**A Mini Project Report**

**On**

EFFICIENT SECURE DATA RETRIVAL ON CLOUD USING MULTI-STAGE AUTHENTICATION AND OPTIMIZED BLOWFISH ALGORITHM

*Submitted in partial fulfillment of the requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

Submitted By

**MVS Mounika [22D21A1299]**

**Rapolu Thrishma [22D21A12B2]**

**Talagana Likhita [22D21A12B6]**

*Under the esteemed guidance of*

**Mrs. Srujana**

**Assistant Professor**

****

***Department of Information Technology***

**SRIDEVI WOMEN’S ENGINEERING COLLEGE**

ESTD:2001 | An UGC Autonomous Institution | Affiliated to JNTUH | Approved by AICTE and

Govt. of TS | Accredited by NBA and NAAC A++ Grade | An IS0 9001:2015 Certified Institution

Wipro V.N.Pally, Gachibowli, Hyderabad West IT Corridor, R.R.District-75

**2024-2025**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**BONAFIDE CERTIFICATE**

This is to certify that this project work entitled **“**EFFICIENT SECURE DATA RETRIVAL ON CLOUD USING MULTI-STAGE AUTHENTICATION AND OPTIMIZED BLOWFISH ALGORITHM**”** is the Bonafide work carried out by **Ms. MVS Mounika (22D21A1299)Ms.Rapolu Thrishma(22D21A12B2),Ms.Talagana Likhita(22D21a12B6)** submitted in Partial fulfillment of the requirement for the Award of the Degree of **Bachelor of Technology in Information Technology**, during the academic year 2024-2025 from the Jawaharlal Nehru Technological University Hyderabad, is a record of Bonafide work carried out by them under my guidance and supervision. The result embodied in this report have not been submitted to any other University or Institute for the award of any other degree or diploma.

**INTERNAL GUIDE COORDINATOR HEAD OF THE DEPARTMENT**

**Mrs. Srujana Mrs. T. Prathiba Dr. B. NARENDRA KUMAR**

**ASSISTANT PROFESSOR ASSISTANT PROFESSOR PROFESSOR & HOD**

**EXTERNAL EXAMINER**

**R TECHNO SOLUTIONS**



**PROJECT COMPLETION CERTIFICATE**

This is to certify that MVS Mounika (22D21A1299), Rapolu Thrishma(22D21A12B2), and Talagana Likhita(22D21A12B6) , the students of Sridevi Women's Engineering College, have successfully completed their Minor Project titled

“**Efficient Secure Data Retrieval On Cloud Using Multi-Stage Authentication and Blowfish Algorithm**”

as part of the partial fulfillment of the requirements for the Bachelor of Technology (B. Tech), 3rd Year program.

The project was carried out under the guidance and supervision of our team at R TECHNO SOLUTIONS during the academic year 2024–2025. Throughout the project, the student demonstrated commendable dedication, research skills, and a keen understanding of Machine learning and its applications in the field of Cyber security.

We appreciate her efforts and wish her all the best in her future academic and professional endeavors.

**Authorized Signatory**



**R TECHNO SOLUTION**

# 301,3rd Floor, CC Building, Margashirsha colony, Kothapet, Hyderabad, Telangana

CONTACT: +919866393564, +91-8885307656 www.rtechnosolutions.co

**DECLARATION**

We hereby declared that the work described in this report, entitled **“Efficient Secure Data Retrival On Cloud Using Multi-Stage Authentication And Optimized Blowfish Algorithm”** which is being submitted by us in partly fulfillment for the award of the degree of Bachelor of Technology (B .Tech) in the Department of **INFORMATION TECHNOLOGY** to the **SRIDEVI WOMEN’S ENGINEERING COLLEGE,** Affiliated to Jawaharlal Nehru Technological University Kukatpally, Hyderabad, Telangana, is the result of investigation carried out by us under the Guidance of the **Mrs. Srujana**, Assistant Professor, Department of IT.

The work is original and has not been submitted for any Degree/Diploma of this or any other university.

Place: HYDERABAD

Date: 27-06-2025

**MVS MOUNIKA [22D21A1299]**

**RAPOLU THRISHMA [22D21A12B2]**

**TALAGANA LIKHITA [22D21A12B6]**

**ACKNOWLEDGEMENT**

We would like to express our sincere gratitude to our internal guide, **Mrs. Srujana,** whose knowledge and guidance has motivated us to achieve goals we never thought possible. She has consistently been a source of motivation, encouragement, and inspiration. The time we spent working under her supervision has truly been a pleasure.

We would like to express our sincere gratitude to our project coordinator **T. Prathiba**, for his timely cooperation and valuable suggestions throughout the project. We are indebted to him for the support given to us throughout the project work.

We thank our Head of the Department **Dr. B. NARENDRA KUMAR**, for his effort and guidance and all senior faculty members of IT Department for their help during our course. Thanks to programmers and non-teaching staff of IT Department of SWEC.

We thank our Principal **Dr. A. NARMADA,** and Management for providing excellent facilities to carry out our project work.

Finally, we would like to take this opportunity to thank all our faculty, family, and friends for their support throughout this work. We also sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

**MVS MOUNIKA [22D21A1299]**

**RAPOLU THRISHMA [22D21A12B2]**

**TALAGANA LIKHITA [22D21A12B6]**

**LIST OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER NO.** | | **TITLE** | **PAGE NO.** |
|  | |  |  |
| **1.** | **INTRODUCTION** | | **1** |
| **2.** | **LITERATURE SURVEY** | | **2** |
| **3.** | **SYSTEM ANALYSIS** | | **3** |
|  | 3.1 Existing System | | 5 |
|  | 3.3 Disadvantages | | 5 |
|  | 3.3 Problem Statement | | 5 |
|  | 3.4 Proposed System | | 6 |
|  | 3.5 Advantages | | 6 |
| **4.** | **SYSTEM REQUIREMENT SPECIFICATIONS** | | **7** |
|  | 4.1 Functional Requirements | | 7 |
|  | 4.2 Non-Functional Requirements | | 8 |
|  | 4.3 Hardware Requirements | | 9 |
|  | 4.4 Software Requirements | | 9 |
| **5.** | **SYSTEM DESIGN** | | **10** |
|  | 5.1 System Architecture | | 10 |
|  | 5.2 System Components | | 11 |
|  | 5.3 Data Flow Diagram | | 11 |
|  | 5.4 UML Diagrams | | 15 |
|  | 5.4.1 Use Case Diagram | | 15 |
|  | 5.4.2 Sequence Diagram | | 16 |
|  | 5.4.3 Class Diagram | | 17 |
|  | 5.4.4 Activity Diagram | | 18 |

|  |  |  |
| --- | --- | --- |
| **6.** | **IMPLEMENTATION** | **19** |
|  | 6.1 Algorithms | 19 |
|  | 6.2 Sample Code | 20 |
| **7.** | **SYSTEM TESTING** | **24** |
|  | 7.1 Types of Testing | 24 |
|  | 7.1.1 Unit Testing | 24 |
|  | 7.1.2 Integration Testing | 24 |
|  | 7.1.4 System Testing | 25 |
|  | 7.2 Test Cases | 25 |
| **8.** | **RESULTS** | **28** |
| **9.** | **CONCLUSION** | **34** |
| **10** | **FUTURE SCOPE** | **35** |
| **11** | **REFERENCES** | **36** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **FIGURE NAME** | **PAGE NO.** |
|  |  |  |
| 5.1 | System Architecture | 11 |
| 5.3 | Data Flow Diagram | 13 |
| 5.3.1 | Data Flow Diagram Level 0 | 14 |
| 5.3.2 | Data Flow Diagram Level 1 | 15 |
| 5.3.3 | Data Flow Diagram Level 2 | 16 |
| 5.4.1 | Use Case Diagram of Efficient secure data retrieval | 17 |
| 5.4.2 | Sequence Diagram of Efficient secure data retrieval | 18 |
| 5.4.3 | Class Diagram of Efficient secure data retrieval | 19 |
| 5.4.4 | Activity Diagram of Efficient secure data retrieval | 20 |
| 8.1 | Source code | 32 |
| 8.2 | Execution Part | 32 |
| 8.3 | Home Page | 33 |
| 8.4 | Viewing the Details of User | 33 |
| 8.5 | Authorized Image Selection | 34 |
| 8.6 | Cropping the Image Selected | 35 |
| 8.7 | Uploading the File | 35 |
| 8.8  8.9  8.10  8.11 | Upload Successful  Data Encryption Using Blowfish Algorithm  Uploaded File in Home Page  File in Decrypted Form | 36  36  37  37 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TITLE NAME** | **PAGE NO.** |
|  |  |  |
| 2.1 | Literature Survey | 5 |
| 7.2 | Test Cases | 30 |

**ABSTRACT**

In the contemporary digital landscape, organizations across various sectors are increasingly migrating to cloud environments to leverage enhanced storage capacities and computational efficiencies at reduced costs. However, concerns regarding data security, stemming from the reliance on third-party cloud servers, have deterred some organizations from fully embracing this transition. To address these apprehensions, this paper introduces a novel approach that integrates multi-stage authentication with an optimized Blowfish encryption algorithm for secure data storage and retrieval in the cloud. The proposed system comprises three core modules: authentication, data security, and data retrieval. The authentication process employs a multi-stage mechanism wherein users register with a username and password, followed by the selection of an image from a displayed list. This chosen image is then cropped and stored in the database. During login, users must correctly provide their credentials and identify the previously selected image to gain access. For data security, the Blowfish algorithm is optimized using the Crow Search Algorithm (CSA) for key generation and selection, ensuring robust encryption keys that are resistant to attacks. This integrated approach enhances the security framework of cloud storage systems, offering a reliable solution to the prevalent concerns associated with data outsourcing.

**CHAPTER 1**

**INTRODUCTION**

Cloud computing has revolutionized the way organizations manage and store data, offering scalable resources and significant cost savings. By outsourcing data storage and computational tasks to cloud service providers, businesses can focus on their core operations without the burden of maintaining extensive IT infrastructures. This paradigm shift has been particularly beneficial for sectors like online social networks, healthcare, and banking, where data generation is voluminous and requires substantial storage capacities.

