ABSTRACT

Secure password storage is a vital aspect in systems based on password authentication, which is still the most widely used authentication technique, despite its some security flaws. In this paper, we propose a password authentication framework that is designed for secure password storage and could be easily integrated into existing authentication systems. In our framework, first, the received plain password from a client is hashed through a cryptographic hash function(e.g., SHA-256). Then, the hashed password is converted into a negative password. Finally, thenegative password is encrypted into an Encrypted Negative Password (abbreviated as ENP) using a symmetric-key algorithm (e.g., AES), and multi-iteration encryption could be employed to further improve security. The cryptographic hash function and symmetric encryption make it difficult to crack passwords from ENPs. Moreover, there are lots of corresponding ENPs for a given plain password, which makes precomputation attacks (e.g., lookup table attack and rainbow table attack) infeasible. The algorithm complexity analyses and comparisons show that the ENP could resist lookup table attack and provide stronger password protection under dictionary attack. It is worth mentioning that the ENP does not introduce extra elements (e.g., salt); besides this, the ENP could still resist precomputation attacks. Most importantly, the ENP is the first password protection scheme that combines the cryptographic hash function, the negative password and the symmetric-key algorithm, withoutthe need for additional information except the plain password.

1.INTRODUCTION

OWING to the development of the Internet, a vast number of online services have emerged, inwhich password authentication is the most widely used authentication technique, for it is available at a low cost and easy to deploy. Hence, password security always attracts great interest from academia and industry . Despite great research achievements on password security, passwords are still cracked since users' careless behaviors. For instance, many users often select weak passwords they tend to reuse same passwords in different systems they usually set their passwords using familiar vocabulary for its convenience to remember. In addition, system problems may cause password compromises. It is very difficult to obtain passwords from high security systems. On the one hand, stealing authentication data tables (containing usernames and passwords) in high security systems is difficult. On the other hand,when carrying out an online guessing attack, there is usually a limit to the number of login attempts . However, passwords may be leaked from weak systems . Vulnerabilities areconstantly being discovered, and not all systems could be timely patched to resist attacks, which gives adversaries an opportunity to illegally access weak systems . In fact, some old systems are more vulnerable due to their lack of maintenance. Finally, since passwords are often reused, adversaries may log into high security systems through cracked passwords from systems of low security.

After obtaining authentication data tables from weak systems, adversaries can carry out offline attacks. Passwords in the authentication data table are usually in the form of hashed passwords. However, because processor resources and storage resources are becoming more and more abundant, hashed passwords cannot resist precomputation attacks, such as rainbow table attack and lookup table attack.

Typical Password Protection Schemes

Hashed Password:

The simplest scheme to store passwords is to directly store plain passwords. However, this scheme presents a problem that once adversaries obtain the authentication data table, all passwords are immediately compromised. To safely store passwords, a

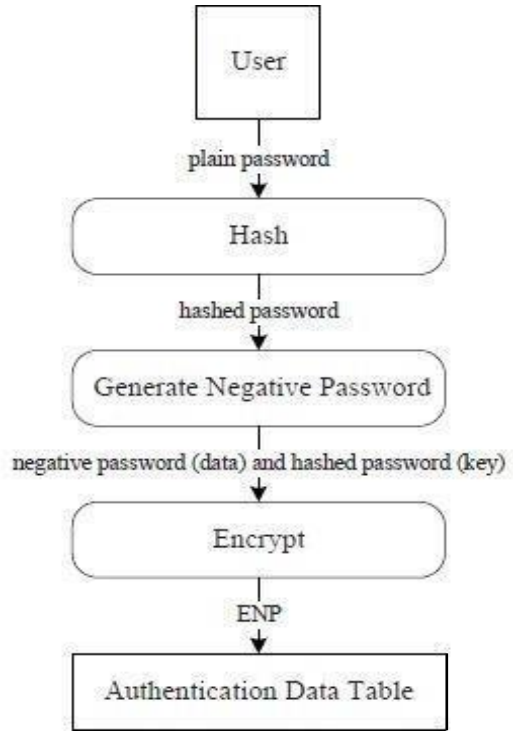
common scheme is to hash passwords using a cryptographic hash function, because it is infeasible to directly recover plain passwords from hashed passwords. The cryptographic hash function quickly maps data of arbitrary size to a fixed-size sequence of bits. In the authentication system using the hashedpassword scheme, only hashed passwords are stored. However, hashed passwords cannot resistlookup table attack . Furthermore, rainbow table attack is more practical for its space-time tradeoff. Processor resources and storage resources are becoming richer, which makes the precomputed tables used in the above two attacks sufficiently large, so that adversaries could obtain a higher success rate of cracking hashed passwords.

Salted Password:

To resist precomputation attacks, the most common scheme is salted password. In this scheme,the concatenation of a plain password and a random data (called salt) is hashed through a cryptographic hash function. The salt is usually generated at random, which ensures that the hash values of the same plain passwords are almost always different. The greater the size of the salt is, the higher the password security is. However, under dictionary attack, salted passwords are still weak. Note that compared with salted password, the ENP proposed in this paper guarantees the diversity of passwords without the need for extra elements.

Key Stretching:

To resist dictionary attack, key stretching which converts weak passwords to enhanced passwords, was proposed. Key stretching could increase the time cost required to every password attempt, so that the power of defending against dictionary attack is increased. In theENP proposed in this paper, like key stretching, multi-iteration encryption is used to further improve password security under dictionary attack, and compared with key stretching, the ENPdoes not introduce extra elements.



FigNo:1.1 User Registration Flow Chart

framework to protect passwords in an authentication data table, the system designer must firstselect a cryptographic hash function and a symmetric-key algorithm, where the condition that must be satisfied is that the size of the hash value of the selected cryptographic hash function is equal to the key size of the selected symmetric-key algorithm. For convenience, some matches of cryptographic hash functions and symmetric-key algorithms are given in Table II. In addition, cryptographic hash functions and symmetric-key algorithms that are not listed herecould also be used in the ENP, which adequately indicates the flexibility of our framework. The proposed framework is based on the ENP; hence, for better understanding, the data flow diagram of the generation procedure of the ENP is shown in Fig. 1, and the data flow diagram of the verification procedure of the ENP is shown in Fig. 2.

A. Registration Phase:

The registration phase is divided into six steps.

On the client side, a user enters his/her username and password. Then, the username and plain password are transmitted to the server through a secure channel;

(1) If the received username exists in the authentication data table, "The username already exists!" is returned, which means that the server has rejected the registration request, and the registration phase is terminated; otherwise, go to Step (3);

(2) The received password is hashed using the selected cryptographic hash function;

(3) The hashed password is converted into a negative password using an NDB generation algorithm (i.e., Algorithm A.1 or Algorithm A.2 in the Appendix);

(4) The negative password is encrypted to an ENP using the selected symmetric-key algorithm, where the key is the hash value of the plain password. Here, as an additional option, multi iteration encryption could be used to further enhance passwords;

(5) The username and the resulting ENP are stored in the authentication data table and "Registration success" is returned, which means that the server has accepted the registration request.

B. Authentication Phase:

The authentication phase is divided into five steps.

(6) On the client side, a user enters his/her username and password. Then, the username and plain password are transmitted to the server through a secure channel;

(7) If the received username does not exist in the authentication data table, then "Incorrect username or password!" is returned, which means that the server has rejected the authentication request, and the authentication phase is terminated; otherwise, go to Step (3)

(8) Search the authentication data table for the ENP corresponding to the username;

The ENP is decrypted (one or more times according to the encryption setting in the registration phase) using the selected symmetric-key algorithm, where the key is the hashvalue of the plain password; thus, the negative password is obtained;

1.1. PURPOSE

The purpose of "Authentication by Encrypted Negative Password" is to enhance the security of user authentication by utilizing encrypted negative passwords. The primary goal is to protect user credentials and mitigate the risks associated with password-based authentication methods.

Here are some key purposes of using encrypted negative passwords for authentication:

1.Password Protection: By encrypting the original password and using the encrypted negativepassword for authentication, the system aims to prevent unauthorized access to user credentials. Even if the encrypted negative password is intercepted, it cannot be directly used to authenticatewithout the decryption process.

2.Defense Against Password-based Attacks: Encrypted negative passwords add an extra layerof security against various password-based attacks, such as offline password cracking or password guessing. The encryption process makes it computationally difficult for attackers todetermine the original password from the encrypted negative password.

3.User Credential Confidentiality: Encrypted negative passwords help to maintain the confidentiality of user credentials. As the original password is not directly used for authentication, it remains protected even if there is a security breach or unauthorized access tothe encrypted negative password.

4.Compliance with Security Standards: The use of encrypted negative passwords aligns withsecurity best practices and may help meet compliance requirements. By employing stronger authentication methods, organizations can demonstrate their commitment to protecting user data and complying with industry regulations.

