also discuss practical insights with respect to the integration of Bastion in commercial dispersed storage systems. Our evaluation results suggest that Bastion is well-suited for integration in existing systems since it incurs less than 5% overhead compared to existing semantically secure encryption mode

# 1. PREAMBLE

## 1.1 INTRODUCTION

THE world recently witnessed a massive surveillance program aimed at breaking users privacy. Perpetrators were not hindered by the various security measures deployed within the targeted services. For instance, although these services relied on encryption mechanisms to guarantee data confidentiality, the necessary keying material was acquired by means of backdoors, bribe, or coercion. If the encryption key is exposed, the only viable means to guarantee confidentiality is to limit the adversarys access to the ciphertext, e.g., by spreading it across multiple administrative domains, in the hope that the adversary cannot compromise all of them. However, even if the data is encrypted and dispersed across different administrative domains, an adversary equipped with the appropriate keying material can compromise a server in one domain and decrypt ciphertext blocks stored therein.

we study data confidentiality against an adversary which knows the encryption key and has access to a large fraction of the ciphertext blocks. The adversary can acquire the key either by exploiting flaws or backdoors in the key-generation software, or by compromising the devices that store the keys (e.g., at the user-side or in the cloud). As far as we are aware, this adversary invalidates the security of most cryptographic solutions, including those that protect encryption keys by means of secret-sharing (since these keys can be leaked as soon as they are generated). To counter such an adversary, we propose Bastion, a novel and efficient scheme which ensures that plaintext data cannot be recovered as long as the adversary has access to at most all but two ciphertext blocks, even when the encryption key is exposed. Bastion achieves this by combining the use of standard encryption functions with an efficient - nothing transform. An AONT is not an encryption by itself, but can be used as a pre-processing step before encrypting the data with a block cipher.

This encryption paradigm called AON encryption was mainly intended to slow down brute-force attacks on the encryption key. However, AON encryption can also preserve data confidentiality in case the encryption key is exposed, as long as the adversary has access to at most all but one ciphertext blocks. Existing AON encryption schemes, however, require at least two rounds of block cipher encryptions on the data: one preprocessing round to create the AONT, followed by another round for the actual encryption. Notice that these rounds are sequential, and cannot be parallelized. This results in considerableoften unacceptableoverhead to encrypt and decrypt large files. On the other hand, Bastion requires only one round of encryp-

deterioration (less than 5%) when compared to symmetric encryption schemes, and considerably improves the performance of existing AON encryption schemes. We also discuss practical insights with respect to the possible integration of Bastion in commercial dispersed storage systems. Our contributions in this paper can be summarized as follows:

- We propose Bastion, an efficient scheme which ensures data confidentiality against an adversary that knows the encryption key and has access to a large fraction of the ciphertext blocks.
- We analyze the security of Bastion, and we show that it prevents leakage of any plaintext block as long as the adversary has access to the encryption key and to all but two ciphertext blocks.
- We evaluate the performance of Bastion analytically and empirically in comparison to a number of existing encryption techniques. Our results show that Bastion considerably improves (by more than 50%) the performance of existing AON encryption schemes, and only incurs a negligible overhead when compared to existing semantically secure encryption modes (e.g., the CTR encryption mode).
- We discuss practical insights with respect to the deployment of Bastion within existing storage systems, such as the HYDRAstor grid storage system.

## 2. LITERATURE SURVEY

### 2.1 INTRODUCTION

Literature survey is mainly carried out in order to analyze the background of the current project which helps to find out flaws in the existing system and guides on which unsolved problems we can work out. So, the following topics not only illustrate the background of the project but also uncover the problems and flaws which motivated to propose solutions and work on this project.

### 2.2 LITERATURE SURVEY

Literature survey is the documentation of a comprehensive review of the published and unpublished work from secondary sources data in the areas of specific interest to the researcher. The library is a rich storage base for secondary data and researchers used to spend several weeks and sometimes months going through books, journals, newspapers, magazines, conference proceedings, doctoral dissertations, master's theses, government publications and financial reports to find information on their research topic. Reviewing the literature on the topic area at this time helps the researcher to focus further interviews more meaningfully on certain aspects found to be important is the published studies even if these had not surfaced during the earlier questioning. So the literature survey is important for gathering the secondary data for the research which might be proved very helpful in the research. The literature survey can be conducted for several reasons

### 2.3 PAPER 1

**Title:** Secret-Sharing Schemes: A Survey.

**Context:**

A secret-sharing scheme is a method by which a dealer distributes shares to parties such that only authorized subsets of parties can reconstruct the secret. Secret-sharing many secure protocols, e.g., general protocol for multiparty computation, Byzantine. In this survey, we will describe the most important constructions of secret-sharing schemes, explaining the connections between secret-sharing schemes and monotone formulae and monotone span programs. The main problem with known secret-sharing schemes is the large share size: it is exponential in the number of parties..

## 2.4 PAPER 2

**Title:**Using Erasure Codes Efficiently for Storage in a Distributed System.

**Context:**

Erasure codes provide space-optimal data redundancy to protect against data loss. A common use is to reliably store data in a distributed system, where erasure-coded data are kept in different nodes to tolerate node failures without losing data. In this paper, we propose a new approach to maintain ensure-encoded data in a distributed system. The approach allows the use of space efficient k-of-n erasure codes where n and k are large and the overhead n-k is small. Concurrent updates and accesses to data are highly optimized: in common cases, they require no locks, no two-phase commits, and no logs of old versions of data. We evaluate our approach using an implementation and simulations for larger systems.

## 2.5 PAPER 3

**Title:**Security amplification by composition: The case of doublyiterated, ideal ciphers.

**Context:**

One concern in using cloud storage is that the sensitive data should be confidential. We investigate, in the Shannon model, the security of constructions corresponding to double and (two-key) triple DES. That is, we consider  $F_{k1} \circ F_{k2}()$  and  $F_{k1} \circ F_{k1} \circ F_{k2} \circ F_{k1}()$  with the component functions being ideal ciphers. This models the sense. We compute a bound on the probability of breaking the double cipher as a function of the number of computations of the base cipher made, and the number of examples of the composed cipher seen, and show that the success probability is the

## 2.6 PAPER 4

**Title:**The security of all-or-nothing encryption: Protecting against exhaustive key search.

**Context:**

We investigate the all-or-nothing encryption paradigm which was introduced by Rivest as a new mode of operation for block ciphers. The paradigm involves composing an

have secure encryption modes with the additional property that exhaustive key-search attacks on them are slowed down by a factor equal to the number of blocks in the ciphertext. We give a new notion concerned with the privacy of keys that provably captures this key-search resistance property. We suggest a new characterization of AONTs and establish that the resulting all-or-nothing encryption paradigm yields secure encryption modes that also meet this notion of key privacy. .

## 2.7 PAPER 5

**Title:** Deniable encryption with negligible detection probability.

**Context:**

Deniable encryption, introduced in 1997 by Canetti, Dwork, Naor, and Ostrovsky, guarantees that the sender or the receiver of a secret message is able to fake the message encrypted in a specific ciphertext in the presence of a coercing adversary, without the adversary detecting that he was not given the real message. To date, constructions are only known either for weakened variants with separate honest and dishonest encryption algorithms, or for single-algorithm schemes with non-negligible detection probability. We propose the first sender-deniable public key encryption system with a single encryption algorithm and negligible detection probability.

# 3. METHODOLOGY

## 3.1 INTRODUCTION

### What is cloud computing?

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure user's data, software and computation. Cloud computing consists of hardware and software resources made available on the Internet as managed third-party services. These services typically provide access to advanced software applications and high-end networks of server computers.