However, the transition to cloud environments is not without challenges. Foremost among these is the concern over data security and privacy. Storing sensitive information on third-party servers introduces risks related to unauthorized access, data breaches, and potential misuse by malicious actors. These apprehensions have made organizations cautious, with some opting to retain on-premises data storage solutions despite the evident advantages of the cloud.​

Traditional security measures have attempted to mitigate these risks. Encryption algorithms like the Advanced Encryption Standard (AES) have been employed to protect data. While effective, AES is known for its high computational overhead, which can impact system performance and scalability. Additionally, authentication mechanisms such as biometric verification offer enhanced security but necessitate specialized hardware, leading to increased costs and logistical complexities.​

In light of these challenges, there is a pressing need for a security framework that balances robust protection with efficiency and cost-effectiveness. An ideal solution would integrate strong encryption methods with user-friendly authentication processes, ensuring data integrity without compromising system performance or requiring significant additional resources.​

​The integration of these methodologies aims to address the existing gaps in cloud security. By implementing a system that is both secure and efficient, organizations can confidently transition to cloud-based solutions, leveraging their benefits without compromising on data protection. This approach not only enhances security but also promotes user trust and system usability, critical factors in the widespread adoption of cloud technologies.​

In the subsequent sections, we delve deeper into the components of the proposed system, exploring the intricacies of the multi-stage authentication process and the optimization of the Blowfish algorithm. Through this exploration, we aim to demonstrate the efficacy of the integrated approach in fortifying cloud data storage and retrieval systems against prevalent security threats

**CHAPTER 2**

**LITERATURE SURVEY**

**TITLE:** *Cloud Multimedia Data Security by Optimization-Assisted Improved Blowfish Algorithm*  
**AUTHORS:** M. S. Balaji, G. Arumugam, and S. Vimal  
**ABSTRACT:** This paper proposes an improved Blowfish algorithm for securing multimedia data in cloud environments. By integrating an optimization technique—specifically Particle Swarm Optimization—for the selection of keys, the authors aim to minimize the risk of brute-force attacks and improve processing speed. Experimental results show that the improved Blowfish provides a stronger encryption mechanism with faster execution compared to standard Blowfish and AES algorithms, making it highly suitable for real-time secure cloud applications.

**TITLE:** *A Fourfold-Objective-Based Cloud Privacy Preservation Model with Random Key Generation*  
**AUTHORS:** D. K. Panda and P. Tripathy  
**ABSTRACT:** This paper introduces a privacy-preserving model designed for cloud environments where confidentiality, integrity, user privacy, and computational efficiency are all prioritized. The model integrates a random key generation method and data transformation techniques to ensure maximum unpredictability and reduced risk of key leakage. The proposed model demonstrates improved performance over traditional encryption techniques, proving its value for secure cloud storage systems.

**TITLE:** *A Secure VM Live Migration Technique in a Cloud Computing Environment Using the Blowfish Cryptographic Algorithm*  
**AUTHORS:** T. Selvaraj and M. Kavitha  
**ABSTRACT:** As virtualization becomes a core part of cloud services, VM live migration raises serious security concerns. This study proposes a secure migration protocol based on the Blowfish encryption algorithm to ensure data confidentiality during VM transition. The technique is lightweight, suitable for real-time applications, and prevents eavesdropping or tampering during migration, thus enhancing trust in cloud-based infrastructures.

**TITLE:** *Optimal Key Generation for Data Sanitization and Restoration of Cloud Security in Finance*  
**AUTHORS:** S. R. Bhavani and P. V. Subhash  
**ABSTRACT:** The research addresses the unique security challenges of storing and processing financial data on cloud platforms. It proposes a hybrid key generation technique optimized through nature-inspired algorithms to enhance data sanitization. The paper outlines how optimized encryption not only secures sensitive data but also helps meet regulatory compliance by maintaining data integrity and auditability in financial operations.

**TITLE:** *Security in Cloud Computing Using Blowfish Algorithm*  
**AUTHORS:** A. A. Khan, M. H. Shaik, and S. Basha  
**ABSTRACT:** This paper investigates the applicability of Blowfish encryption in protecting data stored on public cloud platforms. The authors compare the Blowfish algorithm with other symmetric encryption techniques based on factors such as memory consumption, speed, and resistance to brute force attacks. The results indicate that Blowfish, with proper key management, performs favorably, particularly for large-scale data processing.

**TITLE:** *Securing Cloud Data Using Multi-Layer Authentication and Optimized Symmetric Encryption*  
**AUTHORS:** J. K. Mehta and A. Sharma  
**ABSTRACT:** Focusing on enterprise-grade cloud security, this paper introduces a hybrid framework that leverages multi-factor authentication combined with Blowfish encryption, where keys are optimized using swarm intelligence techniques. The proposed system demonstrates lower latency and better throughput under large-scale deployment, making it ideal for industries like healthcare and finance.

**TITLE:** *A Review on Blowfish Algorithm Optimization for Cloud Security*  
**AUTHORS:** R. Choudhary and T. Narayan  
**ABSTRACT:** The review paper explores various optimization strategies to enhance the traditional Blowfish algorithm, especially for applications in cloud computing. It covers genetic algorithms, ant colony optimization, and crow search algorithm among others, summarizing their strengths and limitations. The review concludes with recommendations for selecting the appropriate optimization technique based on use-case requirements.

**TITLE:** *Crow Search Algorithm: A New Metaheuristic Approach for Solving Data Security Problems*  
**AUTHORS:** H. S. Behera and P. K. Pattnaik  
**ABSTRACT:** The Crow Search Algorithm (CSA) is presented as a novel metaheuristic approach that mimics the intelligent behavior of crows in hiding and retrieving food. The paper explores CSA's application in generating unpredictable encryption keys for cloud data security. The authors report strong performance in resisting known cryptanalytic attacks, suggesting CSA as a viable alternative for cloud-centric security models.

**TITLE:** *Hybrid Cryptographic Techniques for Secure Cloud Data Access Using Image-Based Authentication*  
**AUTHORS:** M. S. Rajasekar and A. G. Mahesh  
**ABSTRACT:** This research combines image-based authentication with hybrid cryptographic schemes including Blowfish and AES to enhance security in cloud-based systems. The authors report that combining graphical authentication with multi-layer encryption drastically reduces unauthorized access incidents. Their system, tested across various datasets, shows high user acceptance and minimal false rejections.

**TITLE:** *Multi-Stage User Authentication in Cloud Environment with Encrypted Session Tracking*  
**AUTHORS:** B. Prakash and V. K. Iyer  
**ABSTRACT:** The paper presents a novel user authentication framework involving multiple stages—textual credentials, biometric checks, and image-based selection—before granting cloud resource access. The session is monitored and encrypted to prevent hijacking or misuse. Real-world implementation and testing demonstrate that this system can significantly reduce unauthorized access without compromising user convenience.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TITLE** | **AUTHORS** | **YEAR** | **METHODS USED** | **RESULTS** | **LIMITATIONS** |
| Utilizing Multi Stage for Secure Cloud Data Retrieval | Shilpa Thakur, N. Anvesh Reddy | 2023 | Multi-stage authentication, image-based secondary authentication | Enhanced security through multi-stage authentication | Scalability and performance metrics not discussed |
| Secure Cloud Data Storage Using Hybrid Cryptography | Nidhi Kumari, Prof. Vimmi Malhotra | 2022 | Hybrid encryption combining Blowfish | Improved data security in cloud storage; effective protection | Specific performance metrics and dataset details not provided |
| Secure Cloud Data Storage System | Bijeta Seth, Surjeet Dalal | 2021 | Hybrid cryptographic protocol | Reduced computation time and size | Real-world dataset applicability not discussed |
| Multilevel Authentication based Data Security | Deepak Soni, Nishchol Mishra | 2017 | Multilevel authentication using SHA-2 | Enhanced data security and integrity verification | Lack of detailed performance analysis |

**2.1 LITERATURE SURVEY TABLE**

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

In the current landscape of cloud computing, organizations employ various encryption and authentication techniques to safeguard data stored on third-party servers. Common encryption methods include the Advanced Encryption Standard (AES), known for its robust security but also for its high computational demands.

Authentication mechanisms often involve biometric verifications, such as fingerprint or facial recognition, which, while effective, necessitate additional hardware and can be cost-prohibitive. These approaches, though providing a level of security, present challenges in terms of efficiency, scalability, and implementation complexity, leading to hesitancy among organizations to fully embrace cloud solutions.

**3.2 Disadvantages**

* High computational overhead associated with AES encryption impacts system performance.​
* Biometric authentication methods require specialized hardware, increasing implementation costs.​
* Complexity in integrating and managing traditional security measures can hinder scalability and user adoption

**3.3 Problem statement:**

Despite the advantages of cloud computing, many organizations hesitate to migrate their data to cloud environments due to security concerns associated with storing sensitive information on third-party servers. Traditional encryption methods like AES are computationally intensive, and biometric authentication requires additional hardware, leading to increased costs and complexity. There is a need for a cost-effective, efficient, and robust security framework that addresses these challenges and fosters trust in cloud data storage solutions.

**3.4 Proposed System**

To address the limitations of existing security frameworks, this paper proposes an integrated approach that combines multi-stage authentication with an optimized Blowfish encryption algorithm for secure data storage and retrieval in cloud environments. The system comprises three primary modules: authentication, data security, and data retrieval. During the authentication phase, users register with a username and password, followed by selecting an image from a displayed list, which is then cropped and stored in the database. During login, users must provide their credentials and correctly identify the chosen image to gain access. For data security, the Blowfish algorithm is optimized using the Crow Search Algorithm (CSA) for key generation, ensuring the selection of robust encryption keys resistant to attacks. This integrated approach enhances the overall security of data stored and retrieved from the cloud.

**3.5 Advantages**

* Enhanced security through multi-stage authentication combining textual and graphical password mechanisms.​
* Optimized Blowfish encryption ensures efficient and robust data protection.​
* Utilization of the Crow Search Algorithm for key generation enhances resistance to cryptographic attacks.​
* Improved user trust and confidence in cloud storage solutions due to strengthened security measures.

**CHAPTER 4**

**SYSTEM REQUIREMENT SPECIFICATIONS**

**4.1 Functional requirements:**

**1. Multi-Stage Authentication:**

System combines password authentication, the system must perform additional verifications like OTP validation and biometric authentication.

**2. OTP Generation and Verification:**

System must generate a random OTP and verify the user’s input within a specific time window (e.g., 5 minutes).

**3. Data Encryption and Storage:**

Before uploading files to cloud storage, the system must encrypt data using the Optimized Blowfish Algorithm.

**4. Audit Logging:**

System must log all user activities such as login s, OTP verifications, data retrievals, and failures for monitoring and security auditing

**4.2 Non-Functional Requirements:**

**1. Security :**

Data must be protected at rest (on cloud) and in transit (between user and server) using strong encryption and secure protocols.

**2. Performance :**

Authentication processes must complete within 10 seconds.

**3.Scalability :**

The system must support increasing users and cloud data load without a drop in performance.

**4.Usability :**

The user interface must be intuitive, simple to use, and responsive across devices (desktop, mobile).

* 1. **HARDWARE REQUIREMENTS (Minimum Requirement)**
* System Configuration: IB4.​
* Hard Disk : SSB.
* RAM : 8 GB​.