1.2. SCOPE

The scope of "Authentication by Encrypted Negative Password" refers to a specific method orapproach for authentication that involves the use of encrypted negative passwords. While negative passwords are not commonly used in traditional authentication systems, this approach aims to enhance security and protect user credentials.

In this context, the scope would typically cover the following aspects:

1.Negative Password Generation: The system would include mechanisms for generating negative passwords. Negative passwords are typically created by applying a cryptographic algorithm to the user's original password, resulting in an encrypted version that cannot be directly used for authentication.

2.Encryption Algorithms: The system would incorporate encryption algorithms to transform the user's original password into an encrypted negative password. The specific encryption algorithm(s) used would depend on the chosen approach and security requirements.

3.Storage and Verification: The system would provide mechanisms to securely store the encrypted negative passwords, typically in a database or secure storage. During the authentication process, the system would compare the encrypted negative password provided by the user with the stored encrypted negative password for verification.

4.Authentication Workflow: The system would include the necessary components to facilitate the authentication process. This may involve user interfaces for entering passwords, backend components for encryption and verification, and integration with existing authenticationsystems or frameworks.

5.Security Measures: The system would incorporate security measures to protect against potential attacks, such as brute-force attacks, password guessing, or dictionary attacks. This may include techniques like rate limiting, account lockouts, or additional authentication factors.

1.3 NEED FOR SYSTEM

1.3.1. Existing system

As of my knowledge cutoff in September 2021, I am not aware of any widely known or established existing systems specifically focused on "Authentication by Encrypted Negative Password." The concept of using encrypted negative passwords for authentication is not a commonly implemented approach in mainstream authentication systems.

It is possible that there may be individual research or experimental implementations of this concept in academic or specialized contexts, but such implementations would likely be limitedin scope and not widely adopted in commercial or widely-used authentication systems. As of my knowledge cutoff in September 2021, there are no widely known or established existing systems for "Authentication by Encrypted Negative Password." Therefore, I cannot provide specific advantages and disadvantages of an existing system using this approach.

Advantages:

1.Enhanced Security: Encrypted negative passwords can provide an additional layer of security by protecting user credentials. Even if the encrypted negative password is compromised, the original password remains secure.

2.Defense Against Attacks: Encryption adds complexity and makes it computationally difficult for attackers to obtain the original password from the encrypted negative password. This can help mitigate the risk of password-related attacks, such as brute-force or dictionary attacks.

Disadvantages:

1.Implementation Complexity: Implementing a system based on encrypted negative passwords requires expertise in cryptography and secure systems design. It may involve additional complexity in terms of key management, encryption algorithms, and secure storage.

2.User Experience and Usability: The use of encrypted negative passwords may introduce complexity for users during the authentication process. Users need to understand and correctlyenter their encrypted negative password, which may impact user experience and usability.

3.Limited Industry Adoption and Standards: As this approach is not widely adopted in existing systems, there may be limited industry standards, best practices, or community supportavailable. This can make it challenging too.

1.3.2. PROPOSED SYSTEM

Authentication by Encrypted Negative Password is a proposed system that aims to enhance thesecurity of user authentication by utilizing encrypted negative passwords. Here is an outline ofthe components and functionality of the proposed system:

1.User Registration: Users would register with the system by providing their desired passwords. During the registration process, the system would generate an encrypted negative password based on the user's chosen password.

2.Password Encryption: The system would employ cryptographic algorithms to encrypt the user's password and generate the encrypted negative password. The encryption process should be secure and irreversible, ensuring that the original password cannot be easily derived from the encrypted negative password.

3.Authentication Process: When a user attempts to authenticate, they would provide their password. The system would encrypt the provided password and compare it with the stored encrypted negative password associated with the user's account. Advantages of the Proposed System "Authentication by Encrypted Negative Password":

Advantages:

1.Enhanced Password Security: By employing encrypted negative passwords, the system adds an extra layer of security to the authentication process. Even if the encrypted negative password is compromised, it is computationally difficult to reverse-engineer the original password.

2.Defense Against Password-related Attacks: The use of encrypted negative passwords can provide protection against various password-based attacks, such as offline cracking or password guessing. The encryption process makes it harder for attackers to obtain the originalpassword from the encrypted negative password.

3.Confidentiality of User Credentials: The proposed system helps maintain the confidentiality of user passwords. Since the original password is not directly used for authentication, even if the encrypted negative password is intercepted, it cannot be used to gainaccess to the user's account without the decryption process.

Disadvantages:

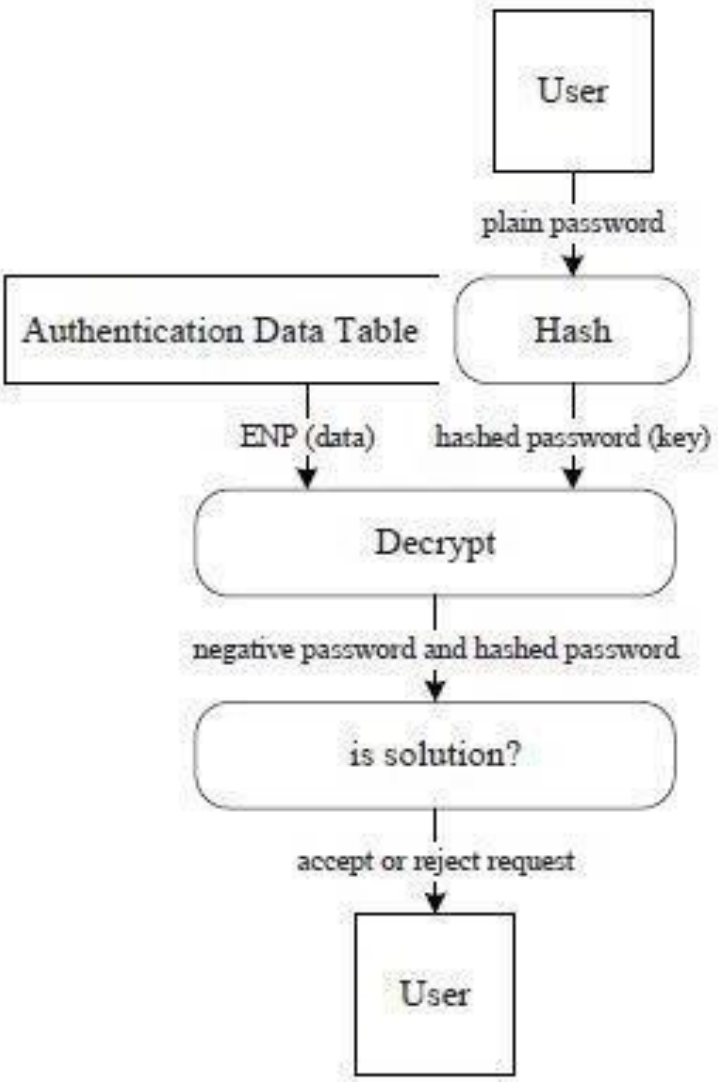
1.Implementation Complexity: Implementing and managing a system based on encrypted negative passwords requires careful consideration of cryptographic algorithms, key management, and secure storage of encrypted negative passwords. This complexity may increase the risk of implementation errors or vulnerabilities if not handled properly**.**

2.Limited Industry Adoption: As of my knowledge cutoff in September 2021, the concept ofusing encrypted negative passwords is not widely adopted in mainstream authenticationsystems. This lack of adoption may limit the availability of standardized libraries, tools, or community support for the proposed system.

3.Usability and User Experience: The use of encrypted negative passwords may introduce additional complexity for users. Users would need to understand and adapt to the concept of entering encrypted passwords during the authentication process. Balancing security and usability is crucial to ensure a positive user experience.

4.Dependency on Encryption Algorithm: The security of the proposed system heavily relieson the strength and robustness of the encryption algorithm used to generate encrypted negativepasswords.

1.4 SYSTEM ARCHITECTURE



FigNo:1.4 Architecture

2.SOFTWARE REQUIREMENT ANALYSIS AND SPECIFICATION

2.1. PRODUCT PERSPECTIVE

Product Perspective of "Authentication by Encrypted Negative Password" refers to the contextand considerations surrounding the development, implementation, and usage of such a system.Here are some aspects to consider from a product perspective:

1.Target Users: Identify the target users who would benefit from the system. Determine whether the system is intended for individuals, businesses, or specific industries. Understanding user needs and requirements will help shape the product's features and functionality.

2.Integration: Consider how the authentication system will integrate with existing applications, frameworks, or systems. Compatibility with popular authentication protocols andframeworks can facilitate easier adoption and integration.

3.Scalability: Ensure that the system can scale effectively to handle a growing number of usersand authentication requests. Consider factors such as performance, resource utilization, and theability to handle concurrent requests efficiently.