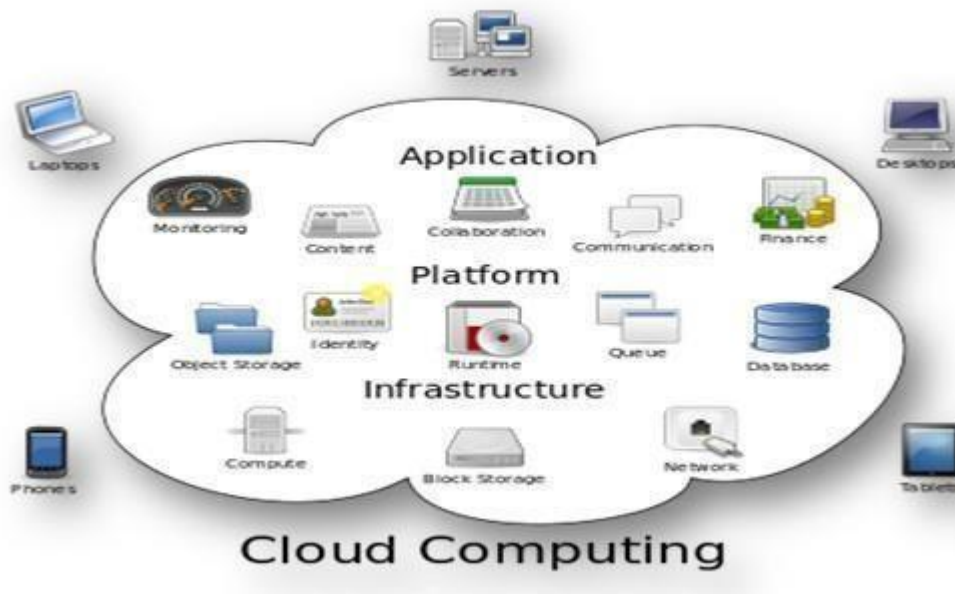


Figure 3.1: Structure of cloud computing

### How cloud computing works?

The cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. This shared IT infrastructure contains large pools of systems that are linked together. Often, virtualization techniques are used to maximize the power of cloud computing.

without requiring human interaction with each services provider.

- **Broad network access:** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).
- **Resource pooling:** The providers computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location-independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.
- **Rapid elasticity:** Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.
- **Measured service:** Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be managed, controlled, and reported providing transparency for both the provider and consumer of the utilized service.
- **Services Models:**  
Cloud Computing comprises three different service models, namely Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). The three service models or layer are completed by an end user layer that encapsulates the end user perspective on cloud services. The model is shown in figure below. If a cloud user accesses services on the infrastructure layer, for instance, she can run her own applications on the resources of a cloud infrastructure and remain responsible for the support, maintenance, and security of these applications herself. If she accesses a service on the application layer, these tasks are normally taken care of by the cloud service provider.

#### **Benefits of cloud computing:**

1. **Achieve economies of scale** - increase volume output or productivity with fewer people. Your cost per unit, project or product plummets. The numbers starts at 1 with every call to the enumerate environment.





Figure 3.2: Structure of cloud computing

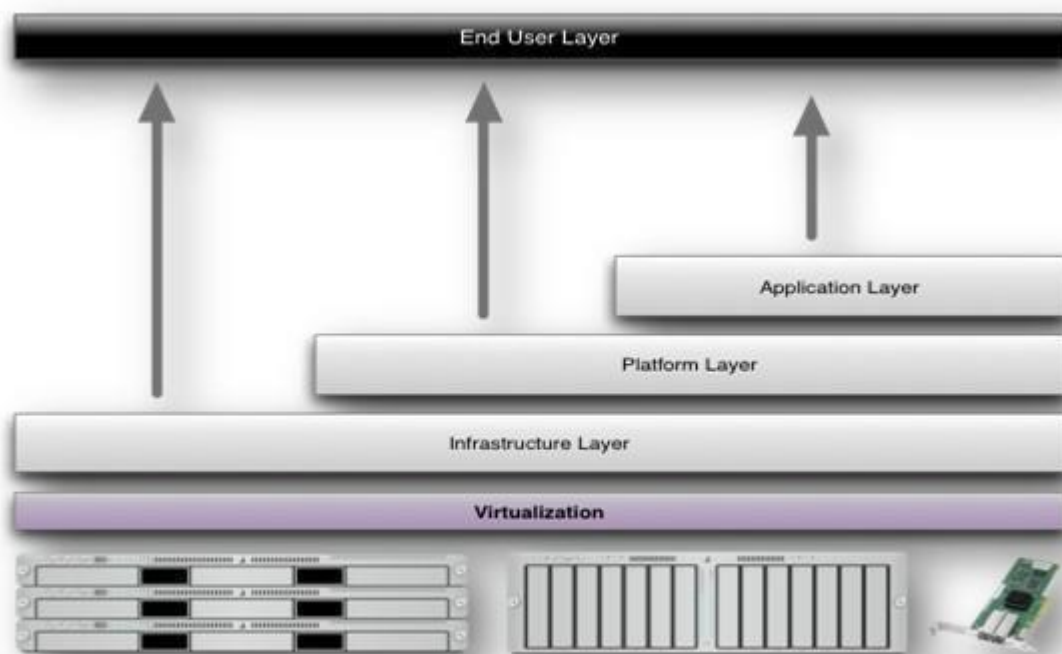


Figure 3.3: Structure of service models

2. **Reduce spending on technology infrastructure-** Maintain easy access to your information with minimal upfront spending. Pay as you go (weekly, quarterly or yearly), based on demand.
3. **Globalize your workforce on the cheap.** People worldwide can access the cloud, provided they have an Internet connection.
4. **Streamline processes.** Get more work done in less time with less people.
5. **Reduce capital costs.** There's no need to spend big money on hardware,

software or licensing fees.

6. **Improve accessibility.** You have access anytime, anywhere, making your life so much easier!
7. **Monitor projects more effectively.** Stay within budget and ahead of completion cycle times.
8. **Less personnel training is needed.** It takes fewer people to do more work on a cloud, with a minimal learning curve on hardware and software issues.
9. **Minimize licensing new software.** Stretch and grow without the need to buy expensive software licenses or programs.
10. **Improve flexibility.** You can change direction without serious people or financial issues at stake.

#### **Advantages:**

1. **Price.** Pay for only the resources used.
2. **Security.** Cloud instances are isolated in the network from other instances for improved security.
3. **Performance.** Instances can be added instantly for improved performance. Clients have access to the total resources of the Clouds core hardware.
4. **Scalability.** Auto-deploy cloud instances when needed.
5. **Uptime.** Uses multiple servers for maximum redundancies. In case of server failure, instances can be automatically created on another server.
6. **Control.** Able to login from any location. Server snapshot and a software library lets you deploy custom instances.
7. **Traffic.** Deals with spike in traffic with quick deployment of additional instances to handle the load.

## **4. SYSTEM REQUIREMENT SPECIFICATION**

### **4.1 INTRODUCTION**

This chapter describes about the requirements. It specifies the hardware and software requirements that are required in order to run the application properly. The Software Requirement Specification (SRS) is explained in detail, which includes overview of dissertation as well as the functional and non-functional requirement of this dissertation.

A SRS document describes all data, functional and behavioral requirements of the software under production or development. SRS is a fundamental document, which forms the foundation of the software development process. Its the complete description of the behavior of a system to be developed. It not only lists the requirements of a system but also has a description of its major feature. Requirement Analysis in system engineering and software engineering encompasses those tasks that go into determining the need or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. Requirement Analysis is critical to the success to a development project. Requirement must be documented, measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

The SRS functions as a blueprint for completing a project. 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 plans, and documentation plans, are related to it. It is important to note that an SRS contains functional and non-functional requirements only.

Thus the goal of preparing the SRS document is to

- To facilitate communication between the customer, analyst, system developers, maintainers.
- To serve as a contrast between purchaser and supplier.
- To firm foundation for the design phase.

## 4.2 FUNCTIONAL REQUIREMENTS

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 functional requirements:-

- Input test case must not have compilation and runtime errors.
- The application must not stop working when kept running for even a long time.
- The application must function as expected for every set of test cases provided.
- The application should generate the output for given input test case and input parameters.
- The application should generate on-demand services.