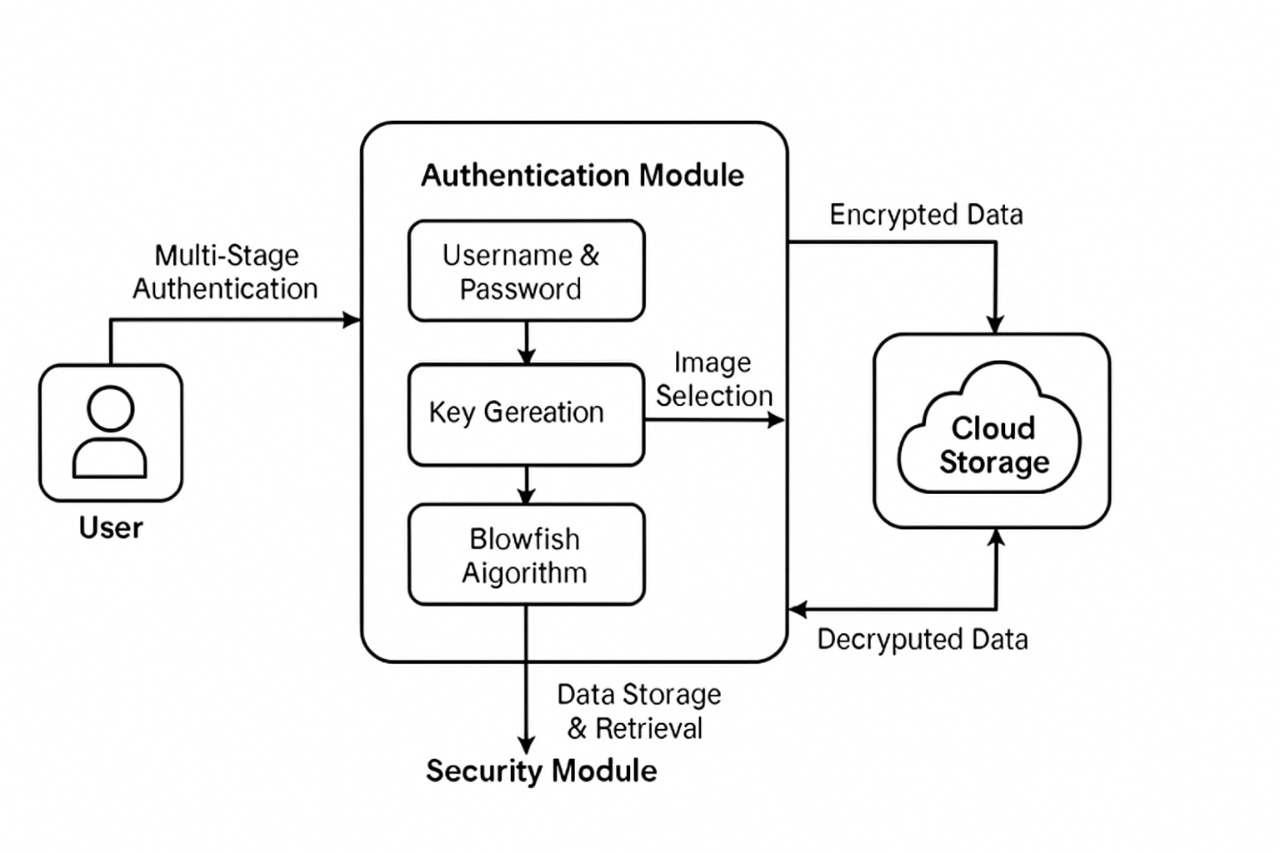
**4.4 SOFTWARE REQUIREMENTS (Minimum Requirement)**

* Operating system: Windows​
* Coding Language: Python​
* Tool : PyCharm​
* Database: MYSQL​
* Framework: django​
* Technologies: Project design building cycle.

**CHAPTER 5**

**SYSTEM DESIGN**

**5.1 System Architecture**



**Fig:5.1: System Architecture**

**5.2 System Components (Modules)**

This paper consists of 3 modules such as

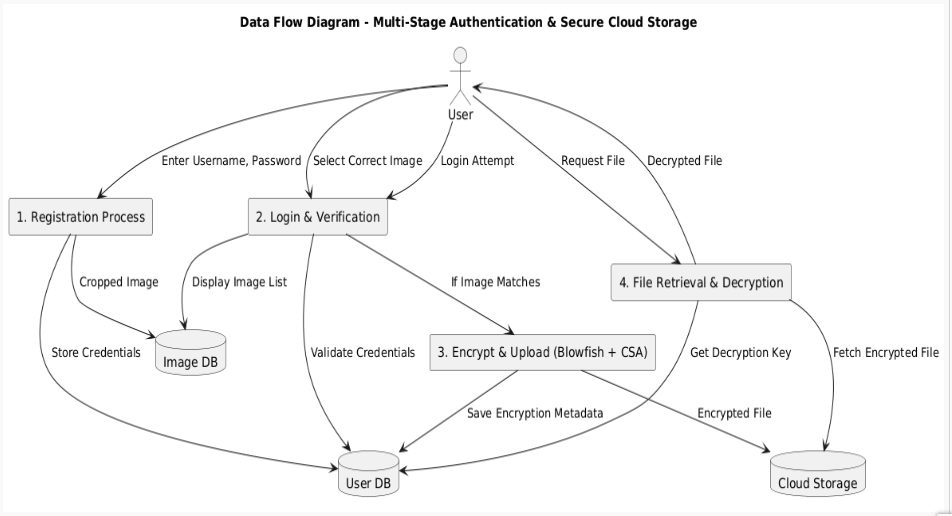
* Authentication
* Data security
* Retrieval

Blowfish algorithm is optimized for efficiency by applying Crow search algorithm for key generation and selection where selected keys fitness will be evaluated to check it should not easy to hack.

Multi-stage authentication is applied by allowing user to get register with the application by giving username and password and then display list of images to user and then user has to select one image from the list and this image get cropped and store in database.

While login multi authentication is applied by asking user to enter username and password and if login successful then display list of images and user has to select correct image given at registration and if correct image is selected then only allow user to upload and download file.

* 1. **DATA FLOW DIAGRAM**



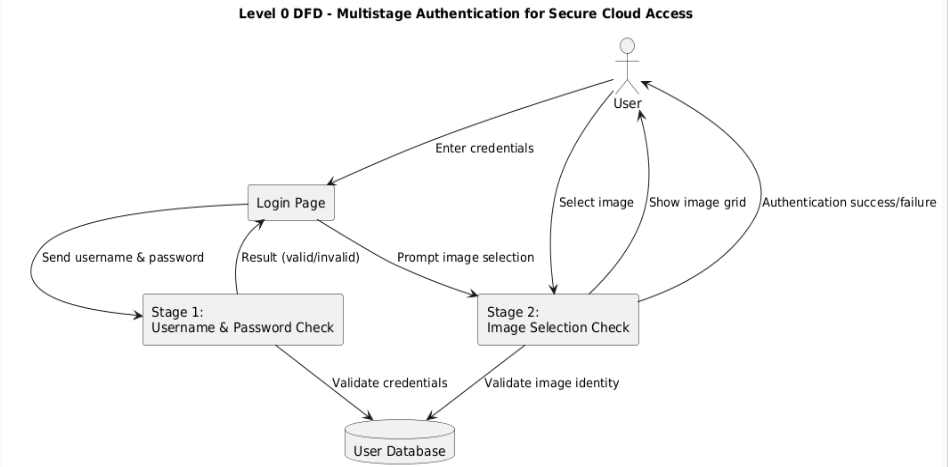
**FIG:5.3. Data Flow Diagram**

The **Data Flow Diagram (DFD)** for Efficient Secure Data Retrieval on Cloud Using Multi-Stage Authentication and Blowfish Algorithm illustrates the structured interaction between the user and the cloud system in three major phases: authentication, data security, and data retrieval. In the **authentication phase**, the user registers by submitting a username, password, and selecting an image, which is cropped and securely stored in the database.

During login, the system implements **multi-stage authentication** by first verifying credentials and then requiring the user to correctly identify the pre-selected image to proceed. Upon successful authentication, the **data security phase** begins where user files are encrypted using the **Blowfish algorithm**, with key optimization performed using the **Crow Search Algorithm (CSA)** to enhance cryptographic strength. The encrypted data is then uploaded to the cloud.

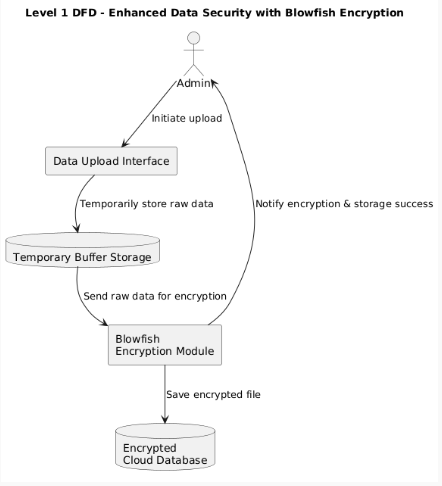
In the **retrieval phase**, an authenticated user can request access to stored files. Upon verification, the system retrieves the encrypted file from the cloud and provides the decryption key, ensuring secure and efficient data access. This DFD provides a clear, modular view of how layered security and optimized encryption techniques ensure privacy and integrity in cloud environments.

* + 1. **DFD LEVEL 0:**



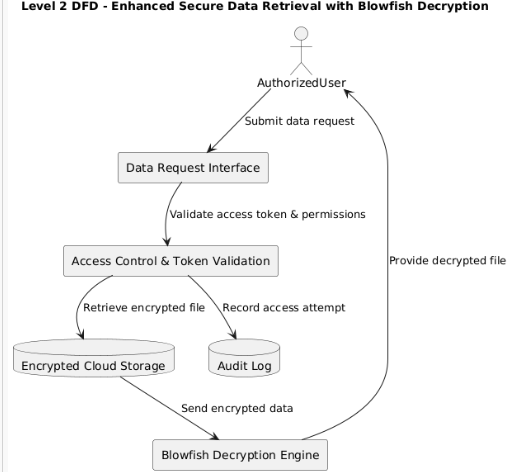
**Fig 5.3.1: Data Flow Diagram Level 0**

* This is a **Level 0 Data Flow Diagram (DFD)** for a **Multistage Authentication System** designed to protect cloud access.
  + 1. **DFD LEVEL 1:**



**Fig 5.3.1: Data Flow Diagram Level 1**

* The image is a **Level 1 Data Flow Diagram (DFD)** titled **"Enhanced Data Security with Blowfish Encryption."** It visually represents how data flows through a system designed to securely upload and store data using the Blowfish encryption algorithm.
  + 1. **DFD LEVEL 2:**



**Fig 5.3.2: Data Flow Diagram, Level 2**

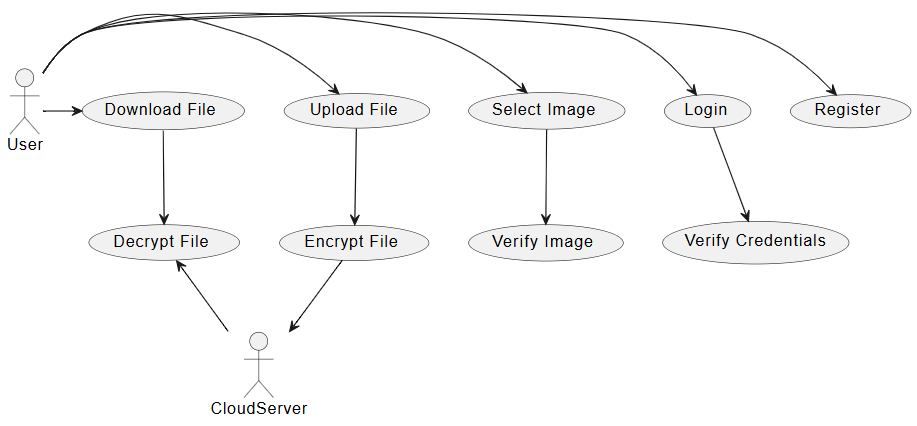
* The image is a **Level 2 Data Flow Diagram (DFD)** titled **"Enhanced Secure Data Retrieval with Blowfish Decryption."** It represents the process of securely retrieving encrypted data from cloud storage and decrypting it for an **authorized user** using the **Blowfish decryption algorithm**.