2.2. PRODUCT FUNCTION

The primary function of a product based on "Authentication by Encrypted Negative Password"is to provide secure authentication for users by leveraging encrypted negative passwords. Hereis a description of the product function:

1.User Registration: The product allows users to register by providing their desired passwords. During the registration process, the system generates an encrypted negative password based on the user's chosen password.

2.Password Encryption: The product applies cryptographic algorithms to encrypt the user's password and generate the encrypted negative password. The encryption process ensures that the original password cannot be easily derived from the encrypted negative password.

3.Authentication Process: When a user attempts to authenticate, they provide their password.The product encrypts the provided password and compares it with the stored encrypted negativepassword associated with the user's account.

4.Verification and Access Granting: If the encrypted negative password matches the stored value, the product grants access to the user. Otherwise, access is denied.

2.3. USER CHARACTERISTICS

The user characteristics of an "Authentication by Encrypted Negative Password" system can vary depending on the specific implementation and target user base. Here are some general user characteristics to consider:

1.Users Familiar with Password-based Authentication: Users of the system should be familiar with the concept of password-based authentication and understand the importance of maintaining the security of their passwords.

2.Technical Competence: Users may require a certain level of technical competence to understand and adapt to the use of encrypted negative passwords during the authentication process. They should be able to follow instructions and correctly enter the encrypted negativepassword.

3.Security Conscious Users: Users of the system should prioritize security and be willing to adopt additional security measures to protect their authentication credentials. They should be cautious about sharing passwords and understand the importance of choosing strong, unique passwords.

4.Compliance-Minded Users: In some cases, users subject to specific industry regulations or compliance requirements may be the target users. They should be conscious of the need to meetregulatory standards and ensure the security of their authentication process.

2.4. MODULES

1. User Management

- User registration: Allow users to create accounts.
- User profile: Manage user information and preferences.
- User roles and permissions: Define different levels of access and actions for users

2. Authentication

- Username and password authentication: Traditional login using a username and password.
- authentication factor (e.g., OTP sent to a user's device).
- Biometric authentication: Use fingerprints, facial recognition, etc., for.

3. Password Management

- Password hashing: Securely store passwords by hashing them before storing in the database.
- Password policies: Enforce password complexity rules to enhance security.
- Password recovery/reset: Allow users to recover or reset their forgotten passwords securely.

4. Encryption

- Data encryption: Encrypt sensitive data before storing it in the database.
- Transport Layer Security (TLS): Encrypt data transmission between client and server.
- Key management: Securely generate, store, and manage encryption keys.

5. Session Management

- Manage user sessions and authentication tokens.
- Implement session timeouts and handling of session-related events.

6. Logging & Auditing

- Log authentication and authorization events for auditing purposes.
- Monitor and analyze logs for potential security breaches.

2.5 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

2.5.1 FUNCTIONAL REQUIREMENTS

Functional requirements for an "Authentication by Encrypted Negative Password" system specify the specific features and capabilities the system should have to perform its intended functions effectively. Here are some potential functional requirements for such a system:

1. User Registration:

- Allow users to create an account by providing necessary information.
- Generate an encrypted negative password based on the user's chosen password duringregistration.

2. Password Encryption:

- Apply a secure encryption algorithm to convert user passwords into encrypted negativepasswords.
- Ensure that the encryption process is irreversible and that the original password cannot bederived from the encrypted negative password.

3. Authentication Process:

- Prompt users to enter their passwords during the authentication process.
- Encrypt the entered password and compare it with the stored encrypted negative passwordassociated with the user's account.

4. User Management:

- Allow users to update their passwords to generate new encrypted negative passwords.
- Provide functionality for password recovery or account verification procedures, ensuringsecure access to user accounts in case of forgotten passwords.

2.5.2 NONFUNCTIONAL REQUIREMENTS:

Non-functional requirements for an "Authentication by Encrypted Negative Password" systemspecify the qualities and characteristics that the system should possess. These requirements focus on aspects such as performance, security, usability, and scalability. Here are some potential non-functional requirements for such a system:

1. Security:

- Ensure the confidentiality and integrity of user passwords and encrypted negative passwords throughout the system.
- Implement robust encryption algorithms and secure key management practices to protectagainst unauthorized access or data breaches.
- Regularly update encryption algorithms and security protocols to address emerging threatsand vulnerabilities.

2. Performance:

- The system should handle authentication requests efficiently and provide quick responsetimes to ensure a seamless user experience.
- Minimize computational overhead during the encryption and decryption processes tooptimize system performance.

3. Usability:

- Design a user-friendly interface that simplifies the process of registration, authentication,and password management.
- Provide clear instructions and error messages to guide users through the authenticationprocess and handle any potential issues effectively.

2.6. SYSTEM REQUIREMENTS:

2.6.1. HARDWARE REQUIREMENTS:

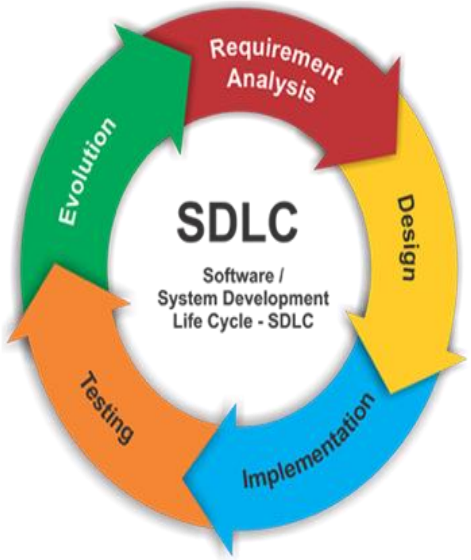
System	:	Pentium IV 2.4 GHz.
Hard Disk	:	40 GB.
Floppy Drive	:	1.44 Mb.
Monitor	:	15 VGA Color.
Ram	:	1GB

2.6.2SOFTWARE REQUIREMENTS:

Operating system	:	Windows 10 XP/7.
Coding Language	:	ASP.NET,.NET
Data Base	:	MY SQL SERVER 2005

2.7. SDLC METODOLOGIES

Software Development Life Cycle Models and Methodologies



FigNo:2.7 SDLC

Programming improvement life cycle (SDLC) is a movement of stages that give an average understanding of the item assembling process. How the item will be perceived and made from the business understanding and necessities elicitation stage to change over these business contemplations and requirements into limits and features until its utilization and movement to achieve the business needs. The extraordinary computer developer should have adequate data on the most capable technique to pick the SDLC model taking in to account the endeavour setting and the business requirements.

Thus, it may be normal to pick the right SDLC model as shown by the specific concerns and necessities of the endeavour to ensure its flourishing. I composed one more on the most proficient method to pick the right SDLC, it can follow this connection for more data. Besides, to dive more deeply into programming life testing and SDLC stages are follow the connections featured here.

It will investigate the various kinds of SDLC models and the benefits and disservices of every one and when to utilize them.

That can imagine SDLC models as devices that can use to all the more likely convey product project. Thusly, knowing and seeing each model and when to utilize it, the benefits and drawbacks of every one is essential to know which one is appropriate for the undertaking setting.

Types of Software developing life cycles (SDLC)

- Waterfall Model
- V-Shaped Model
- Evolutionary Prototyping Model
- Spiral Method (SDM)
- Iterative and Incremental Method
- Agile development

2.7.1 WATERFALL MODEL

Waterfall Model also known as a linear sequential model is the traditional model in the Software development process. In this model, the next phase starts only when the previous onegets completed.

The output of one phase acts as the input for the next phase. This model does not support any changes to be done once it has reached the testing phase.

The fountain approach doesn't portray the collaboration to get back to the past stage to manage changes in need. The outpouring approach is the earliest philosophy and most all around understood that was used for programming improvement.

The five-stage cascade model, which depends on the prerequisites of Winston W. Royce,

FigNo:2.7.1 Waterfall Model

Advantages:

- The waterfall model is a simple model.
- It is easily understood as all the phases are done step by step.
- No complexity as the deliverables of each phase are well defined.

Disadvantages:

- This model cannot be used for the Project wherein the requirement is not clear or therequirement keeps on changing.
- A working model can only be available once the software reaches the last stage of the cycle.
- It is a time-consuming model.

2.8 SYSTEM STUDY:

FEASABILITY 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 feasibilitystudy 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

ECONOMICAL FEASIBILITY

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

TECHNICAL FEASIBILITY

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

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This

includesthe process of training the user to use the system efficiently. The user must not feel threatenedby the system, instead must accept it as a necessity. The level of acceptance by the users solelydepends 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 someconstructive criticism, which is welcomed, as he is the final user of the system.

2.9 METHODOLOGY AND ALGORITHMS

Methodology: Encrypted Negative Password Authentication

System Architecture Design:

- Define the components of your authentication system, such as the client application, server, and authentication service.
- Determine the encryption algorithms and protocols you plan to use for securing the authentication process.