## 4.3 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that 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 systems or because of external factors such as:-

- Product Requirements
- Organizational Requirements
- User Requirements
- Basic Operational Requirements

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. This should be contrasted with functional requirements that define specific behaviour or functions. The plan for implementing requirements define what a system is supposed to do and non- functional requirements define how a system is supposed to be. Functional requirements are usually in the form of system shall do requirement, an individual action of part of the system, \_

perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or IPO Model. In contrast, non-functional requirements are in the form of system shall be requirement, an overall property of the system as a whole or of a particular aspect and not a specific function. The systems' overall properties commonly mark the difference between whether the development project has succeeded or failed. Non-functional requirements of our project include:

- Response time The time the system takes to load and the time for responses on any action the user does.
- Processing time - How long is acceptable to perform key functions or export / import data?
- Throughput The number of transactions the system needs to handle must be kept in mind.
- Storage - The amount of data to be stored for the system to function.
- Growth Requirements As the system grows it will need more storage space to keep up with the efficiency.
- Locations of operation - Geographic location, connection requirements and the restrictions of a local network prevail.
- Architectural Standards The standards needed for the system to work and sustain.

#### **4.3.1 PRODUCT REQUIREMENTS**

- Portability: Since the SLR system is designed to run using Arduino (whose library is written in C), the system is portable.
- Correctness: It follows 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 interfaces 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 utmost relevancy and correctness.

whereas evolution quality involves testability, maintainability, extensibility or scalability.

#### **4.3.1.1 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 stages in the software engineering process. This stage is the first step in 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.

#### **4.3.1.2 USER REQUIREMENTS**

The user requirements document (URD) or user requirements specification is a document usually used to software engineering that specifies the requirements the user expects from software to be constructed in a software project. Once the required information is completely gathered it is documented in a URD, which is meant to spell out exactly what the software must do and becomes part of the contractual agreement. A customer cannot demand feature not in the URD, whilst the developer cannot claim the product is ready if it does not meet an item of the URD. The URD can be used as features are described. Formulating a URD requires negotiation to determine what is technically and economically feasible. Preparing a URD is one of those skills that lies between a science and economically feasible. Preparing a URD is one of those skills that lies between a science and an art, requiring both software technical skills and interpersonal skills.

#### **4.3.1.3 BASIC OPERATIONAL REQUIREMENTS**

Operational requirement is the process of linking strategic goals and objectives to tactic goals and objectives. It describes milestones, conditions for success and explains how, or what portion of, a strategic plan will be put into operation during a given operational period, in the case of, a strategic plan will be put into operation during a given operational period, in the case of commercial application, a fiscal year or another given budgetary term. An operational plan is the basis for, and justification

of an annual operating budget request. Therefore, a five-year strategic plan would typically require five operational plans funded by five operating budgets. Operational plans should establish the activities and budgets for each part of the organization for the next 1-3 years. They link the strategic plan with the activities the organization will deliver and the resources required to deliver them. An operational plan draws directly from agency and program strategic plans to describe agency and program missions and goals, program objectives, and program activities. Like a strategic plan, an operational plan addresses four questions:

- Where are we now?
- Where do we want to be?
- How do we get there?

The customers are those that perform the eight primary functions 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.

## **4.4 HARDWARE REQUIREMENTS**

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Monitor : 15 VGA Colour.
- Ram : 512 Mb.

## 4.5 SOFTWARE REQUIREMENTS

- Operating System : Windows XP/7
- Coding Language : JAVA/J2EE.
- Data Base: MYSQL

## 4.6 SOFTWARE QUALITY ATTRIBUTES

- **Functionality:** the capability of the software to provide functions which meet stated and implied needs when the software is used under specified conditions.
- **Reliability:** the capability of the software to maintain its level of performance.
- **Usability:** the capability of the software to be understood, learned, used and liked by the user, when used under specified conditions.
- **Efficiency:** the capability of the software to provide the required performance, relative to the amount of resources used, under stated conditions.
- **Maintainability:** the capability of the software to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.

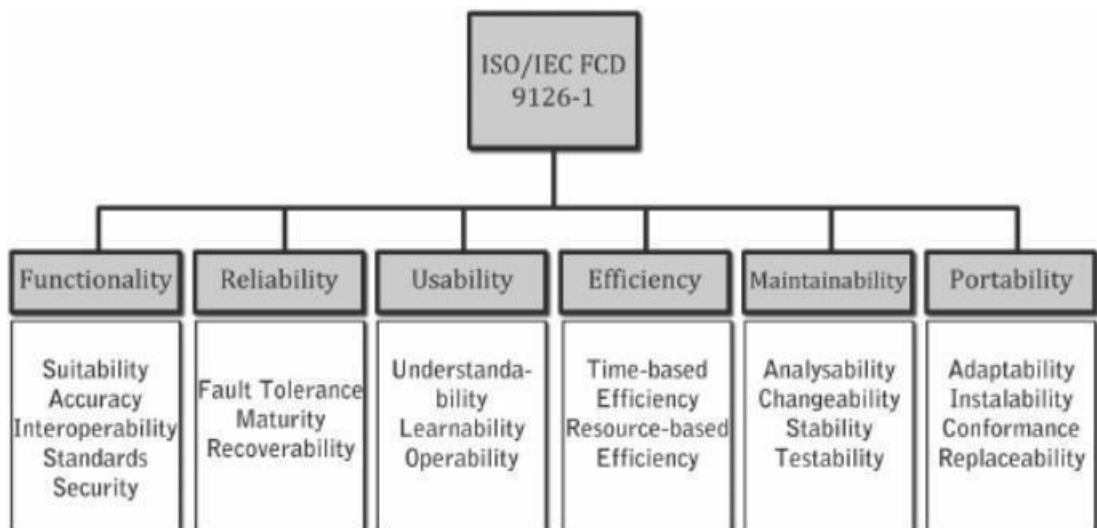


Figure 4.1: Software Quality Attribute



## **5. SYSTEM ANALYSIS**

### **5.1 INTRODUCTION**

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. Design is a place where design is fostered in software Engineering. Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. Design is the perfect way to accurately translate a customers requirement in the finished software product. Design creates a representation or model, provides details about software data structure, architecture, interfaces and components that are necessary to implement a system. The logical system design arrived at as a result of systems analysis is converted into physical system design.

### **5.2 FEASIBILITY STUDY**

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

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

### **5.3 TECHNICAL FEASIBILITY**

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

### **5.4 SOCIAL FEASIBILITY**

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

## 6. SYSTEM DESIGN

### 6.1 INTRODUCTION

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. Design is a place where design is fostered in software Engineering. Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. Design is the perfect way sign creates a representation or model, provides details about software data structure, architecture, interfaces and components that are necessary to implement a system. The logical system design arrived at as a result of systems analysis is converted into physical system design.

### 6.2 SYSTEM DEVELOPMENT METHODOLOGY

System development method is a process through which a product will get completed 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.

#### 6.2.1 MODEL PHASES

The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Requirement initiation, Analysis, Design, Implementation, Testing and maintenance.

- **Requirement Analysis:** This phase is concerned about collection of requirement of the system. This process involves generating document and requirement review.
- **System Design:** Keeping the requirements in mind the system specifications are translated in to a software representation. In this phase the designer emphasizes on:-algorithm, data structure, software architecture etc.
- **Coding:** In this phase programmer starts his coding in order to give a full

- **Implementation:** The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executables, user manuals and additional software documentation
- **Testing:** In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.
- **Maintenance:** The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

### **6.2.2 REASON FOR CHOOSING WATERFALL MODEL AS DEVELOPMENT METHOD**

- Clear project objectives.
- Stable project requirements.
- Progress of system is measurable.
- Strict sign-off requirements.
- Helps you to be perfect.
- Logic of software development is clearly understood.
- Production of a formal specification
- Better resource allocation.
- Improves quality. The emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage.
- Less human resources required as once one phase is finished those people can start working on to the next phase.