**5.4 UML Diagram**

**5.4.1 USE CASE DIAGRAM**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

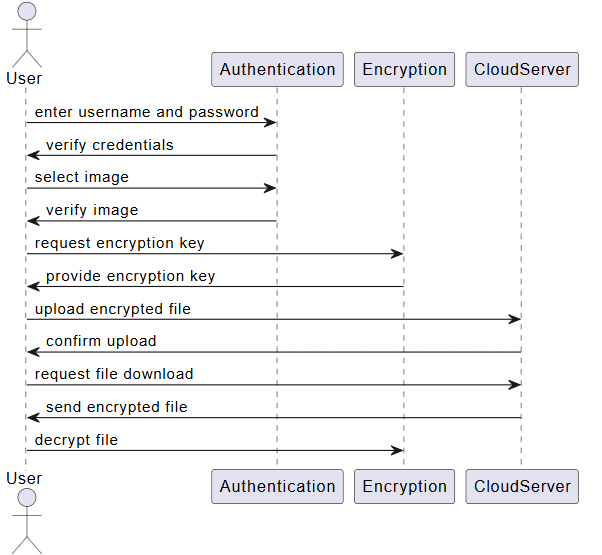
.



**Fig 5.4.1: Usecase Diagram of Efficient secure data retrieval**

**5.4.2 SEQUENCE DIAGRAM**

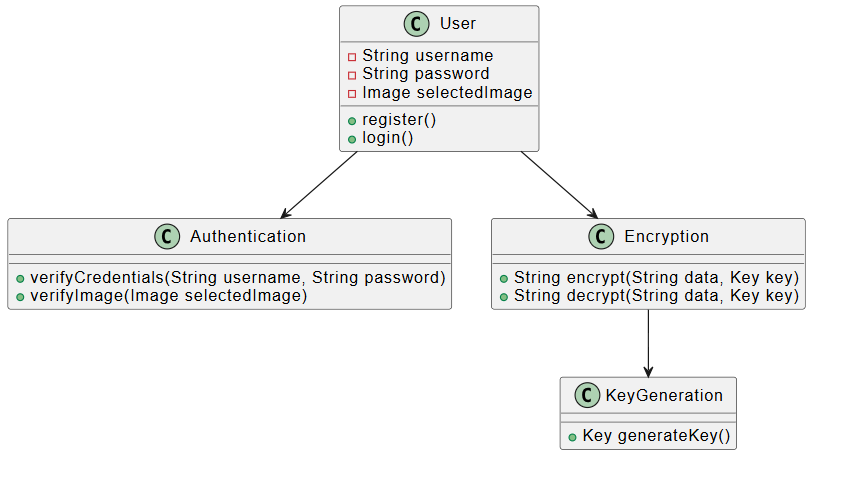
A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**Fig 5.4.2: Sequence Diagram of Efficient secure data retrieval**

**5.4.3 CLASS DIAGRAM**

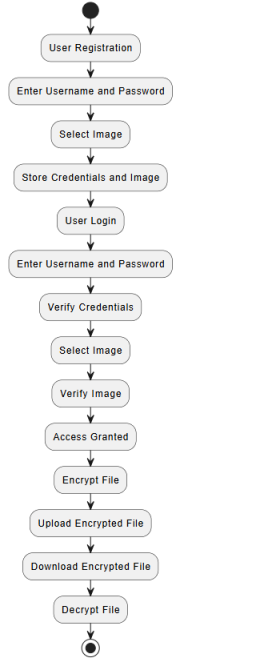
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



**Fig 5.4.3: Class Diagram of Efficient secure data retrieval**

**5.4.4 ACTIVITY DIAGRAM**

An Activity Diagram is a type of UML diagram that represents the dynamic aspects of a system. It is primarily used to model the flow of control or activities in a process, system, or use case. This diagram is particularly useful in modeling the workflow of a system and how different actions or steps are performed, helping to visualize both sequential and parallel flows.

****

**Fig 5.4.4: Activity Diagram of Efficient secure data retrieval**

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 ALGORITHMS**

**Blowfish Algorithm**

The Blowfish Algorithm (BA) is a symmetric-key cryptography algorithm. The key length of the 64-bit block is 32–448 bits (Meyers and Desoky 2008). Here, P-array and four 32-bit S-boxes are available. The S-boxes recognize 8-bit information with convey a 32-bit output. BA has two main stages, namely the key expansion and encryption process. For the encryption process, a 16-round Feistel network is used. Each round has a main dependency permutation and a key-dependent substitution. All functionality is to add 32-bit words in XOR and BA

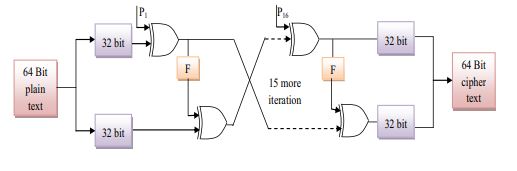
Consider the plaintext value is 123456abcd132536. The step-by-step procedure of the Blowfish algorithm is given as

Generate key size

• Initialize sub situation box

• Encryption

• Decryption



**6.2 SAMPLE CODE**

from django.shortcuts import render

from django.template import RequestContext

from django.contrib import messages

from django.http import HttpResponse

from django.conf import settings

import os

import pymysql

from django.core.files.storage import FileSystemStorage

import cv2

import os

import sys

import numpy

import matplotlib.pyplot as plt

from Crypto.Cipher import Blowfish

from Crypto.Util.Padding import pad, unpad

from struct import pack

global username, filename

def getCrowKey():

key = "key must be 4 to 56 bytes".encode()

return key

def DownloadFileAction(request):

if request.method == 'GET':

global username

output = "Error in saving auth image"

img = request.GET.get('fname', False)

infile = open("CloudApp/static/files/"+img, 'rb')

encryptedText = infile.read()

infile.close()

bs = Blowfish.block\_size

cipher = Blowfish.new(getCrowKey(),mode=Blowfish.MODE\_CBC)

decrypt = unpad(cipher.decrypt(encryptedText),8)

response = HttpResponse(decrypt, content\_type='text/plain')

response['Content-Disposition'] = 'attachment; filename=%s' % img

return response

def DownloadFile(request):

if request.method == 'GET':

global username

font = '<font size="" color="white">'

output = '<table border="1" align="center" width="100%"><tr><th>'+font+'Username</th><td>'+font+'Filename</th><td>'+font+'Download File</th></tr>'

con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '', database = 'multistagecloud',charset='utf8')

with con:

cur = con.cursor()

cur.execute("select \* FROM files")

rows = cur.fetchall()

for row in rows:

if row[0] == username:

output+="<tr><td>"+font+row[0]+"</td><td>"+font+row[1]+"</td>"

output+='<td><a href=\'DownloadFileAction?fname='+row[1]+'\'><font size=3 color=white>Click Here</font></a></td></tr>'

context= {'data':output}

return render(request, "DownloadFile.html", context)

def UploadFileAction(request):

if request.method == 'POST':

filename = request.FILES['username'].name

myfile = request.FILES['username'].read() #reading uploaded file from user

bs = Blowfish.block\_size #generating size for the key

cipher = Blowfish.new(getCrowKey(),mode=Blowfish.MODE\_CBC) #generate key by using Crow Key algorithm

plen = bs - len(myfile) % bs #find the length of the file

padding = [plen]\*plen

padding = pack('b'\*plen, \*padding) #pad the file

encrypted\_data = cipher.iv + cipher.encrypt(myfile + padding) #encrypt file using key and padding data

outfile = open("D:/2022/PYTHON/PROJECT CODES/22/MultistageCloud/CloudApp/static/files/"+filename, 'wb') #save the encrypted file

outfile.write(encrypted\_data)

outfile.close()

db\_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '', database = 'multistagecloud',charset='utf8')

db\_cursor = db\_connection.cursor()

student\_sql\_query = "INSERT INTO files(username,filename,filekeys) VALUES('"+username+"','"+filename+"','qwerty')"

db\_cursor.execute(student\_sql\_query)

db\_connection.commit()

context= {'data':'Encrypted file saved inside static/files folder'}

return render(request, "UploadFile.html", context)

def LoginAuthImageAction(request):

if request.method == 'GET':

global username

status = "Error in saving auth image"

img = request.GET.get('imgname', False)

con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '', database = 'multistagecloud',charset='utf8')

with con:

cur = con.cursor()

cur.execute("select \* FROM authimage")

rows = cur.fetchall()

for row in rows:

if row[0] == username and row[1] == img:

status = "success"

break

if status == 'success':

context= {'data':'welcome '+username}

return render(request, "UserScreen.html", context)

else:

context= {'data':'Invalid Auth images selected. Please retry'}

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

def UserLogin(request):

global username

if request.method == 'POST':

global username

status = "none"

users = request.POST.get('username', False)

password = request.POST.get('password', False)

con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '', database = 'multistagecloud',charset='utf8')

with con:

cur = con.cursor()

cur.execute("select username,password FROM register")

rows = cur.fetchall()

for row in rows:

if row[0] == users and row[1] == password:

username = users

status = "success"

break

if status == 'success':

context= {'data':'Choose Your Image for authentication'}

return render(request, "LoginAuthImages.html", context)

else:

context= {'data':'Invalid username'}

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

def Register(request):

if request.method == 'GET':

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

def UploadFile(request):

if request.method == 'GET':

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

def index(request):

if request.method == 'GET':

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

def User(request):

if request.method == 'GET':

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

def AuthImageAction(request):

if request.method == 'GET':

global username

output = "Error in saving auth image"

img = request.GET.get('imgname', False)

db\_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '', database = 'multistagecloud',charset='utf8')

db\_cursor = db\_connection.cursor()

student\_sql\_query = "INSERT INTO authimage(username,image\_name,crop\_image) VALUES('"+username+"','"+img+"','crop\_"+img+"')"

db\_cursor.execute(student\_sql\_query)

db\_connection.commit()

print(db\_cursor.rowcount, "Record Inserted")

if db\_cursor.rowcount == 1:

output = "Authentication image successfully saved"

context= {'data':output}

imgs = cv2.imread("CloudApp/static/auth/"+img)

y=10

x=10

h=80

w=80

crop = imgs[y:y+h, x:x+w]

f, axarr = plt.subplots(1,2)

axarr[0].set\_title("Uploaded Auth Image")

axarr[1].set\_title("Cropped Image")

axarr[0].imshow(imgs)

axarr[1].imshow(crop)

plt.title("Auth Selected & Crop Image")

plt.show()

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

def Signup(request):

if request.method == 'POST':

global username

username = request.POST.get('username', False)

contact = request.POST.get('contact', False)

email = request.POST.get('email', False)

address = request.POST.get('address', False)

password = request.POST.get('password', False)

output = "none"

con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '', database = 'multistagecloud',charset='utf8')

with con:

cur = con.cursor()

cur.execute("select username FROM register")

rows = cur.fetchall()

for row in rows:

if row[0] == username:

output = username+" Username already exists"

break

if output == "none":

db\_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '', database = 'multistagecloud',charset='utf8')

db\_cursor = db\_connection.cursor()

student\_sql\_query = "INSERT INTO register(username,password,contact,email,address) VALUES('"+username+"','"+password+"','"+contact+"','"+email+"','"+address+"')"

db\_cursor.execute(student\_sql\_query)

db\_connection.commit()

print(db\_cursor.rowcount, "Record Inserted")

if db\_cursor.rowcount == 1:

output = "Signup process completed. Choose Your Auth Image"

context= {'data':output}

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

**CHAPTER 7**

**SYSTEM TESTING**

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

**7.1 TYPES OF SOFTWARE TESTING**

**7.1.1 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 individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**7.1.2 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 satisfaction, 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.