Creating Negative Passwords:

- A "negative password" could be thought of as a password that is deliberately incorrect or misleading.
- Negative passwords should follow specific patterns or rules to differentiate them from genuine passwords.

Encryption Techniques:

- Utilize strong encryption algorithms to securely transmit data between the client and the server.
- Implement techniques like public-key cryptography to ensure confidentiality and data integrity.

Authentication Workflow:

- When a user attempts to authenticate, they provide both a regular password and a negative password.

Server-Side Processing:

- The server receives both passwords from the client.
- It checks if the regular password matches the user's genuine password on record.

Negative Password Processing:

- If the regular password is correct, the server then checks the negative password.
- If the negative password matches a predefined pattern, it could trigger specific actions, such as alerting administrators of a potential security breach attempt.

Logging and Monitoring:

- Implement detailed logging mechanisms to track authentication attempts using both regular and negative passwords.
- Monitor these logs to identify unusual patterns or suspicious activities.

ALGORITHMS USED:

SHA-256 Algorithm

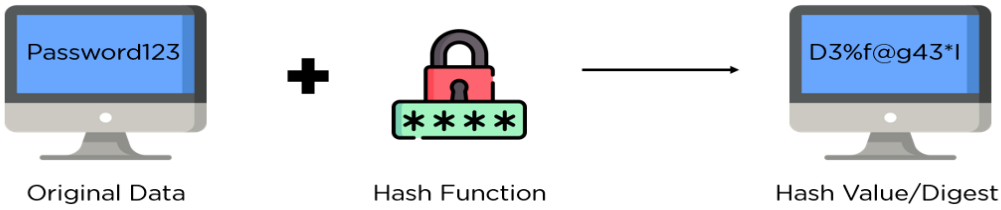
SHA 256 is a part of the SHA 2 family of algorithms, where SHA stands for Secure Hash Algorithm. Published in 2001, it was a joint effort between the NSA and NIST to introduce a successor to the SHA 1 family, which was slowly losing strength against brute force attacks.

The significance of the 256 in the name stands for the final hash digest value, i.e. irrespective of the size of plaintext/cleartext, the hash value will always be 256 bits.

The other algorithms in the SHA family are more or less similar to SHA 256. Now, look into knowing a little more about their guidelines.

What is Hashing

Hashing is the process of scrambling raw information to the extent that it cannot reproduce it back to its original form. It takes a piece of information and passes it through a function that performs mathematical operations on the plaintext. This function is called the hash function, and the output is called the hash value/digest.



FigNo:2.9.1 Hash Function

Characteristics of the SHA-256 Algorithm

FigNo:2.9.2 Characteristics of SHA-256

Some of the standout features of the SHA algorithm are as follows:

Message Length: The length of the cleartext should be less than 264 bits. The size needs to be in the comparison area to keep the digest as random as possible.

Digest Length: The length of the hash digest should be 256 bits in SHA 256 algorithm, 512 bits in SHA-512, and so on. Bigger digests usually suggest significantly more calculations at the cost of speed and space.

Irreversible: By design, all hash functions such as the SHA 256 are irreversible. You should neither get a plaintext when you have the digest beforehand nor should the digest provide its original value when you pass it through the hash function again.

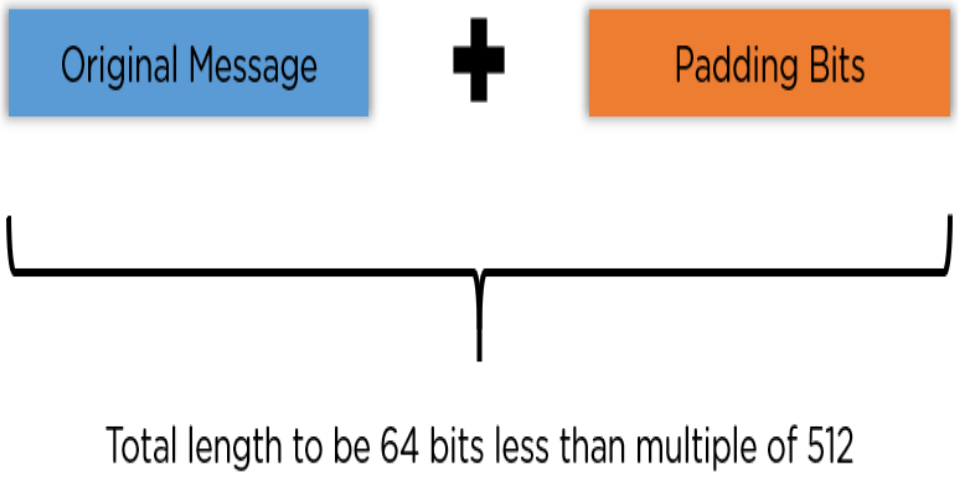
Now that you got a fair idea about the technical requirements for SHA, you can get into its complete procedure, in the next section.

Steps in SHA-256 Algorithm

You can divide the complete process into five different segments, as mentioned below:

Padding Bits

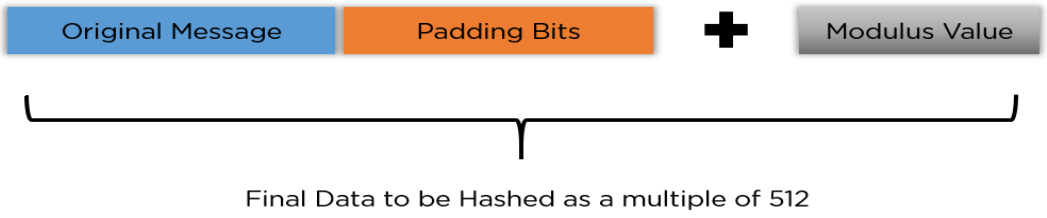
It adds some extra bits to the message, such that the length is exactly 64 bits short of a multiple of 512. During the addition, the first bit should be one, and the rest of it should be filled with zeroes.



FigNo:2.9.3 Bits Padding

Padding length

You can add 64 bits of data now to make the final plaintext a multiple of 512. You can calculate these 64 bits of characters by applying the modulus to your original cleartext without the padding.



FigNo:2.9.4 Padding Length

Initializing the Buffers:

You need to initialize the default values for eight buffers to be used in the rounds as follows:

```
a = 0x6a09e667
b = 0xbb67ae85
c = 0x3c6ef372
d = 0xa54ff53a
e = 0x510e527f
f = 0x9b05688c
g = 0x1f83d9ab
h = 0x5be0cd19
```

You also need to store 64 different keys in an array, ranging from K[0] to K[63]. They are initialized as follows:

```
k[0..63] :=
   0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
   0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
   0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
   0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
   0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
   0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
   0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
   0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2
```

FigNo:2.9.5 Keys

SHA256 Encrypt/Decrypt

SHA256 Encrypt/Decrypt is for generating SHA256 hashes from strings and decrypting SHA256 hashes to strings. In other words, this tool is a combination of SHA256 hash generator and SHA256 decrypter**.** SHA256 is a hashing function that creates a unique 256-bit hash with 64 characters long for every string. SHA256 or (SHA-256) stands for "Secure Hash Algorithm 256-bit" and it is found by National Security Agency (NSA) in the USA. SHA256 is one of the most popular hashing/encrypting function, especially after reveal of MD5 vulnerabilities. It offers a more secure solution and stronger for collusion attacks.

Decrypting SHA256 can seem like a daunting task, but with the right tools and knowledge, it can actually be quite simple.

First, it's important to understand that SHA256 is a cryptographic hash function, meaning that it is a mathematical algorithm that takes a string of any length and produces a fixed-length output. This output is known as a "hash" and is typically represented as a hexadecimal string.