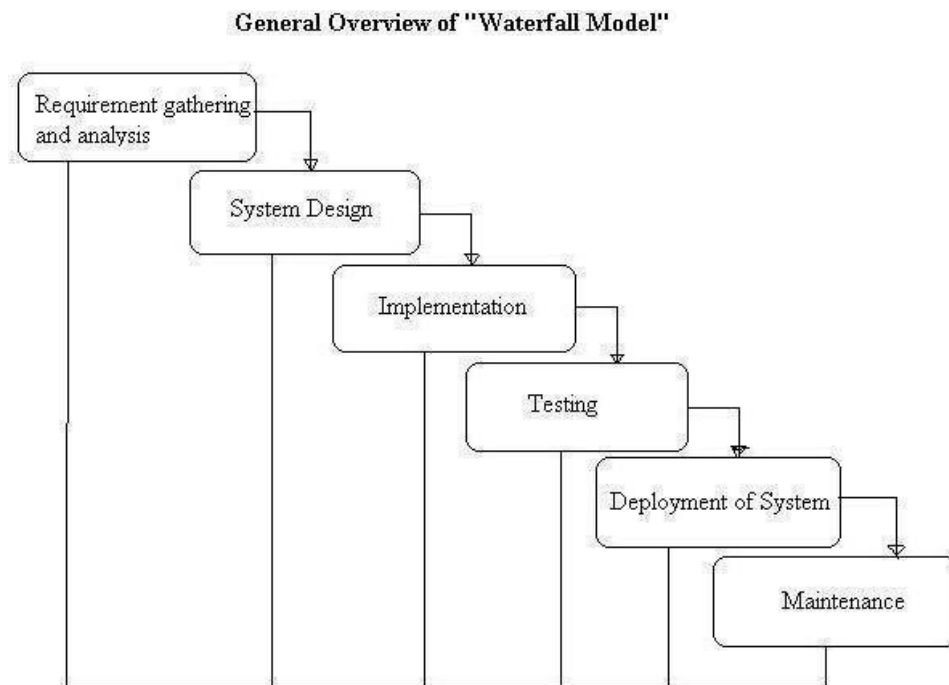


Figure 6.1: Waterfall Model

## 6.3 DESIGN USING UML

Designing UML diagram specifies, how the process within the system communicates along with how the objects within the process collaborate using both static as well as dynamic development, it has been getting harder and harder to develop and manage high quality applications in reasonable amount of time. As a result of this challenge and the need for a universal object modeling language every one could use, the Unified Modeling Language (UML) is the Information industries version of blue print. It is a method for describing the systems architecture in detail. Easier to build or maintain system, and to ensure that the system will hold up to the requirement changes.

## 6.4 DATA FLOW DIAGRAM

A data flow diagram (DFD) is graphic representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data level DFD first which shows the interaction between the system and outside entities. DFDs show the flow of data from external entities into the system, how the data moves from one process to another, as well as its logical storage. There are only four





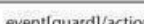





<i>Line</i>	<i>Symbol</i>
Association	<u>AssociationName</u>
Aggregation	
Generalization	
Dependency	
Activity edge	
Event, transition	event[guard]/action 
Link	<u>linkName</u>
Composition	
Realization	
Assembly connection	
Message	some code 
Control flow	

Table 6.1: Symbols used in UML

symbols: 1. Squares representing external entities, which are sources and destinations of information entering and leaving the system. 2. Rounded rectangles representing processes, in other methodologies, may be called 'Activities', 'Actions', 'Procedures', 'Subsystems' etc. which take data as input, do processing to it, and output it. 3. Arrows representing the data flows, which can either, be electronic data or physical items. It is impossible for data to flow from data store to data store except via a process, and external entities are not allowed to access data stores directly. 4. The flat three-sided rectangle is representing data stores should both receive information for storing and provide it for further processing. Figure 6.2, Figure 6.3 and Figure 6.4 show data flow diagrams.

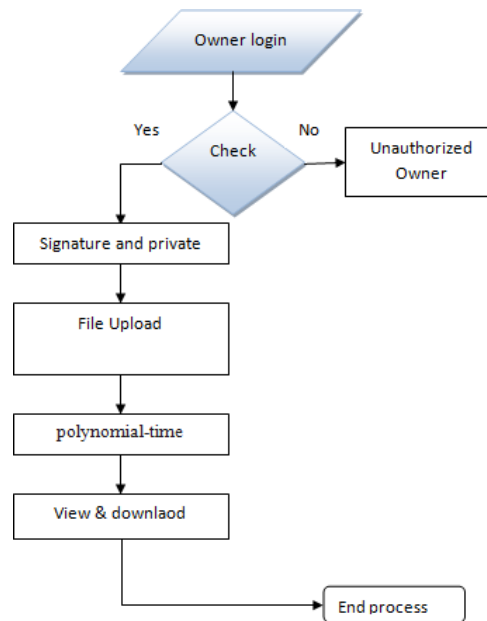


Figure 6.2: Data flow diagram for Owner login

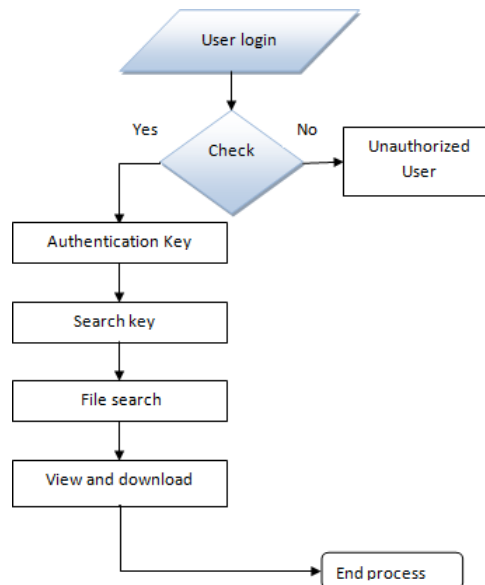


Figure 6.3: Data flow diagram for User login

## 6.5 CLASS DIAGRAM

UML class diagram shows the static structure of the model. The class diagram is a collection of static modeling elements, such as classes and their relationships, connected as a graph to each other and to their contents. The class diagram is the main building the systematic of the application, and for detailed modeling translating the models

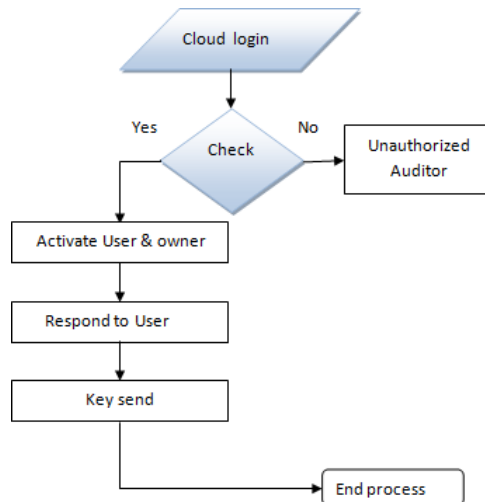


Figure 6.4: Data flow diagram for Cloud login

into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed. Figure 6.5 show class diagram.

## 6.6 USE CASE DIAGRAM

A use case defines a goal-oriented set of interactions between external entities and the system under consideration. The external entities which interact with the system are its actors. A set of use cases describe the complete functionality of the system at a particular level of detail and it can be graphically denoted by the use case diagram. Figure 6.6 shows the use case diagram.

## 6.7 ACTIVITY DIAGRAM

An activity diagram shows the sequence of steps that make up a complex process. An activity is shown as a round box containing the name of the operation. An outgoing solid arrow attached to the end of the activity symbol indicates a transition triggered by the completion.

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.

Activity is a particular operation of the system. Activity diagrams are not only used



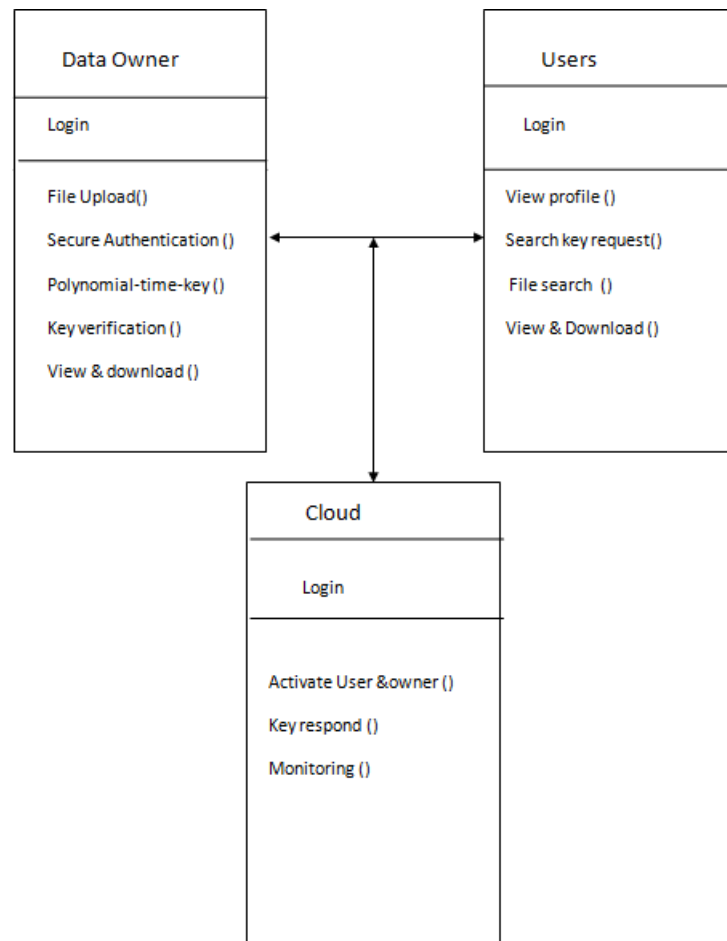


Figure 6.5: Class Diagram

the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single. Figure 6.7 shows activity diagram.