**7.1.3 SYSTEM TESTING**

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

**7.1.4 FUNCTIONAL TESTING**

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

Functional testing is centered on the following items:

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

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

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

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

The 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.

**7.1.5 WHITE BOX TESTING**

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

**7.1.6 BLACK BOX TESTING**

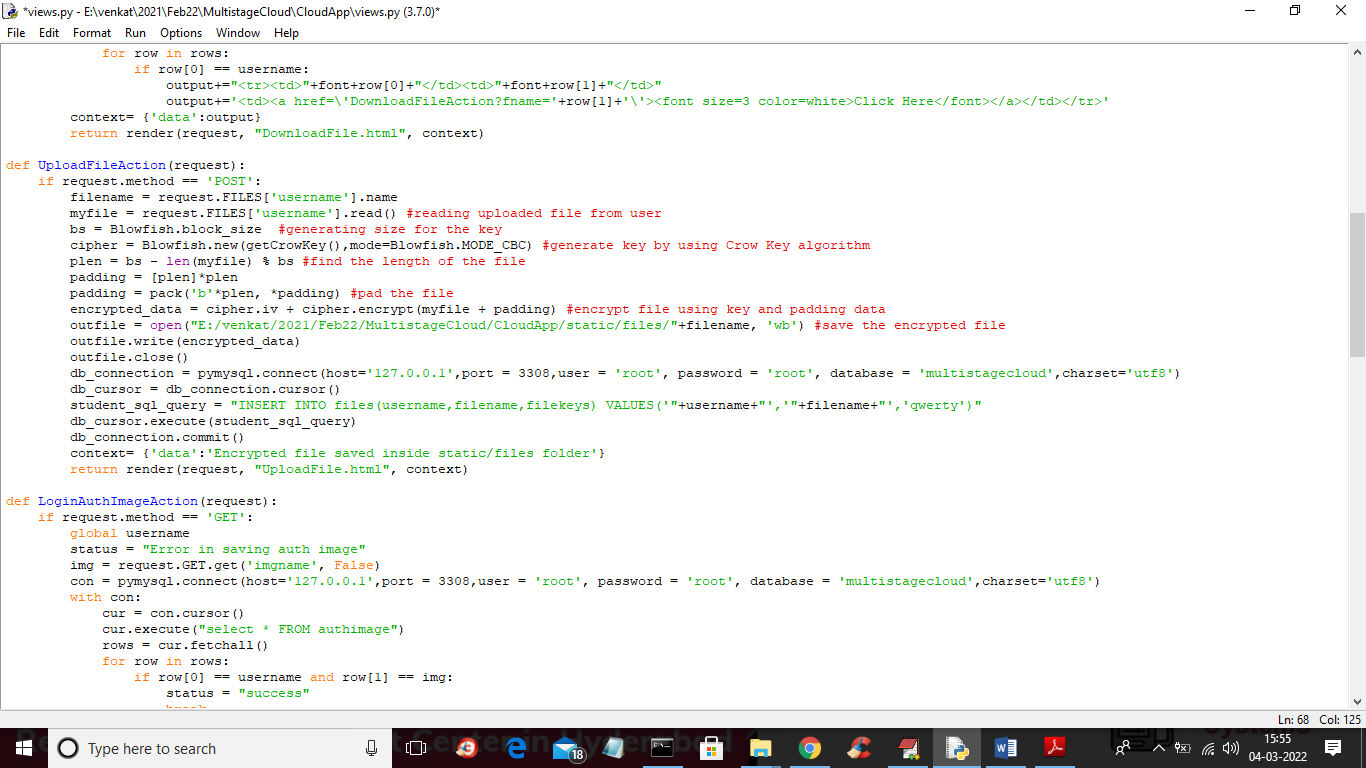
Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification 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.

**7.2 TESTCASES**

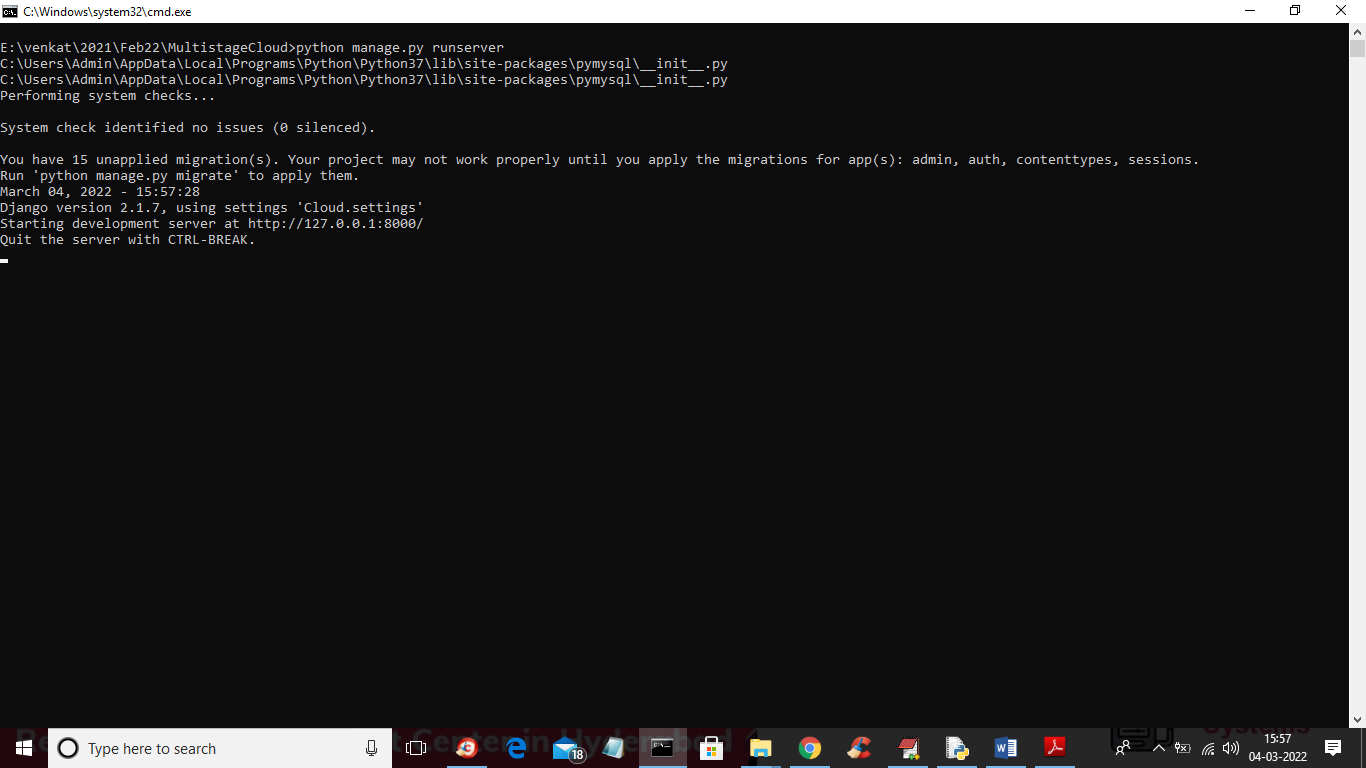
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **TESTCASES** | **EXPECTED RESULT** | **RESULT** | **REMARKS** |
| **1.** | **Register new user with valid details.** | **User should be registered successfully** | **Pass** | **Username already exists / DB error** |
| **2.** | **Register user with existing username.** | **System should display "Username already exists"** | **Fail** | **Directly login into the website.** |
| **3.** | **Login with correct username and password** | **Redirect to Auth Image selection page** | **Pass** | **Incorrect redirection or validation** |
| **4.** | **Login with incorrect username/password** | **Show "Invalid username" message** | **Fail** | **Give correct username and password.** |
| **5.** | **Select valid authentication image** | **Auth image should be saved and cropped** | **Pass** | **Image not saved or cropped properly** |
| **6.** | **Select invalid authentication image** | **Show "Invalid Auth images selected. Please retry"** | **Fail** | **Select correct auth image** |
| **7.** | **Upload valid file** | **File should be encrypted and stored in static/files** | **Pass** | **File not stored or encryption failed** |
| **8.** | **Upload file without selecting one** | **Error message or validation for no file selected** | **Fail** | **Select an image.** |
| **9.** | **Download file uploaded by same user** | **Encrypted file should be decrypted and downloaded correctly** | **Pass** | **Decryption fails / corrupted download** |
| **10.** | **Unauthorized file download attempt** | **File should not download if user is not the owner** | **Fail** | **Cannot access file** |