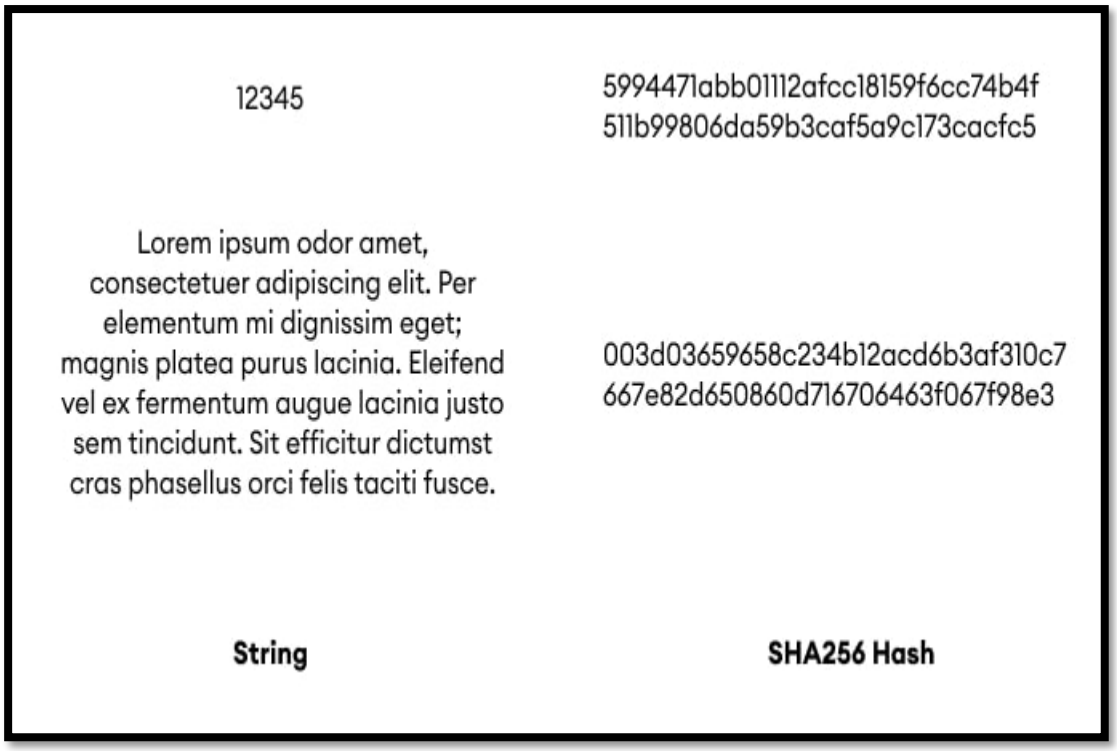
One of the key features of a cryptographic hash function is that it is one-way, meaning that it is virtually impossible to reverse the process and recover the original input from the hash. This makes it a popular choice for storing passwords and other sensitive information, as it is nearly impossible for an attacker to retrieve the original password from the hash.

As all hashing functions, SHA256 function has a one-way execution model, and it is irreversible. Decrypting SHA256 is not possible directly by using a simple function. There are several approaches to decrypt SHA256. If the encrypted text is long, it is very hard and time-consuming operation to decrypt/crack SHA256 hashes, even it is impossible if it is long enough. But, in general, people use SHA256 to decrypt passwords and emails which are mostly ~6-12 characters long. If you have a password or email that is hashed with SHA256, you may decrypt it by using these methods.

Use a predefined list or database: This method is also called a "dictionary attack". If you want to decrypt a password and the one you try to decrypt is a common one, it can be decrypted by using a common password list. There are a lot of lists on internet and one of these lists can be used to detect if SHA256 hash you want to crack is equal to SHA256 hash of a password from this list.

Iterate all possible combinations: This method is called "brute force" which relies on trial and error. This is a more time-consuming process but if you have data about possible characters and maximum length of the text that you want to decrypt, you can narrow down the combinations and crack SHA256 by iterating all possible combinations.

Here is a representation of how **SHA256 encoder decoder** works; there are two different strings with different character lengths, both produces unique SHA256 hashes with 64 characters long.



FigNo:2.9.6 SHA256 Encryption

If you use common password list for decryption of your SHA256 hash, it doesn't take much time. But be careful, if you use character sets and combinations, it may take minutes to hours to decrypt a hash and it uses sources of your computer significantly for computation/iteration.

The history of SHA256 dates back to the early 1990s, when the National Institute of Standards and Technology (NIST) began working on a new cryptographic hash function standard. This new standard was part of a larger effort to strengthen the security of computer systems and networks, and was designed to replace the older SHA-1 standard, which had been shown to be vulnerable to attack.

After several years of development, NIST published the new SHA-2 standard in 2001, which included four different hash functions: SHA-224, SHA-256, SHA-384, and SHA-512. These four functions were designed to be more secure and efficient than

the older SHA-1 standard, and were intended for use in a wide range of applications, including digital signatures, data integrity checks, and password storage.

SHA-256 is a cryptographic hash function that is commonly used in the blockchain and other security-critical applications. It is used to generate a unique, fixed-size string of text (called a "hash") from a larger input, such as a file or a block of data. This hash can then be used to verify the integrity of the original input, since any change to the input will produce a different hash. SHA-256 is considered to be very secure and is one of the most widely-used hash functions in the world. It is a part of the SHA-2 family of hash functions, which also includes SHA-224, SHA-384, and SHA-512.

FigNo:2.9.7 SHA256 Decryption with Successful Result after 1.2 Billion Trial (Brute Force)

Overall, decrypting SHA256 can be a challenging task, but with the right tools and knowledge, it is possible to recover the original password or input from the hash. Whether you are trying to recover a forgotten password or are investigating a security breach, the ability to decrypt SHA256 can be a valuable skill to have.

Algorithm 1 ENPI Verification Algorithm

Input: a hashed password $hashP$;
a negative password np
Output: true or false

$m \leftarrow$ LENGTH$(hashP)$
for $i \leftarrow 1$ **to** m **with stepsize of** 1 **do**
  if NUMBEROFSP$(np_i) \neq i$ **then**
    return false
  end if
end for
for $i \leftarrow 1$ **to** m **with stepsize of** 1 **do**
  if NUMBEROFSP$(np_i) \neq 1$ **then**
    return false
  end if
  $k \leftarrow$ INDEXOFSP(np_i)
  $x[k] \leftarrow \neg$TOBIT$(np_i[k])$
  for $j \leftarrow i+1$ **to** m **with stepsize of** 1 **do**
    if $np_j[k] \neq$ TOSYMBOL$(x[k])$ **then**
      return false
    end if
    $np_j[k] \leftarrow$ '*'
  end for
end for
if $x = hashP$ **then**
  return true
else
  return false
end if

Algorithm 2 ENPII Verification Algorithm

Input: a hashed password $hashP$;
a negative password np
Output: true or false

1: $m \leftarrow$ LENGTH($hashP$)
2: **for** $i \leftarrow 1$ **to** $m+4$ **with stepsize of** 1 **do**
3: **if** NUMBEROFSP(np_i) $\neq 3$ **then**
4: **return** false
5: **end if**
6: **end for**
7: **if** NUMBEROFDS(np_{m-1}, np_m) $\neq 1$ **or**
NUMBEROFDS(np_{m+1}, np_{m+2}) $\neq 1$ **or**
NUMBEROFDS(np_{m+3}, np_{m+4}) $\neq 1$
then
8: **return** false
9: **else**
10: $np_{m-1} \leftarrow$ MERGE(np_{m-1}, np_m)
11: $np_{m+1} \leftarrow$ MERGE(np_{m+1}, np_{m+2})
12: $np_{m+3} \leftarrow$ MERGE(np_{m+3}, np_{m+4})
13: **end if**
14: **if** NUMBEROFDS(np_{m+1}, np_{m+3}) $\neq 1$ **then**
15: **return** false
16: **else**
17: $np_m \leftarrow$ MERGE(np_{m+1}, np_{m+3})
18: **end if**
19: **for** $i \leftarrow m$ **to** 1 **with stepsize of** -1 **do**
20: **if** NUMBEROFSP(np_i) $\neq 1$ **then**
21: **return** false
22: **end if**
23: $k \leftarrow$ INDEXOFSP(np_i)
24: $x[k] \leftarrow \neg$TOBIT($np_i[k]$)
25: **for** $j \leftarrow i-1$ **to** 1 **with stepsize of** -1 **do**
26: **if** $np_j[k] \neq$ TOSYMBOL($x[k]$) **or** '*' **then**
27: **return** false
28: **end if**
29: $np_j[k] \leftarrow$ '*'
30: **end for**
31: **end for**
32: **if** $x = hashP$ **then**
33: **return** true
34: **else**
35: **return** false
36: **end if**

2.10 TECHNOLOGIES USED

Cloud computing

FigNo:2.10.1 Cloud Computing

Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS).

Who is using cloud computing

Organizations of every type, size, and industry are using the cloud for a wide variety of use cases, such as data backup, disaster recovery, email, virtual desktops, software development and testing, big data analytics, and customer-facing web applications. For example, healthcare companies are using the cloud to develop more personalized treatments for patients. Financial services companies are using the cloud to power real-time fraud detection and prevention. And video game makers are using the cloud

to deliver online games to millions of players around the world.

Benefits of cloud computing

Agility

The cloud gives you easy access to a broad range of technologies so that you can innovate faster and build nearly anything that you can imagine. You can quickly spin up resources as you need them–from infrastructure services, such as compute, storage, and databases, to Internet of Things, machine learning, data lakes and analytics, and much more.

You can deploy technology services in a matter of minutes, and get from idea to implementation several orders of magnitude faster than before. This gives you the freedom to experiment, test new ideas to differentiate customer experiences, and transform your business.

Elasticity

With cloud computing, you don't have to over-provision resources up front to handle peak levels of business activity in the future. Instead, you provision the amount of resources that you actually need. You can scale these resources up or down to instantly grow and shrink capacity as your business needs change.