## 6.8 SEQUENCE DIAGRAM

Sequence diagram are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and the environment. A sequence diagram shows an interaction arranged in a time sequence. A sequence diagram has two dimensions: vertical dimension represents time, the horizontal dimension represents the objects existence during the interaction. **Basic elements:**

- **Vertical rectangle:** Represent the object is active (method is being performed).
- **Vertical dashed line:** Represent the life of the object.
- **X:** represent the life end of an object. (Being destroyed from memory)
- **Horizontal line with arrows:** Messages from one object to another.

Figure 6.8 shows sequence diagram.

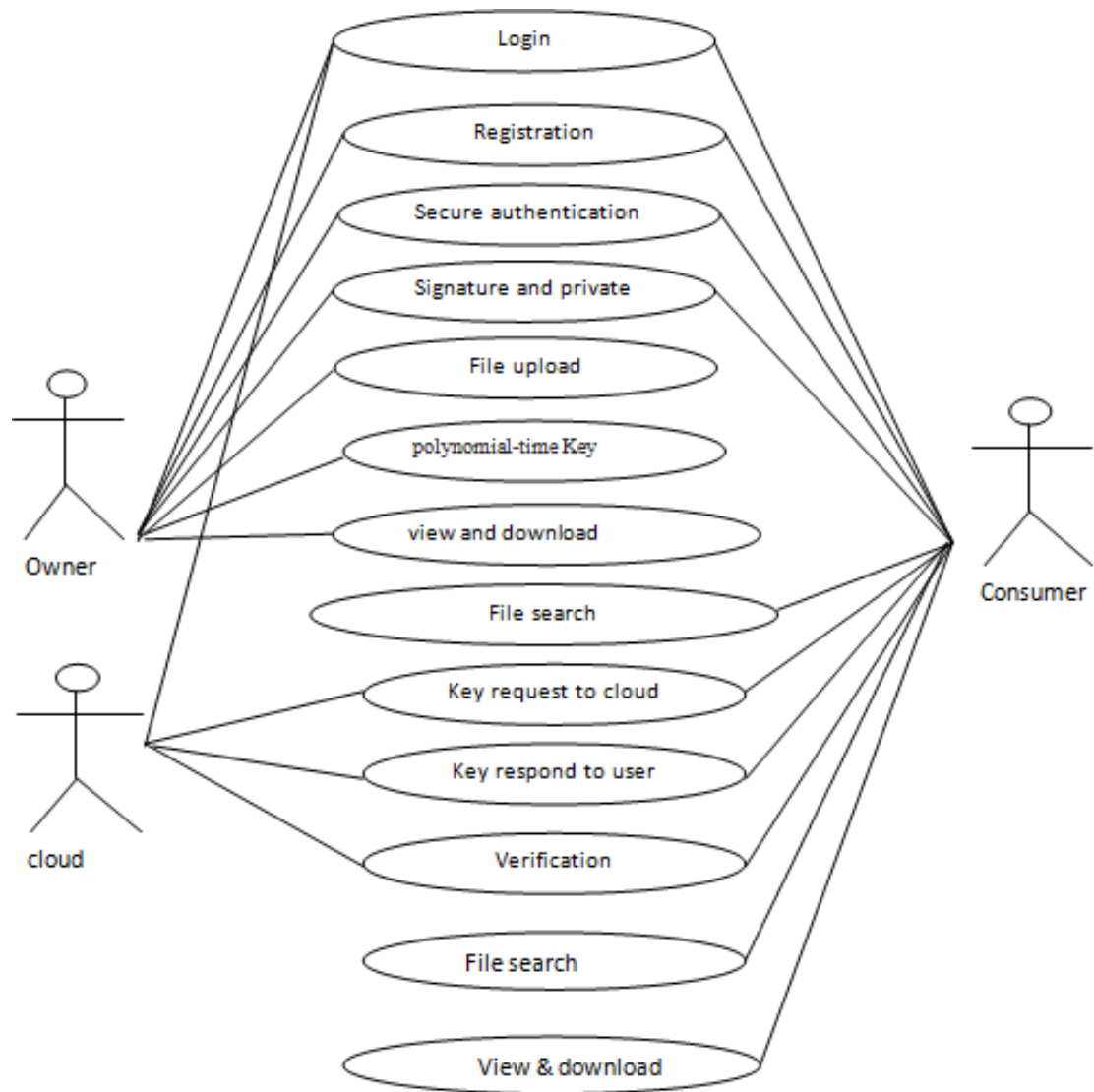


Figure 6.6: Use Case Diagram

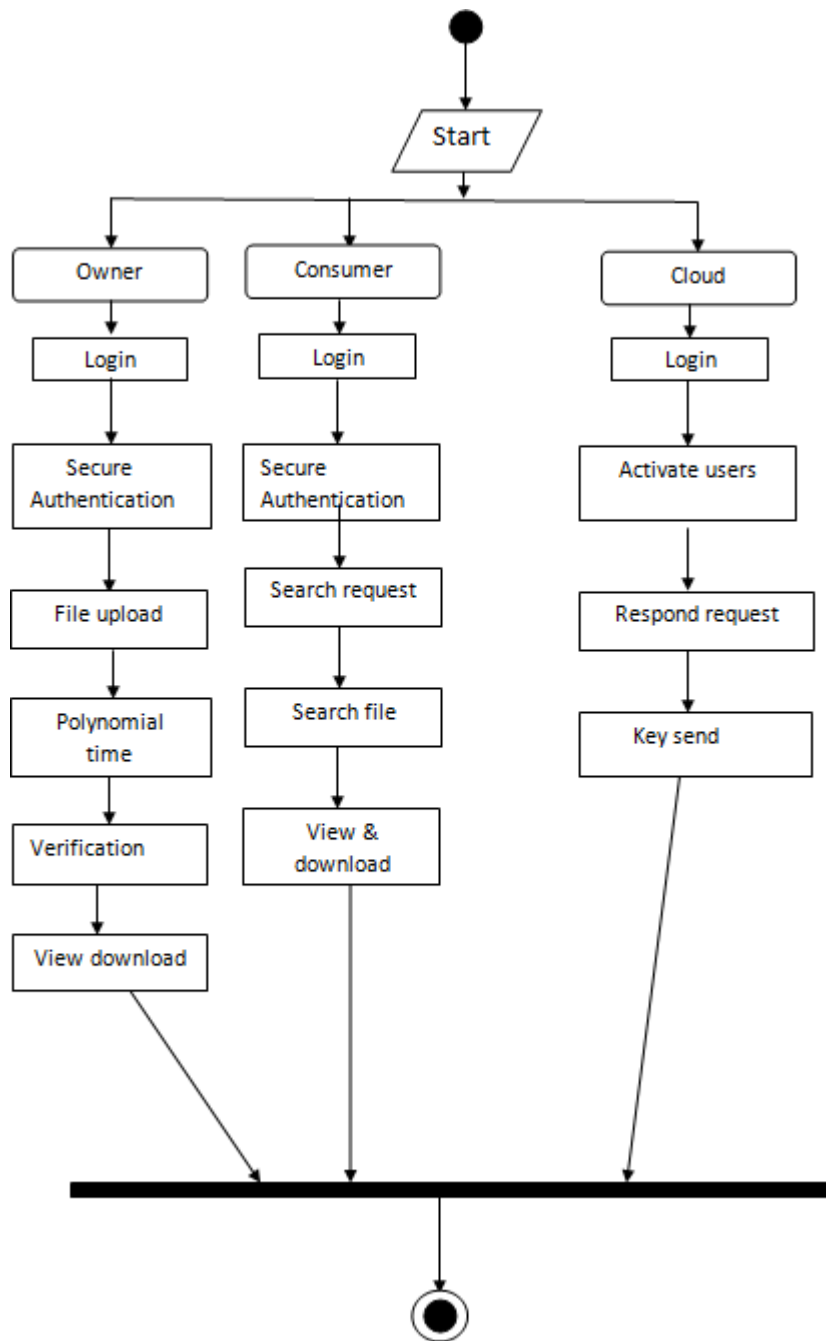


Figure 6.7: Activity Diagram

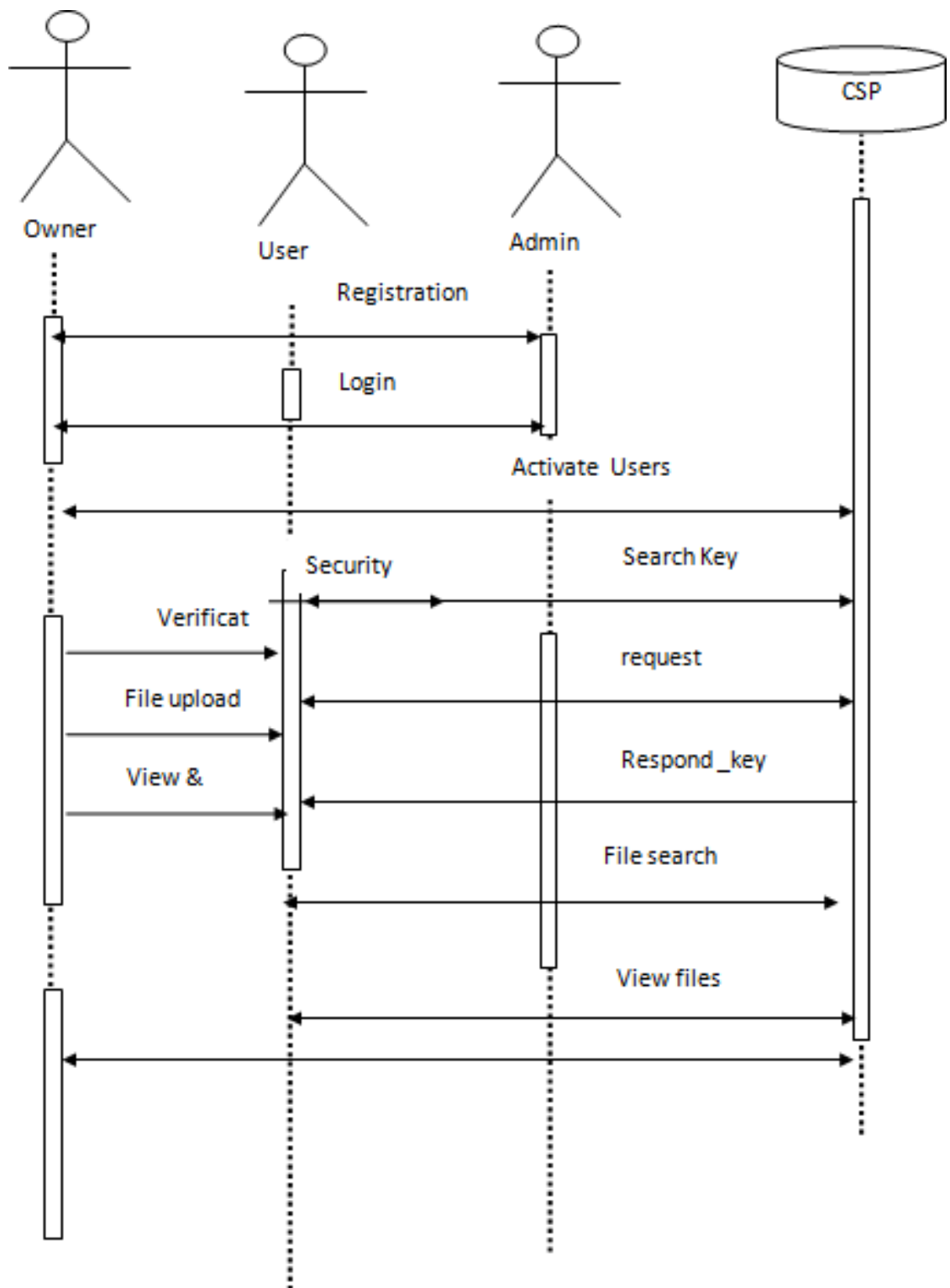


Figure 6.8: Sequence Diagram

# **7. IMPLEMENTATION**

## **7.1 INTRODUCTION**

The implementation phase of the project is where the detailed design is actually transformed into working code. Aim of the phase is to translate the design into a best possible solution in a suitable programming language. This chapter covers the implementation aspects of the project, giving details of the programming language and development environment used. It also gives an overview of the core modules of the project with their step by step flow. The implementation stage requires the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.
- Evaluation of the changeover method.
- Correct decisions regarding selection of the platform.
- Appropriate selection of the language for application development.

## 7.2 FILE UPLOAD CODE

```
package com.db.Connection ;
import com.oreilly.servlet.Multipart Request ;
import java.io.Buffered Reader ;
import java.io.File ;
import java.io.File Reader ;
import java.io.File Write r ;
import java.io.IOException ;
import java.io.InputStream ;
import java.io.InputStream Reader ;
import java.sql.Connection ;
import java.sql.DriverManager ;
import java.sql.Prepared Statement ;
import java.sql.SQLException ;
import java.text.SimpleDateFormat ;
import java.util.Calendar ;