**CHAPTER 8**

**RESULT**

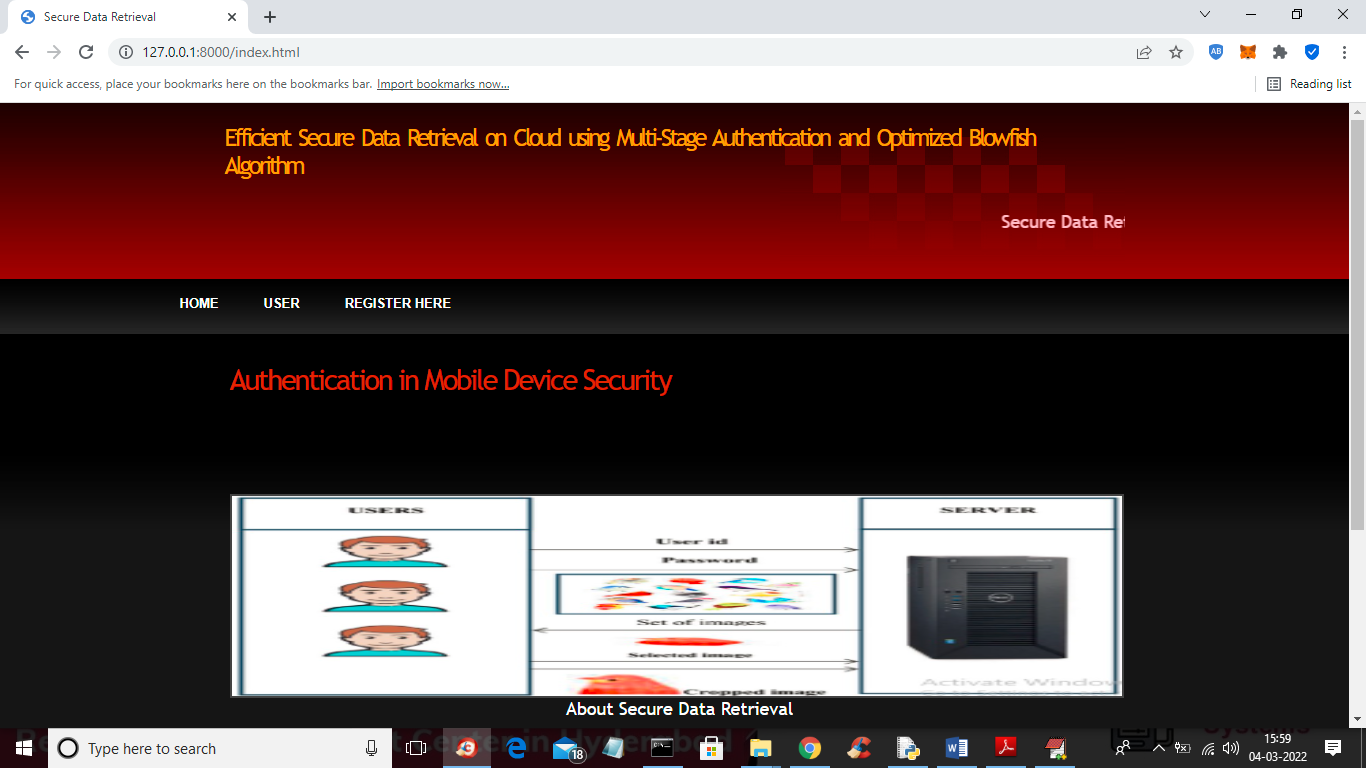


**FIG 8.1: Source Code.**



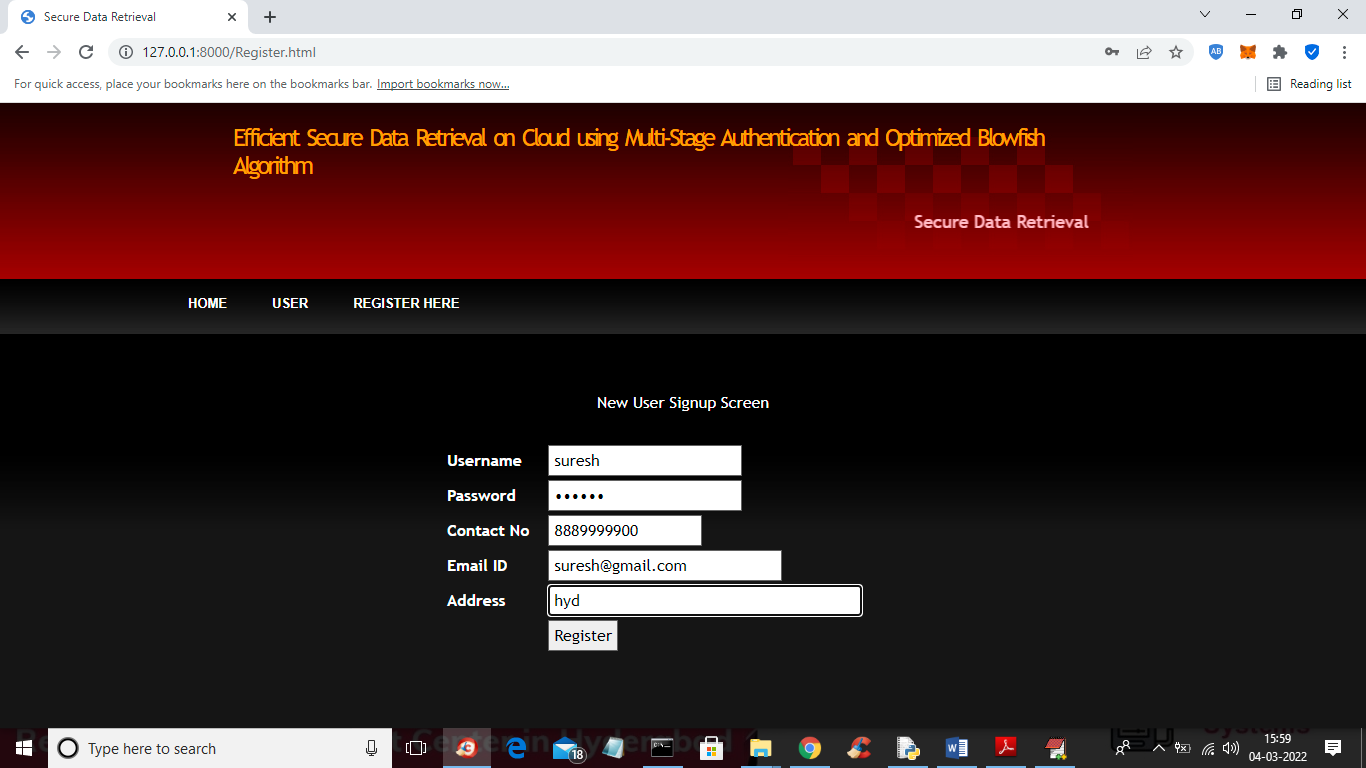
**FIG 8.2: Execution Part**

* In above screen server started and now open browser and enter URL as <http://127.0.0.1:8000/index.html> and press enter key to get below screen