Cost savings

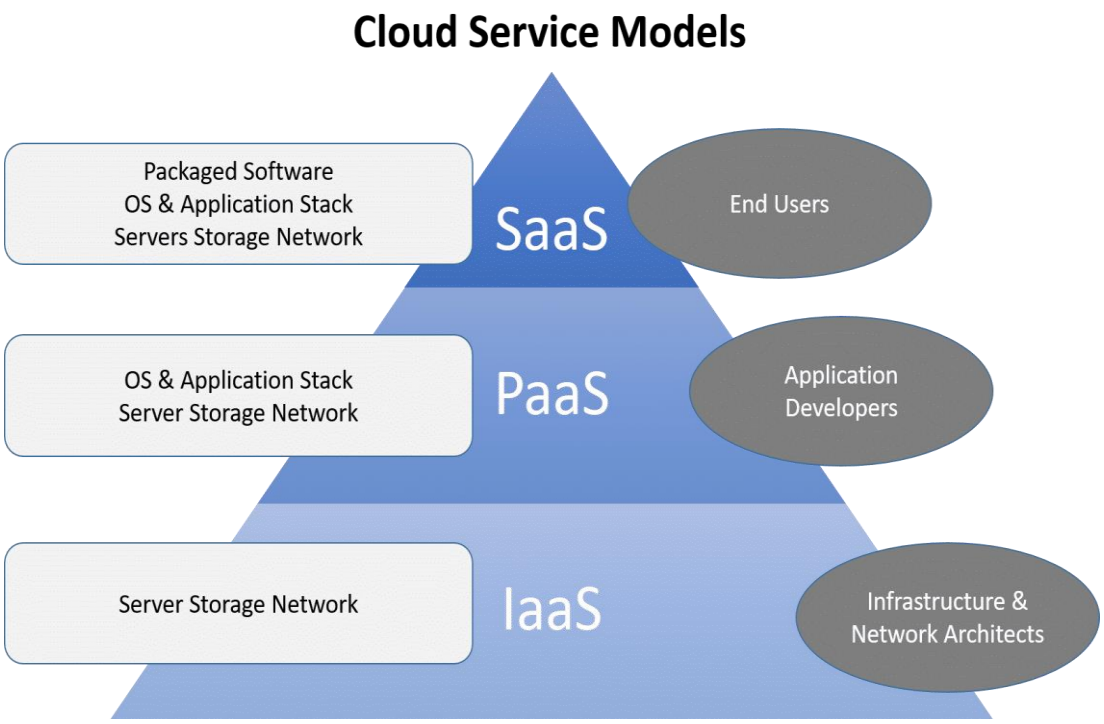
The cloud allows you to trade fixed expenses (such as data centers and physical servers) for variable expenses, and only pay for IT as you consume it. Plus, the variable expenses are much lower than what you would pay to do it yourself because of the economies of scale.

Deploy globally in minutes

With the cloud, you can expand to new geographic regions and deploy globally in minutes. For example, AWS has infrastructure all over the world, so you can deploy your application in multiple physical locations with just a few clicks. Putting applications in closer proximity to end users reduces latency and improves their experience.

Types of cloud computing

The three main types of cloud computing include Infrastructure as a Service, Platform as a Service, and Software as a Service. Each type of cloud computing provides different levels of control, flexibility, and management so that you can select the right set of services for your needs.



FigNo:2.10.2 Cloud Services

Infrastructure as a Service (IaaS)

IaaS contains the basic building blocks for cloud IT. It typically provides access to networking features, computers (virtual or on dedicated hardware), and data storage space. IaaS gives you the highest level of flexibility and management control over your IT resources. It is most similar to the existing IT resources with which many IT departments and developers are familiar.

Platform as a Service (PaaS)

PaaS removes the need for you to manage underlying infrastructure (usually hardware and operating systems), and allows you to focus on the deployment and management

of your applications. This helps you be more efficient as you don’t need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.

Software as a Service (SaaS)

SaaS provides you with a complete product that is run and managed by the service provider. In most cases, people referring to SaaS are referring to end-user applications (such as web-based email). With a SaaS offering, you don’t have to think about how the service is maintained or how the underlying infrastructure is managed. You only need to think about how you will use that particular software.

About JAVA

Certainly! Java is a widely used, versatile, and object-oriented programming language that was developed by James Gosling and his team at Sun Microsystems (later acquired by Oracle Corporation). It was first released in 1995 and has since become one of the most popular and enduring programming languages in the software development industry. Here are some key aspects of Java:

Platform Independence: Java is known for its "write once, run anywhere" capability. This is achieved through the use of the Java Virtual Machine (JVM), which allows Java programs to be executed on various platforms without modification, as long as a compatible JVM is available.

Object-Oriented: Java is designed around the concept of object-oriented programming (OOP). It emphasizes the use of classes and objects to structure code, promoting modularity, reusability, and maintainability.

Syntax and Structure: Java's syntax is similar to that of C and C++, making it relatively easy for programmers familiar with these languages to learn Java. The language enforces strict syntax rules, which helps catch errors during compilation rather than runtime.

Standard Library: Java provides a comprehensive and robust standard library that includes a wide range of classes and methods for common programming tasks, such

as working with data structures, networking, input/output, and more.

Memory Management: Java features automatic memory management through a process called garbage collection. This helps developers avoid memory leaks and reduces the likelihood of segmentation faults and other memory-related errors.

Multi-threading: Java supports multi-threading, allowing developers to create programs that can perform multiple tasks concurrently. This is useful for applications that need to handle multiple operations simultaneously, such as graphical user interfaces and network servers.

Security: Java was designed with security in mind. It includes features such as the sandboxing of applets (small programs designed to run within a web browser) and various security mechanisms to prevent unauthorized access and malicious code execution.

Community and Ecosystem: Java has a large and active developer community, which has contributed to the creation of numerous open-source libraries, frameworks, and tools. Examples include the Spring Framework for enterprise application development and the Apache Maven build automation tool.

Constructors and Destructors:

Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words, destructors are used to release the resources allocated to the object. In C#.NET the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize procedure can be called only from the class it belongs to or from derived classes.

Garbage Collection

Garbage Collection is another new feature in C#.NET. The .NET Framework monitors allocated resources, such as objects and variables. In addition, the .NET Framework automatically releases memory for reuse by destroying objects that are no longer in use.

In C#.NET, the garbage collector checks for the objects that are not currently in use by applications. When the garbage collector comes across an object that is marked for garbage collection, it releases the memory occupied by the object.

Overloading

Overloading is another feature in C#. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of arguments. Besides using overloading for procedures, we can use it for constructors and properties in a class.

Multithreading:

C#.NET also supports multithreading. An application that supports multithreading can handle multiple tasks simultaneously, we can use multithreading to decrease the time taken by an application to respond to user interaction.

Structured Exception Handling

C#.NET supports structured handling, which enables us to detect and remove errors at runtime. In C#.NET, we need to use Try…Catch…Finally statements to create exception handlers. Using Try…Catch…Finally statements, we can create robust and effective exception handlers to improve the performance of our application.

THE .NET FRAMEWORK

The .NET Framework has two main parts:

1. The Common Language Runtime (CLR).

2. A hierarchical set of class libraries.

The CLR is described as the “execution engine” of .NET. It provides the environment within which programs run.

The most important features are:

Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.

Memory management, notably including garbage collection.

Checking and enforcing security restrictions on the running code.

Loading and executing programs, with version control and other such features.

The following features of the .NET framework are also worth description:

Managed Code

The code that targets .NET, and which contains certain extra Information – "metadata" - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.

Managed Data

With Managed Code comes Managed Data. CLR provides memory allocation and Deal location facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you're using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn't get garbage collected but instead is looked after by unmanaged code.

Common Type System

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn't attempt to access memory that hasn't been allocated to it.

Common Language Specification

The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

The Class Library

.NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System. Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of converting value types to object types if and when necessary.

The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O, threading, and so on, as well as XML and database connectivity.

The class library is subdivided into a number of sets (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

LANGUAGES SUPPORTED BY .NET

The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft's old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading.

Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in Visual Basic .NET.

Managed Extensions for C++ and attributed programming are just some of the enhancements made to the C++ language. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework.

C# is Microsoft's new language. It's a C-style language that is essentially "C++ for Rapid Application Development". Unlike other languages, its specification is just the grammar of the language. It has no standard library of its own, and instead has been designed with the intention of using the .NET libraries as its own.

Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages.

Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State's Perl Dev Kit.

Other languages for which .NET compilers are available include

FORTRAN

COBOL

Eiffel

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet.

OBJECTIVES OF. NET FRAMEWORK

1. To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely.

2. To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.

3. Eliminates the performance problems.

There are different types of application, such as Windows-based applications and Web-based applications.