import java.util.logging.Level ;
import java.util.logging.Logger ;
import javax.servlet.Servle tExce ptio n ;
import javax.servlet.annotation .Multipart Config ;
import javax.servlet.annotation .WebServlet ;
import javax.servlet.http .Http Servlet ;
import javax.s e r v l e t . http . Http Servlet Request ;
import javax.s e r v l e t . http . Http Servlet Response ;
import javax.servlet.http .Http Session ;
import javax.servlet.http .Part ;

@WebServlet ("/fileuploads")
@ Multipart Config ( max File Size = 16177215 )           //upload   file's
    size up to 16MB

public class fileuploades extends HttpServlet {
    File file ;
    final String filepath = "E:/";
    // database connection settings
    private String dbURL = "jdbc : mysql :// localhost :3306
```

```

private String dbPass = "root ";

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

    try {
        Calendar cal = Calendar .get Instance () ;
        Multipart Request m = new Multipart Request ( request ,
            filepath );
        File file = m.getFile ( "filess" );
        String filenames = file .getName () .toLowerCase () ;
        SimpleDateFormat format = new SimpleDateFormat ( "
            HH:mm_dd/MM/ yyyy" ) ;
        String date= format . format ( cal .getTime () ) ;
        HttpSession ses = request .getSession () ;
        String ownername = ses .get Attribute ( "oname" ) .
            toString () ;
        String privakey=request .get Parameter ( "privakey" ) ;
        String filename=request .get Parameter ( "filename" );
        String caption=request .get Parameter ( "caption" );
        Connection conn = null ;
        String message = null ;
        DriverManager .register Driver ( new com .mysql .jdbc . Driver
            () );
        conn = DriverManager . get Connection (dbURL, dbUser ,
            dbPass );

        InputStream inputStream = null ;
        Part filePart = request .get Part ( "filess" );
        if (filePart != null ) {
            System .out .println (filePart .getName () );
            System .out .println (filePart .get Siz e () );
            System .out .println (filePart .getContentType () );

            inputStream = filePart .get Input Stream () ;