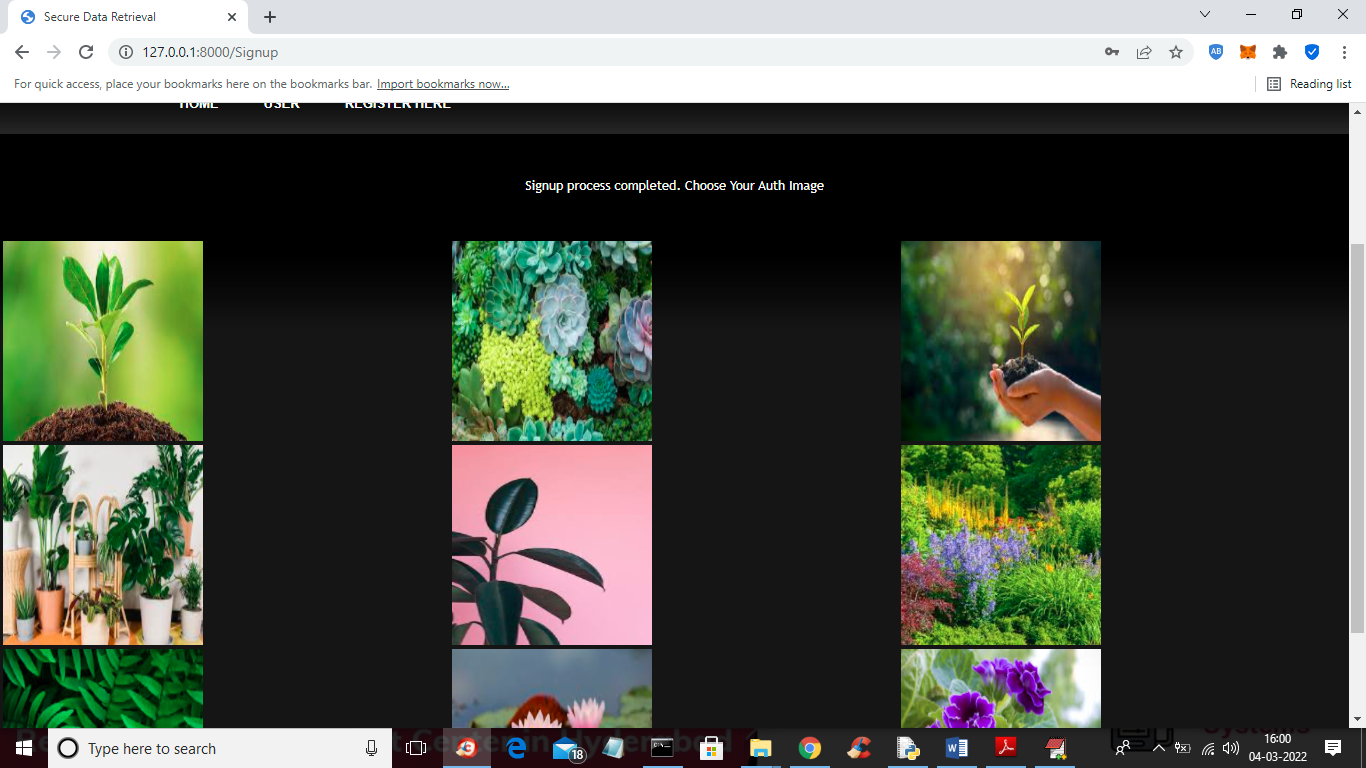
**FIG 8.3:Home Page**

* In above screen click on ‘Register Here’ link to allow user to register



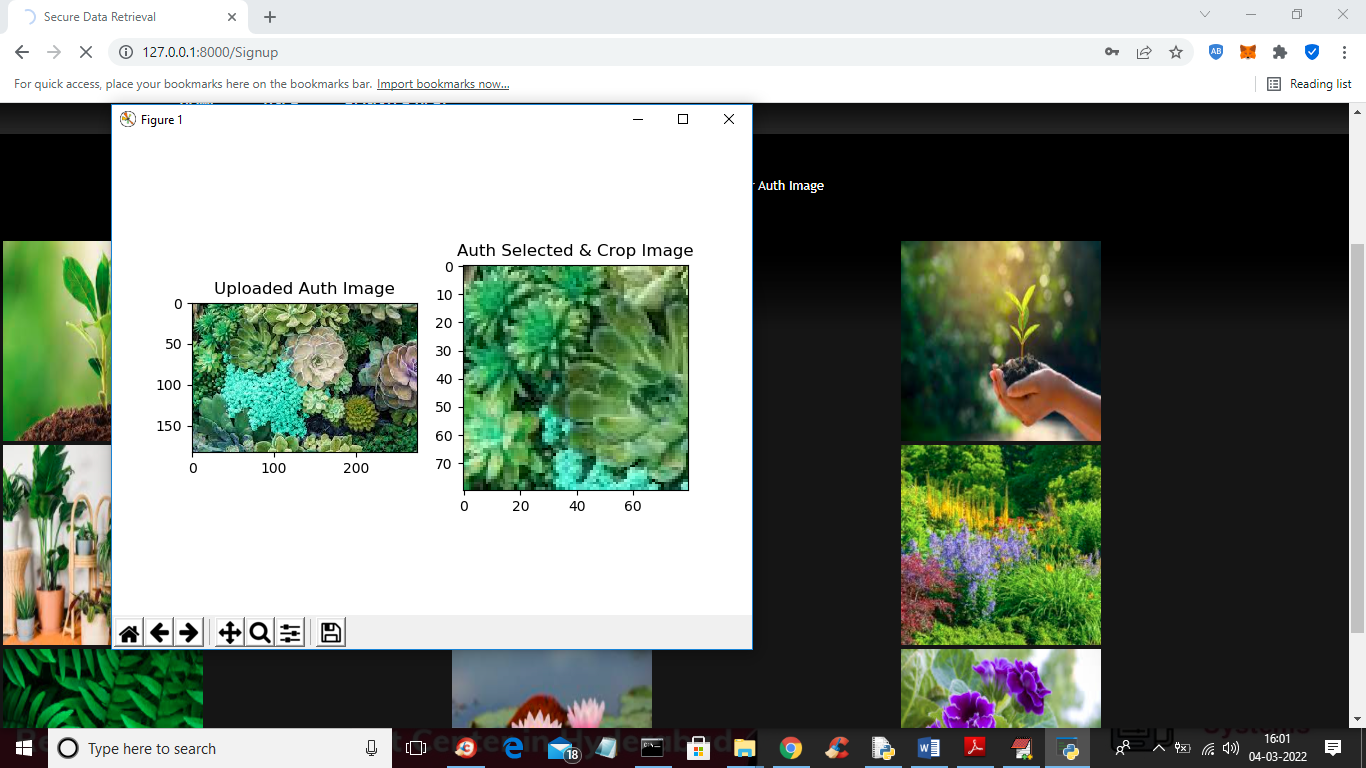
**FIG 8.4: Viewing the details of user**

* In above screen user is enter signup details and then click on ‘Register’ button to get below images screen



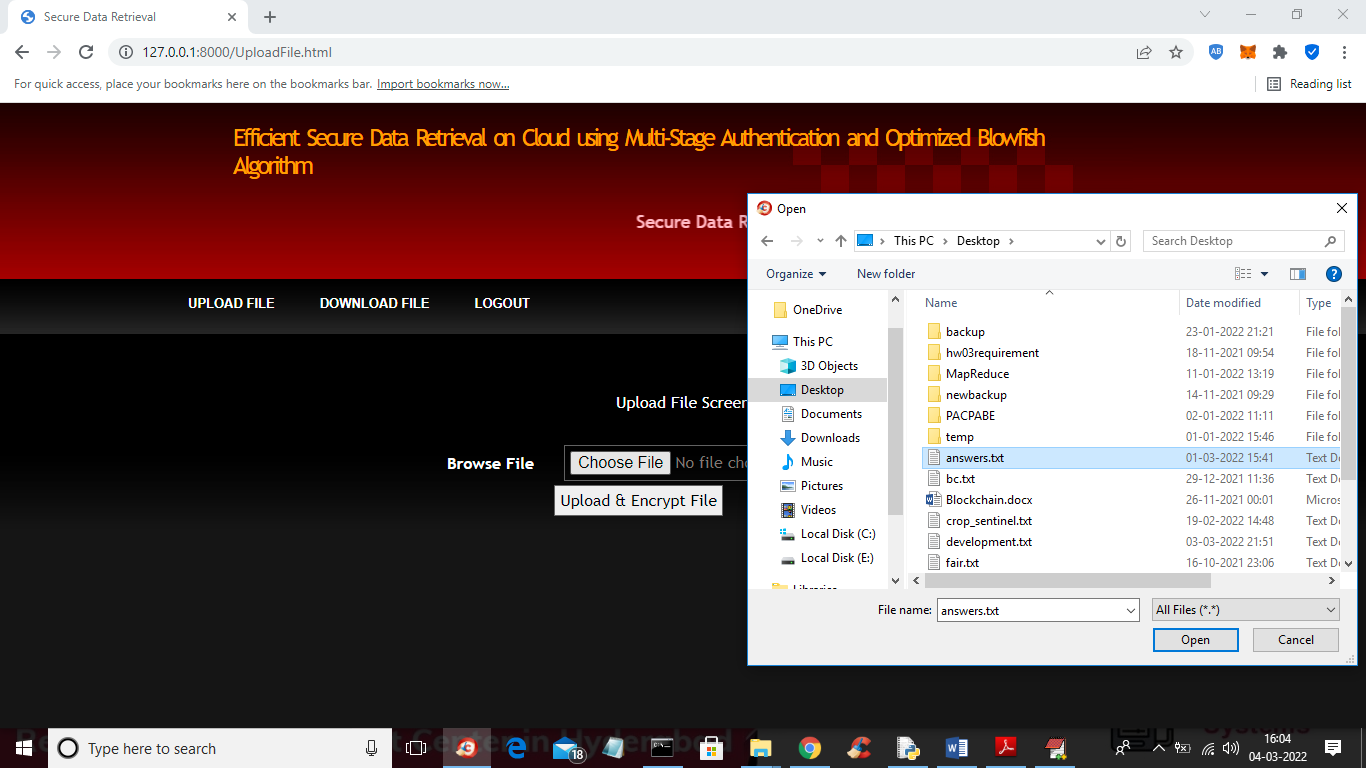
**FIG 8.5: Authorized Image Selection.**

* In above screen user has to click on desired image as second authentication and to get below output



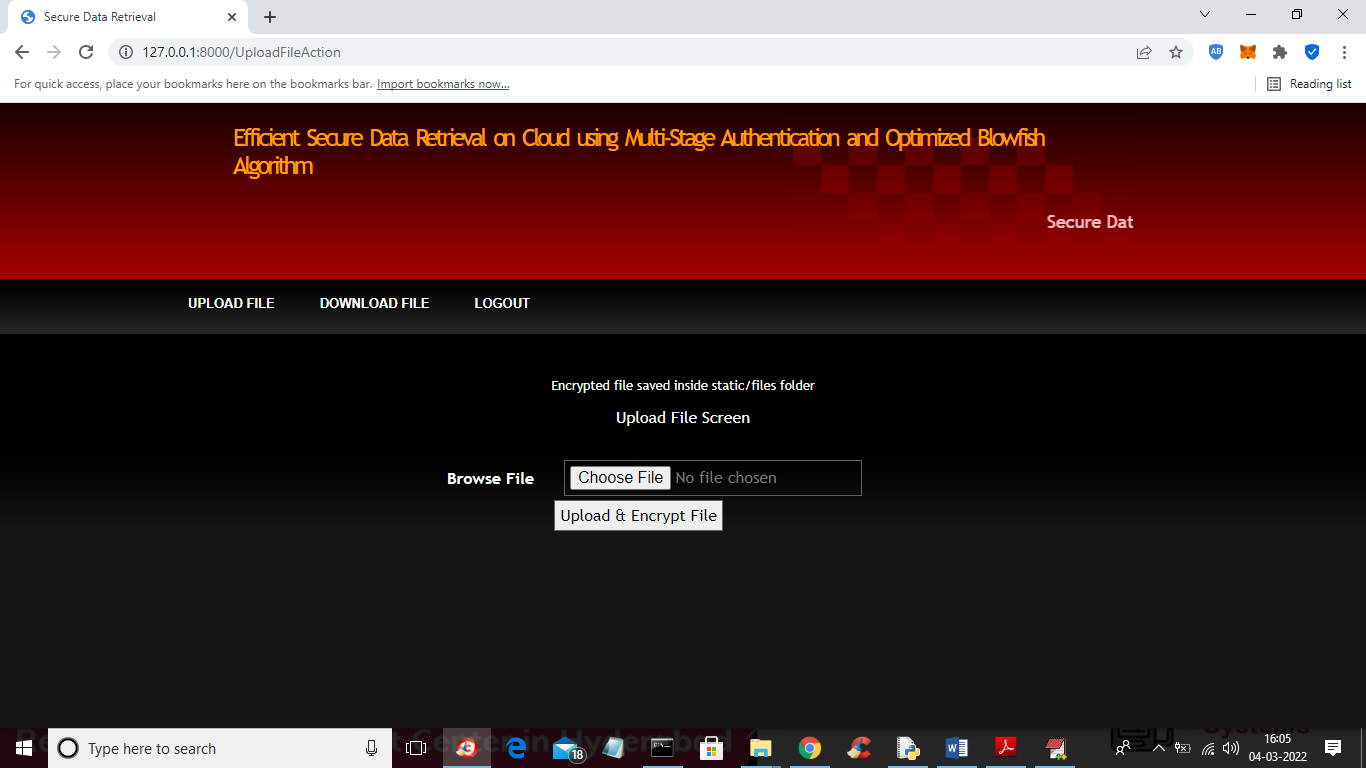
**FIG:8.6: Cropping The Image Selected**

* In above screen user can see selected and cropped image and then close above image to store on server and to get below screen



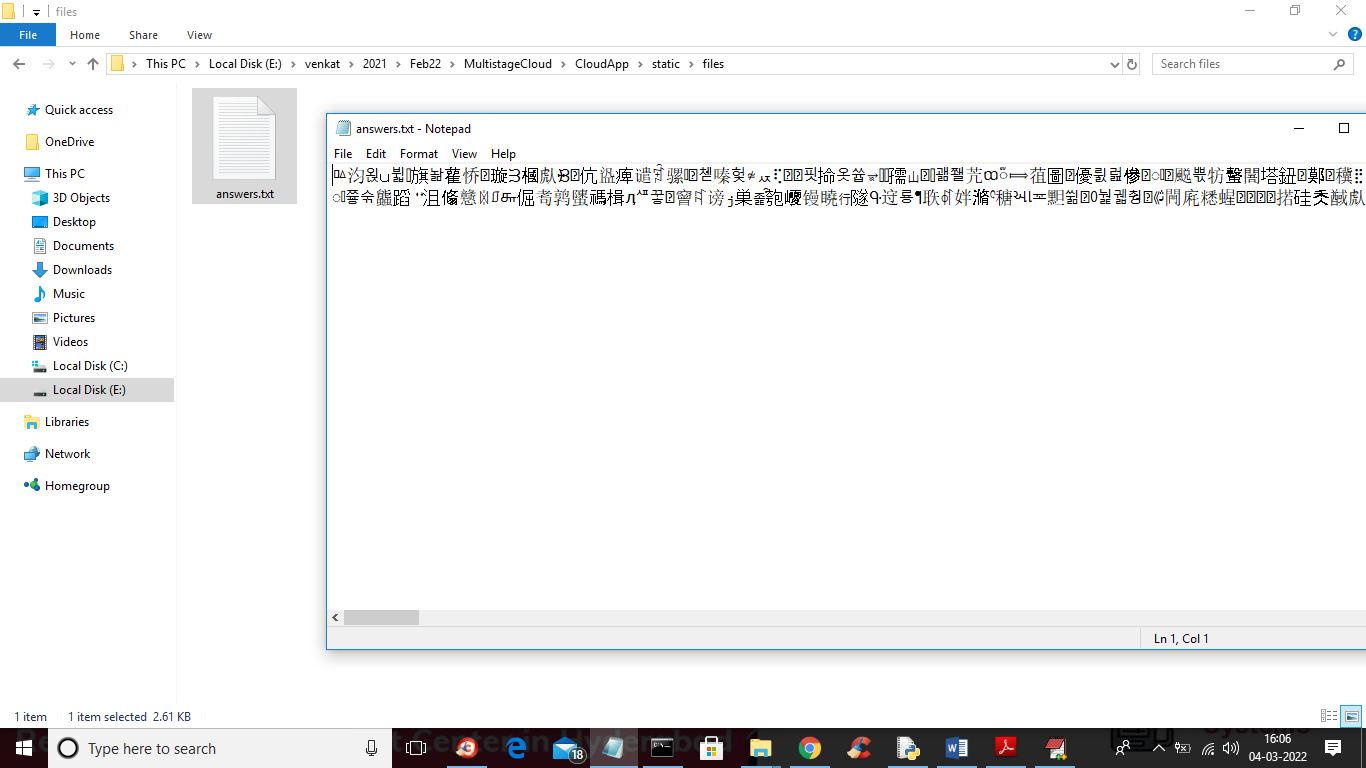
**FIG 8.7:Uploading The File**

* In above screen selecting and uploading file and then click on ‘Open and Upload & encrypt’ button to get below screen