FEATURES OF SQL SERVER

The OLAP Services feature available in SQL Server version 7.0 is now called SQL Server 2000 Analysis Services. The term OLAP Services has been replaced with the term Analysis Services. Analysis Services also includes a new data mining component. The Repository component available in SQL Server version 7.0 is now called Microsoft SQL Server 2000 Meta Data Services. References to the component now use the term Meta Data Services. The term repository is used only in reference to the repository engine within Meta Data Services

SQL-SERVER database consists of six type of objects,

They are,

1. TABLE
2. QUERY
3. FORM
4. REPORT
5. MACRO

TABLE:

A database is a collection of data about a specific topic.

VIEWS OF TABLE:

We can work with a table in two types,

1. Design View
2. Datasheet View

DESIGN VIEW

To build or modify the structure of a table we work in the table design view. We can specify what kind of data will be hold.

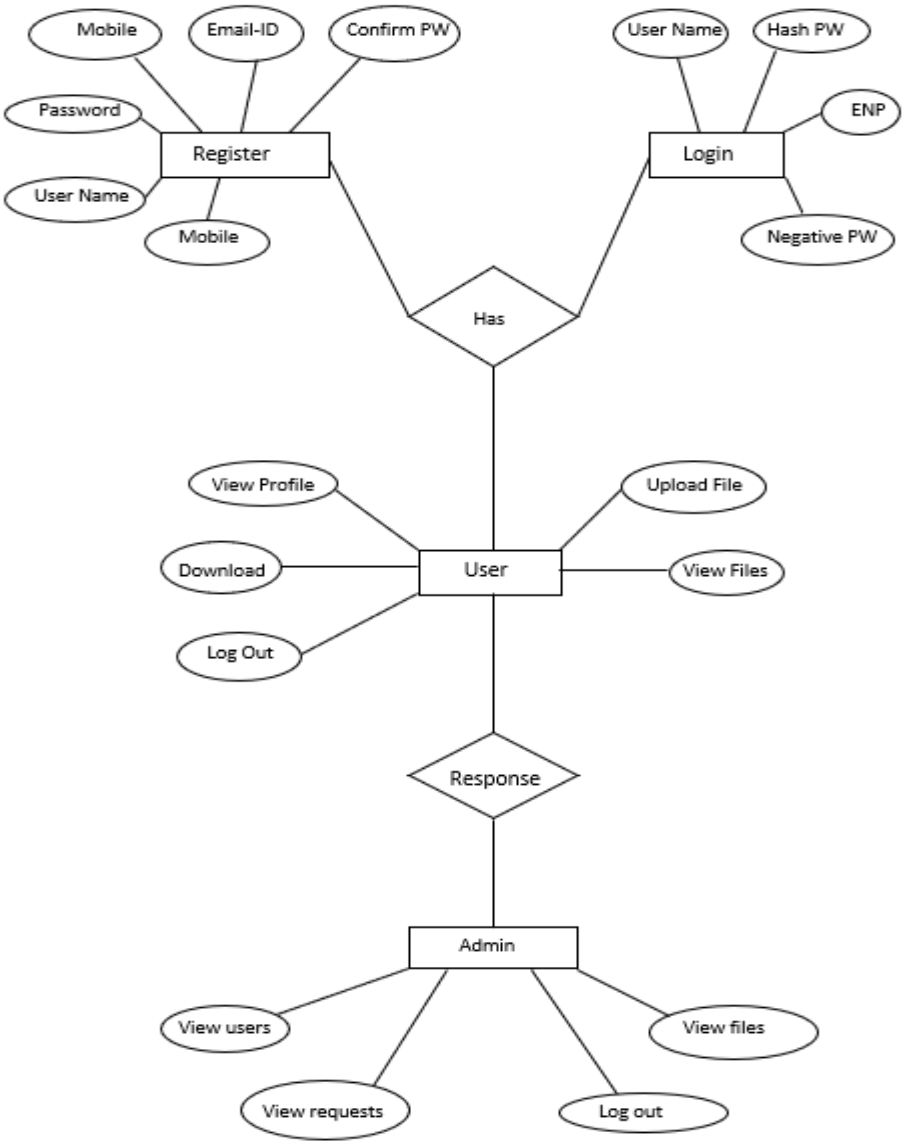
QUERY:

A query is a question that has to be asked the data. Access gathers data that answers the question from one or more table. The data that make up the answer is either dynaset (if you edit it) or a snapshot (it cannot be edited).Each time we run query, we get latest information in the dynaset. Access either displays the dynaset or snapshot for us to view or perform an action on it, such as deleting or updating.

3. SYSTEM DESIGN

3.1 DATABASE DESIGN(E-R DIAGRAM)

ER-Diagram of Authentication by Encrypted Negative Password



FigNo:3.1.1 ER-Diagram

3.2 NORMALIZATION

The method used to organize the data set's information is called normalization.

Normalization is used to prevent an association or collection of interactions from becoming overtly repetitious. Additionally, undesirable qualities like Insertion, Update, and Deletion Anomalies are disposed of using it.

The larger table is split into smaller ones by normalization, which then connects them.

The conventional framework is applied to lessen obvious repetition in the data base table.

Why do we need Normalization

Eliminating these inconsistencies is the main argument for normalizing the relations. As the data collection grows, the inability to ignore idiosyncrasies leads to overt repetition of information, information uprightness, and other problems. Standardization consists of a set of guidelines that help you create an effective information base design.

Three categories of aberrant information change can be made:

Insertion Anomaly: When a new tuple cannot be embedded into a relationship due to a lack of information, this is referred to as an insertion anomaly.

Deletion Anomaly: The term "erase abnormality" refers to a situation in which crucial information is unintentionally lost when data is cancelled.

Update Anomaly: The update irregularity is the point at which a single information change necessitates the updating of many information columns.

Advantages of Normalization

- Normalization helps to reduce overt repetition of information.
- Greater in association with broad information.
- Data set internal consistency.
- A much more flexible data set strategy.
- Promotes the idea of social propriety.

Disadvantages of Normalization

- You need to know the client's requirements before you can start compiling the information base.
- The execution corrupts while restoring relationships to higher ordinarily arranged structures, such as 4NF and 5NF.
- Standardizing relationships on a more serious level is quite time-consuming and difficult.
- A careless breakdown could result in a poor information base plan, which would have serious consequences.

First Normal Form (1NF)

- If a connection contains a nuclear worth, it will be classified as 1NF.one valued item.
- It indicates that a table's property cannot include multiple qualities. It should only include
- The multi-esteemed trait, composite quality, and their mixtures are prohibited by the first ordinary structure.
- Model: Relation EMPLOYEE is not in the multi-valued attribute mindset. EMP PHONE.

Second Normal Form (2NF)

- All non-key credits in the ensuing typical structure are fully functional ward on the necessary key.

Third Normal Form (3NF)

- If the connection is in 2NF and does not have any transitive halfway dependency, it will be in 3NF.
- To reduce information duplication, 3NF is used. It is also used to achieve information integrity.
- The connection should be in the third common structure if there is no transitive reliance for non-prime credits.

- If at least one of the following conditions is true for each non-trivial capacity dependence X Y, a connection is in the third typical structure.
  - ✓ One super key is X.
  - ✓ Y is a prime property, meaning that each element of Y is required for a certain applicant key.

Boyce Codd normal form (BCNF)

- BCNF is the improved version of 3NF.Compared to 3NF, it is harsher.
- A table is in the BCNF if each practical reliance is X > Y, where X is the table's very key.
- The table should be in 3NF for BCNF, and LHS is crucial for each FD.

Fourth normal form (4NF)

- If a link has a standard Boyce Codd structure and no multi-esteemed dependency, it will be in 4NF.
- For dependence A B, the relationship will be a multi-valued reliance if, for a single value of A, different benefits of B exist.