```



```

try {
Buffered Reader br = null ;
StringBu ild er sb = new Str in g Bu ild er () ;

String line;
br = new Buffered Reader (new File Reader (
    file p a t h+file name )) ;
while (( line = br . read Line () ) != null ) {
    sb . append ( line ) ;
}
String content=sb . to Str in g () ;
String connn=content ;
System.out.println ( "*****" +sb.
    to Str in g () ) ;
    //storing encrypted file
File Write r fw=new File Write r ( file ) ;
fw . write ( connn ) ;
fw . close () ;
System . out . println ( " string _ Buffer "+connn ) ;
    boolean sta tu s=new Ftpcon () . upload ( file ) ;
if ( sta tu s ){
String sql = "INSERT INTO file upload ( privateky ,
    filename , f captio n , file s s , dates , owname ) value s _
    ( ? , ? , ? , ? , ? , ? ) " ;
Prepared Statement statement = conn . prepare Statement (
    sql ) ;
statement . set String ( 1 , priva key ) ;
statement . set String ( 2 , file name ) ;
statement . set String ( 3 , caption ) ;
statement . set String ( 5 , date ) ;
statement . set String ( 6 , owname ) ;

if ( inputStream != null ) {
    statement . set Blob ( 4 , inputStream ) ;
}
int row = statement . execute Update () ;
if ( row > 0 ) {

```

```

        response . send Redirect ( " owfileupload . jsp ? fmsg =
            Success " );

        message = " _ Regia tra tio n _ saved _ in to _ database _ ";
    }

    else {
        response . send Redirect ( " owfileupload . jsp ? fmsg =
            Faile d " );
        message = " _ Faile d _ saved _ in to _ database _ ";

    }
}

} catch ( SQLException ex ) {
    ex . print Stack Trace ( ) ;

}

} catch ( SQLException ex ) {
    Logger . get Logger ( file uploads . class . getName ( ) ) .
        lo g ( Level . SEVERE , null , ex );

}

}

}

```

## 7.3 FILE SPLIT CODE

```

/*
 * To change this template , choose Tools | Templates
 * and open the template in the editor.
 */
package com . db . Connection ;
import java . io . Buffered Reader ;
import java . io . File ;
import java . io . File Reader ;
import java . io . File Write r ;
import java . io . IOException ;
import java . io . Print Writer ;
import java . sql . Connection ;
import java . sql . Result Set ;
import java . sql . Statement ;
import java . text . DateFormat ;

```

```

import java . text . SimpleDateFormat ;
import java . util . Date ;
import javax . servlet . ServletException ;
import javax . servlet . http . HttpServlet ;
import javax . servlet . http . HttpServletRequest ;
import javax . servlet . http . HttpServletResponse ;
import javax . servlet . http . HttpSession ;

import com . oreilly . servlet . Multipart Request ;

/**
 *
 * @author java 4
 */
public class fileupload extends HttpServlet {
    File file;
    final String filepath="D:/";

    protected void process Request ( HttpServletRequest request ,
        HttpServletResponse response )

        throws ServletException , IOException {
        response . setContentType ( " text / html ; charset=UTF-8" );
        PrintWriter out = response . get Writer () ;
        try {
            HttpSession session= request . getSession () ;
            String ownername= ( String ) session . get Attribute
                ( "ownername" );
            Multipart Request m=new Multipart Request ( request ,
                filepath );
            File file=m . getFile ( " filess" );
            String filenames=file . getName () . toLowerCase () ;
            System . out . println ( " File _name"+filenames );
            String privatekey=m . get Parameter ( " privatekey " );
            String filename=m . get Parameter ( " file name " );
            String caption=m . get Parameter ( " caption " );
            Connection con= DbConnection . get Connection () ;
            Statement st3=con . create Statement () ;

```

```

Result Set rt3=st3.execute Query ("select * from
fileupload where file name ='"+file name+" '");
if(rt3.next () ){
    response .send Redirect ("owfileupload.jsp?
failed='yes '");
}
else {
//out.println("file location:"+file path+"\n file
name:"+file name+"\n");

Buffered Reader br=new Buffered Reader (new
File Reader ( file path+file names ));
StringBu ffe r sb=new StringBu ffe r ();
Strin g temp=null;

while (( temp=br . read Line () ) != null ){
    sb . append ( temp );
}

    System.out.println ("string Buffer"+sb);
//
    RS_IBE e=new RS_IBE();
//
    String IBEE=e.encrypt (sb.to String ());

Strin g ftest=(sb.to Strin g ());

Strin g IBE = ftest;
as syme tric assy = new as syme tric ();
Strin g encc=assy .get Secret Key ();
Strin g IBEE = new as syme tric ( ) .
    symmetric Encryption (IBE , encc );
System.out.println ("===encrypted _text "+IBEE);
Strin g ItfText=new StringXORer ().encode (IBE , encc )
    ;
System.out.println ("transfor _text=== "+ItfText);
//storing encrypted file
File Write r fw=new File Write r (file);
fw.write (ItfText);
fw.close ();
SplitFile sp = new SplitFile ();

```

```

        sp.Split(filepath+filenames);
//      System.out.println("string Buffer"+IBE) ;

// date and Time
DateFormat dateFormat = new SimpleDateFormat ("
    yyyy.MM.dd_G'at'_HH:mm:ss");
Date date = new Date();
String time= dateFormat.format(date);
System.out.println("current_Date_"+time);

String len=file.length()+" bytes";
long numSplits = 3; //from user input,      extract it
      from args
// uploading file
int flag=0;
for (int destIx = 1; destIx <= numSplits; destIx
    ++){
    File fes1=new File("D:\\Book\\Split\\" +
        destIx + ".txt");
    boolean status=new Ftpcon().upload(fes1);

    if(status){
        flag=1;
    }
}
if(flag==1)
{
    //Connection con= Dbconnection.getConnection();
    Statement st=con.createStatement();
    int i=st.executeUpdate("insert into _
        fileupload(privatekey,filename,fcaption ,
        filess ,dates ,ownname ,key ) value s('"+
        privakey+" ', '"+file name+" ', '"+caption+" ', '
        "+ltfText+" ', '"+time+" ', '"+ownername+" ', '
        +encc+" ')" );
    System.out.println(i);
}

```

```

        if (i != 0) {
            // out.println(" success ");
            response.sendRedirect("owfileupload.jsp?
                fmsg='uploded'");

        }
        else {
            out.println("error_while_uploading_
                additional_informations");
        }
//        out.println("file stored");
//    }    out.println(file.length());
    }
}
}
catch (Exception e) {
    out.println(e);
}
finally {
    out.close();
}
}

```

```

// <editor -fold defaultstate="collapsed" desc="
    HttpServlet methods . Click on the + sign on the left
    to edit the code." >

```

```

/**
 * Handles the HTTP
 * <code>GET</code> method .
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error
 * occurs
 *
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet ( HttpServlet Request request ,
    HttpServlet Response response )

```

```

        throws ServletException , IOException    {
        process Request (request , response );
    }

/**
 * Handles the HTTP
 * <code>POST</code> method .
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error
 * occurs
 *
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost ( Http Servlet Request request ,
        Http Servlet Response response )

        throws ServletException , IOException    {
        process Request (request , response );
    }

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
@Override
public String getServletInfo() {
    return "Short _description";
}

}

```

## **8. TESTING AND RESULTS**

### **8.1 INTRODUCTION**

Testing is an important phase in the development life cycle of the product this was the phase where the error remaining from all the phases was detected. Hence testing performs a very critical role for quality assurance and ensuring the reliability of the software. Once the implementation is done, a test plan should be developed and run on a given set of test data. 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 suppose to do. Testing is the final verification and validation activity within the organization itself. In the testing stage following goals are tried to achieve:-

- To affirm the quality of the project.
- To find and eliminate any residual errors from previous stages.
- To validate the software as the solution to the original problem.
- To provide operational reliability of the system.

During testing the major activities are concentrated on the examination and modification of the source code. The test cases executed for this project are listed below. Description of the test case, steps to be followed; expected result, status and screenshots are explained with each of the test cases.

### **8.2 TESTING METHODOLOGIES**

There are many different types of testing methods or techniques used as part of the software testing methodology. Some of the important types of testing are:

#### **8.2.1 WHITE BOX TESTING**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot from a black box level. Using white box testing we can derive test cases that:Guarantee that all independent paths within a module have been exercised at.