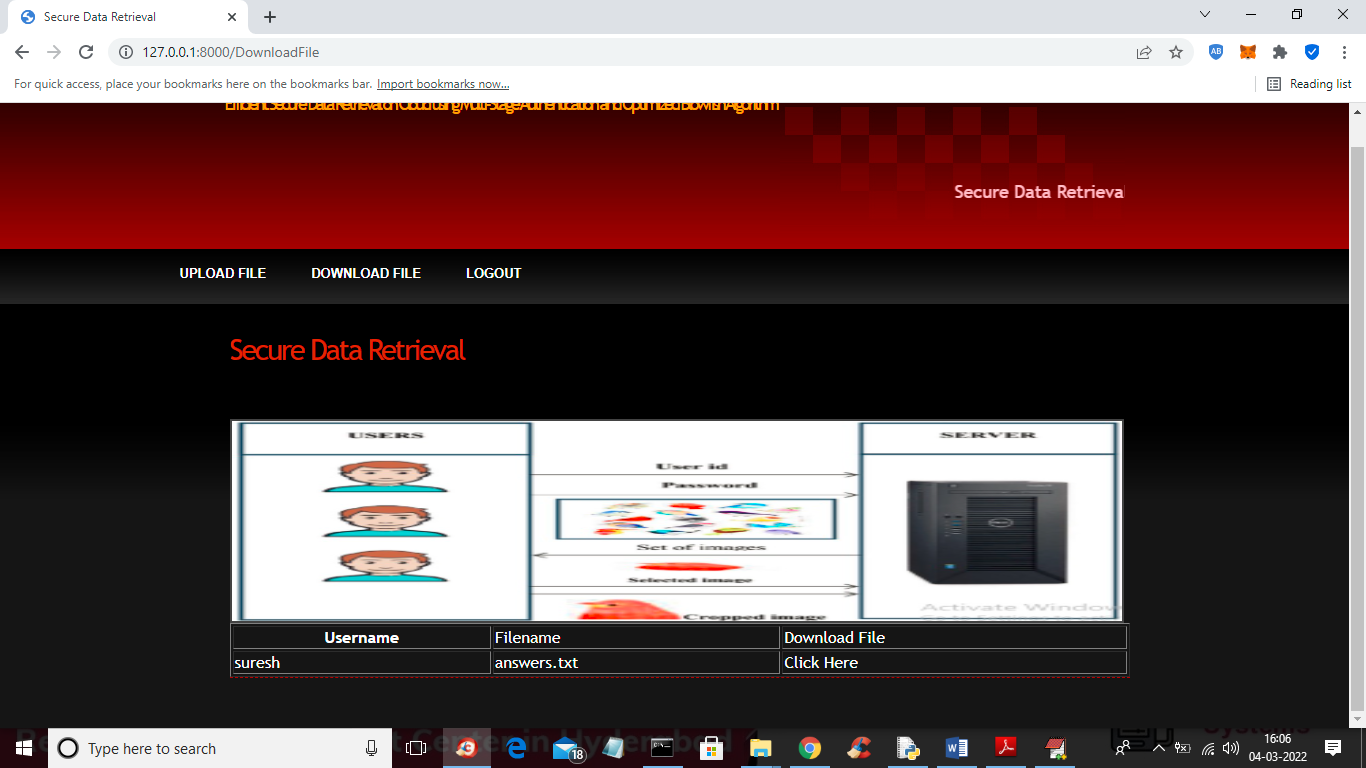
**FIG 8.8: Upload Successful.**

* In above screen we can see file saved in server static folder and in that file we can see files saved in Blowfish encrypted format like below screen



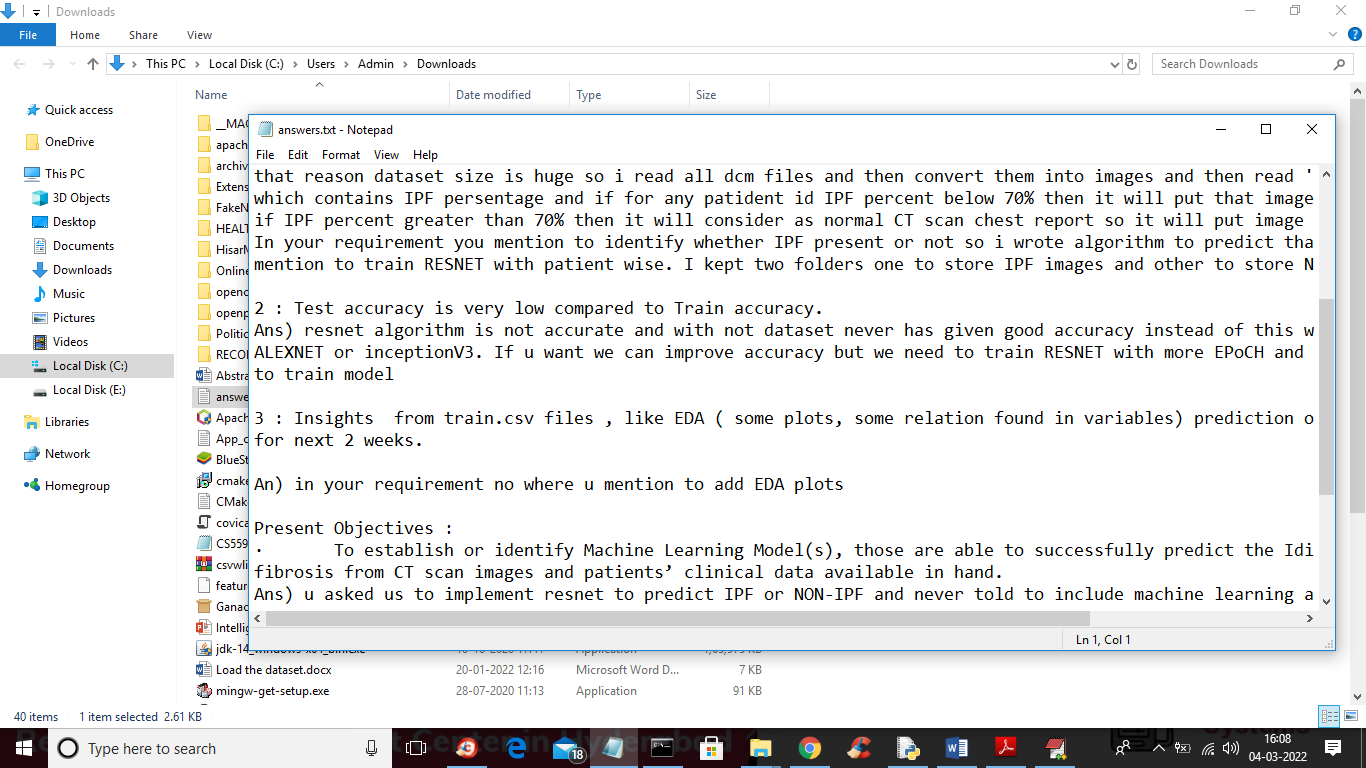
**FIG 8.9:Data Encryption Using Blowfish Algorithm.**

* In above screen we can see file saved in encrypted format and now click on ‘Download File’ link in output screen to get below screen



**FIG 8.10:Uploaded File in the Home Page.**

* In above screen in table we can see all files uploaded by user and then click on ‘Click Here’ link to download that file in decrypted format



**FIG 8.11:File in the Decrypted Format.**

* In above screen we can see file is decrypted and similarly you can upload and download any number

**9.CONCLUSION**

The proposed system effectively addresses prevalent security concerns associated with cloud data storage by integrating multi-stage authentication with an optimized Blowfish encryption algorithm. The use of the Crow Search Algorithm for key generation further fortifies the encryption process, ensuring robust data protection. This comprehensive approach not only enhances the security of data retrieval in cloud environments but also fosters greater trust among users and organizations considering cloud adoption.​

**10.FUTURE SCOPE**

Future research can explore the integration of additional biometric authentication methods to further enhance security. Additionally, the application of other metaheuristic algorithms for key optimization could be investigated to compare performance and security outcomes. Scalability testing in diverse cloud environments would also provide insights into the system's adaptability and robustness.

**11.REFERENCE**

* 1. S. I. Shyla and S. S. Sujatha, "Efficient secure data retrieval on cloud using multi-stage authentication and optimized blowfish algorithm," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, pp. 151–163, 2022.​
  2. Dileep Rao, "Efficient secure data retrieval on cloud using multi-stage authentication and optimized blowfish algorithm," *Semantic Scholar*, 2021.​
  3. S. Immaculate Shyla and S. S. Sujatha, "Utilizing multi-stage authentication and an optimized blowfish algorithm for effective secure data retrieval on cloud computing," *ResearchGate*, 2021.
  4. S. Immaculate Shyla, "Efficient secure data retrieval on cloud using multi-stage authentication and optimized blowfish algorithm," *ORCID*, 2021.​
  5. "Efficient secure data retrieval on cloud using multi-stage authentication and optimized blowfish algorithm," *YouTube*, 2021
  6. "Cloud Multimedia Data Security by Optimization-Assisted Improved Blowfish Algorithm," *World Scientific*, 2024.
  7. "A fourfold-objective-based cloud privacy preservation model with random key generation," *Taylor & Francis Online*, 2024.​
  8. "A secure VM live migration technique in a cloud computing environment using the blowfish cryptographic algorithm," *ACM Digital Library*, 2024.​
  9. "Optimal Key Generation for Data Sanitization and Restoration of Cloud Security in Finance," *World Scientific*, 2020.​
  10. "Security in Cloud Computing Using Blowfish Algorithm," *Kalahari Journals*, 2022

.