- Execute all loops at their boundaries and within their operational bounds.
- Execute internal data structure to assure their validity.

### **8.2.2 BLACK BOX TESTING**

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

- Incorrect or missing function.
- Interface errors.
- Performance errors.
- Initialization and termination errors.
- Errors in objects.

#### **Advantages:**

- The test is unbiased as the designer and the tester are independent of each other.
- The tester does not need knowledge of any specific programming languages.
- The test is done from the point of view of the user, not the designer.
- Test cases can be designed as soon as the specifications are complete.

### **8.2.3 UNIT TESTING**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases. **Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

#### **Test objectives:**

- All Components must work properly.
- Proper coordinates should be sent by the Android app to the Arduino

Test Case ID
Purpose
Preconditions
Inputs
Expected Outputs
Postconditions

Table 8.1: Unit Test Cases

- The entry screen, messages and responses must not be delayed in the Android app.

### 8.3 TEST CASE 1

**Function:**User RegistrationFormTest.

**Purpose:**User registration.

**Preconditions:**Server should be running.

**Inputs:**Username, password as NULL.

**Expected Outputs:** Display Alert messages for null fields.

**Postconditions:**Redirect for user registration page.

### 8.4 TEST CASE 2

**Function:**User RegistrationFormTest.

**purpose:**User registration.

**Preconditions:**Server should be running.

**Inputs:**Username, password as valid inputs.

**Expected Outputs:**Display Registration Success.

**Postconditions:**Store the user details in database.

### 8.5 TEST CASE 3

**Function:**User LoginVerificationTest.

**Purpose:**User login.

**Preconditions:**Server should be running and database should be up.

**Inputs:** Username, password, Sk,PK,as valid inputs.

**Expected Outputs:**Login Success.

**Postconditions:**Redirect to user home page.

## 8.6 TEST CASE 4

**Function:**File upload test.

**Purpose:**Owner file upload.

**Preconditions:**secure connection to cloud.

**Inputs:**File name, caption.

**Expected Outputs:**upload success.

**Postconditions:**spilt and store file in cloud.

## 8.7 TEST CASE 5

**Function:**File Download Test with wrong OTP.

**Purpose:**File download.

**Preconditions:**File request must have been accepted.

**Inputs:**wrong OTP

**Expected Outputs:**Display wrong OTP

**Postconditions:**Redirect to enter OTP page.

## 8.8 TEST CASE 6

**Function:**File download with correct OTP.

**Preconditions:**File request must have been accepted.

**Purpose:**File download.

**Inputs:**Correct OTP.

**Expected Outputs:**Download file.

**Postconditions:**Redirect to user home page.

## 8.9 IMAGE OF HOME PAGE



Figure 8.1: Home page

## 8.10 IMAGE OF ADMIN, OWNER AND USER LOGIN PAGE



Figure 8.2: Admin, Owner and User login page

8.11 IMAGE OF FILE UPLOAD



Figure 8.3: File upload

8.12 IMAGE OF FILE DOWNLOAD

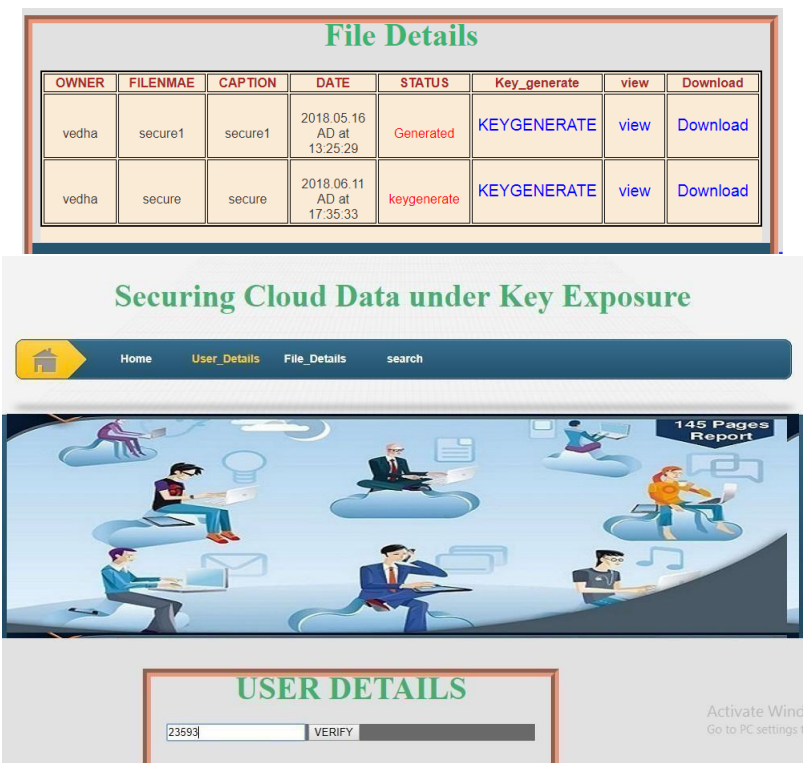


Figure 8.4: File Download

## **9. CONCLUSION AND FUTURE SCOPE**

### **9.1 CONCLUSION**

We addressed the problem of securing data outsourced to the cloud against an adversary which has access to the encryption key. For that purpose, we introduced a novel security definition that captures data confidentiality against the new adversary. We then proposed Bastion, a scheme which ensures the confidentiality of encrypted data even when the adversary has the encryption key, and all but two ciphertext blocks. Bastion is most suitable for settings where the ciphertext blocks are stored in multi-cloud storage systems. In these settings, the adversary would need to acquire the encryption key, and to compromise all servers, in order to recover any single block of plaintext. We analyzed the security of Bastion and evaluated its performance in realistic settings. Bastion considerably improves (by more than 50%) the performance of existing primitives which offer comparable security under key exposure, and only incurs a negligible overhead (less than 5%) when compared to existing semantically secure encryption modes (e.g., the CTR encryption mode). Finally, we showed how Bastion can be practically integrated within existing dispersed storage systems.

### **9.2 FUTURE SCOPE**

Our project provides an environment where a data owner can share text data content with members of his group while preventing any outsiders from gaining any data access in case of any malicious activities such as key theft. In the future we can provide support for other types of multimedia (text, images, audio, video etc.,) content.

# REFERENCES

- [1] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, Fault-Scalable Byzantine Fault-Tolerant Services, in ACM Symposium on Operating Systems Principles (SOSP), 2005, pp. 5974.
- [2] M. K. Aguilera, R. Janakiraman, and L. Xu, Using Erasure Codes Efficiently for Storage in a Distributed System, in International Conference on Dependable Systems and Networks (DSN), 2005, pp. 336345.
- [3] W. Aiello, M. Bellare, G. D. Crescenzo, and R. Venkatesan, Security amplification by composition: The case of doublyiterated, ideal ciphers, in Advances in Cryptology (CRYPTO), 1998, pp. 390407.
- [4] C. Basescu, C. Cachin, I. Eyal, R. Haas, and M. Vukolic, Robust Data Sharing with Key-value Stores, in ACM SIGACT- SIGOPS Symposium on Principles of Distributed Computing (PODC), 2011, pp. 221222.
- [5] A. Beimel, Secret-sharing schemes: A survey, in International Workshop on Coding and Cryptology (IWCC), 2011, pp. 1146.
- [6] A. Bessani, M. Correia, B. Quaresma, F. Andr, and P. Sousa, DepSky: Dependable and Secure Storage in a Cloud-ofclouds, in Sixth Conference on Computer Systems (EuroSys), 2011, pp. 3146.
- [7] G. R. Blakley and C. Meadows, Security of ramp schemes, in Advances in Cryptology (CRYPTO), 1984, pp. 242268.
- [8] V. Boyko, On the Security Properties of OAEP as an Allor- nothing Transform, in Advances in Cryptology (CRYPTO), 1999, pp. 503518.
- [9] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, Deniable Encryption, in Proceedings of CRYPTO, 1997
- [10] Cavalry, Encryption Engine Dongle, <http://www.cavalrystorage.com/en2010.aspx/>