Fifth normal form (5NF)

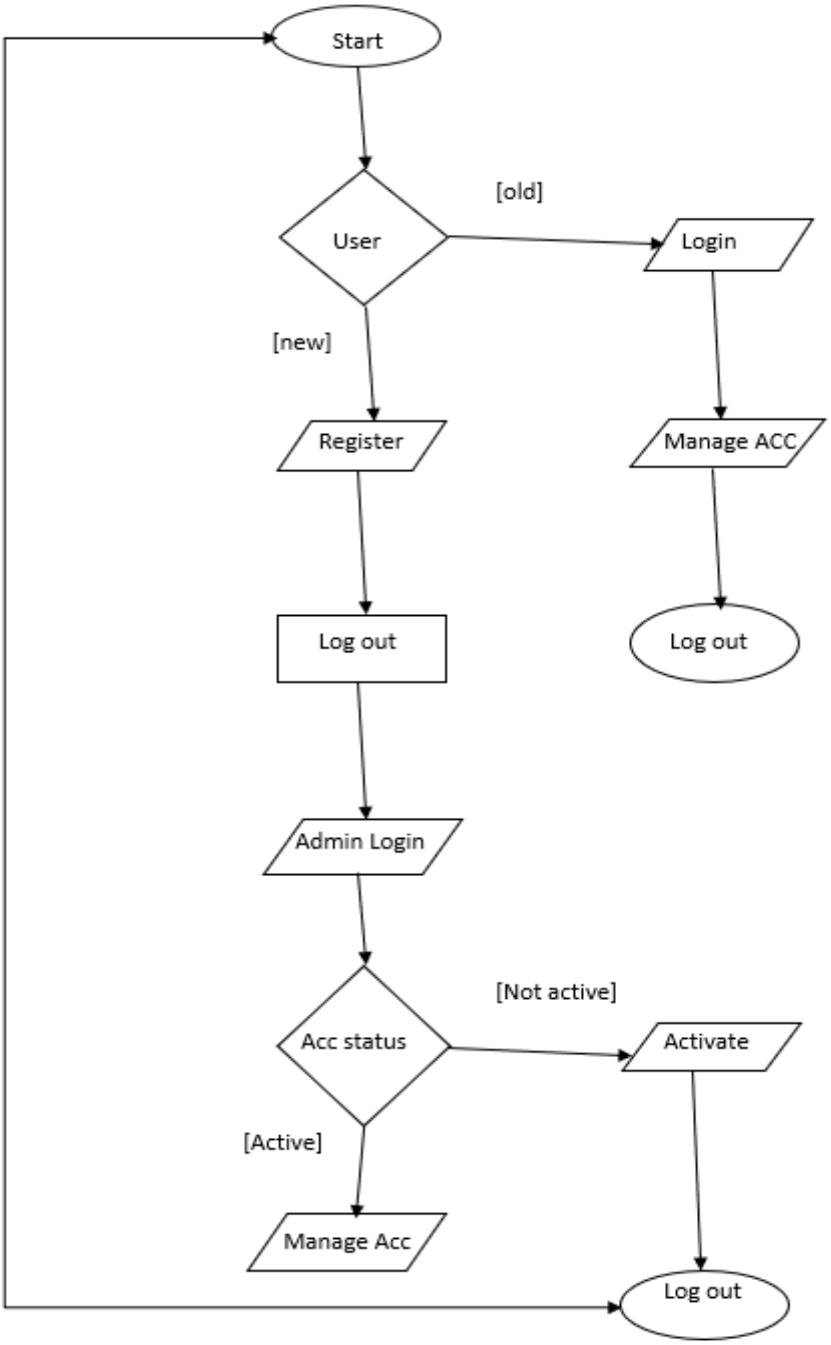
- A connection is considered to be in 5NF if it is in 4NF, is lossless and has no join reliance.
- 5NF is content when all of the tables are divided into as many separate number tables as necessary to avoid obvious repetition.
- Project-join ordinary structure (PJ/NF) is another name for 5NF.

3.3 DATA DICTIONARY

Field Name	Description	Data Type	Length	Constraints
User ID	Unique identifier for each user	Integer	50	Primary Key
Username	User's chosen username	Varchar	50	Unique
Encrypted NP	Encrypted Negative Password	Varchar	100	
Salt	Random data used in password hashing	Varchar	100	
Last Login	Timestamp of the user's last successful login	Datetime	20	
Failed Login	Number of consecutive failed login attempts	Integer	100	
Account Status	Status of the user account (active, locked)	Varchar	10	
Registration Date	Date when the user registered	Datetime	20	
Email Address	User's email address	Varchar	100	
Phone Number	User's Phone Number	Varchar	20	

Table-3.3 : Data Dictionary

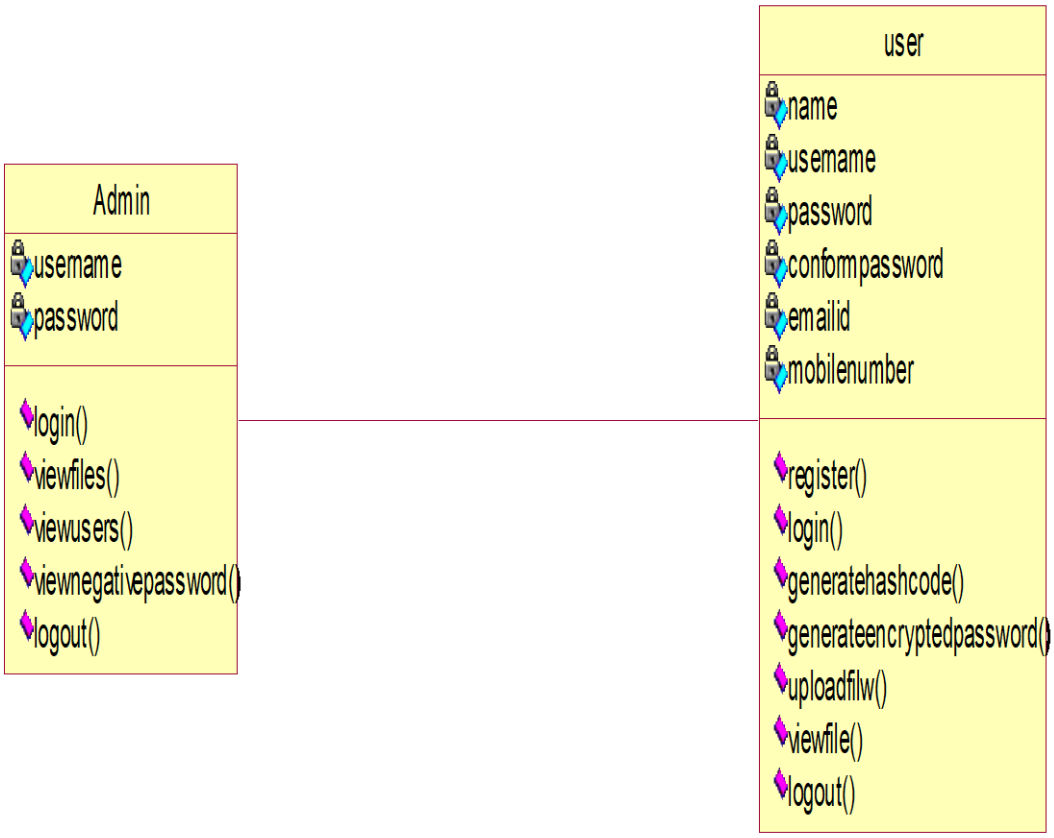
3.4 Data Flow Diagram



FigNo:3.4 Data Flow Diagram

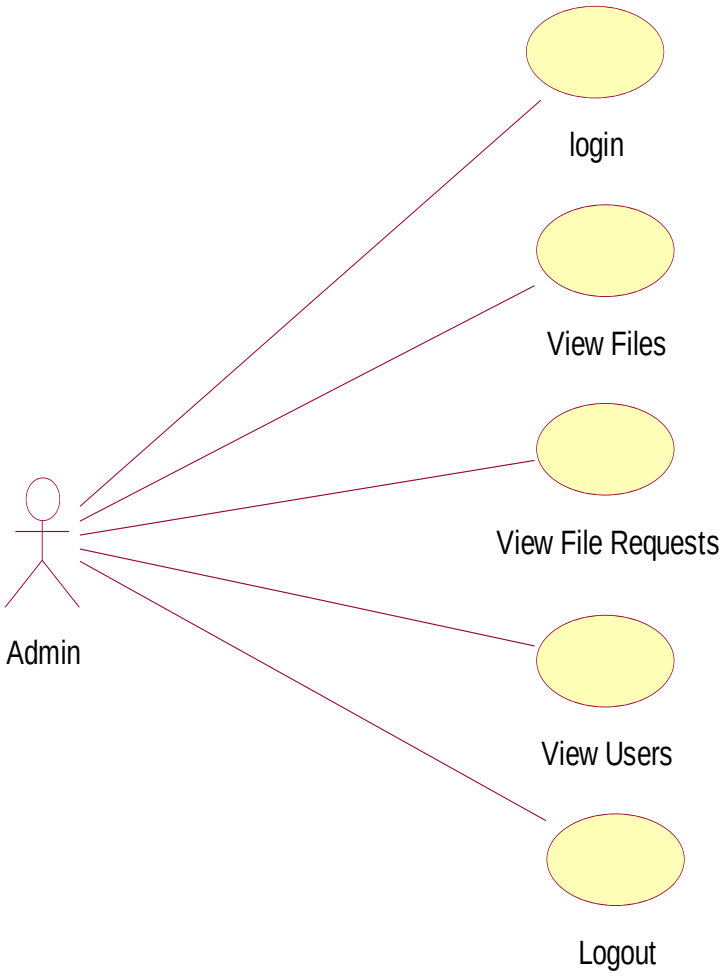
3.5 UMLDIAGRAMS

CLASS DIAGRAM



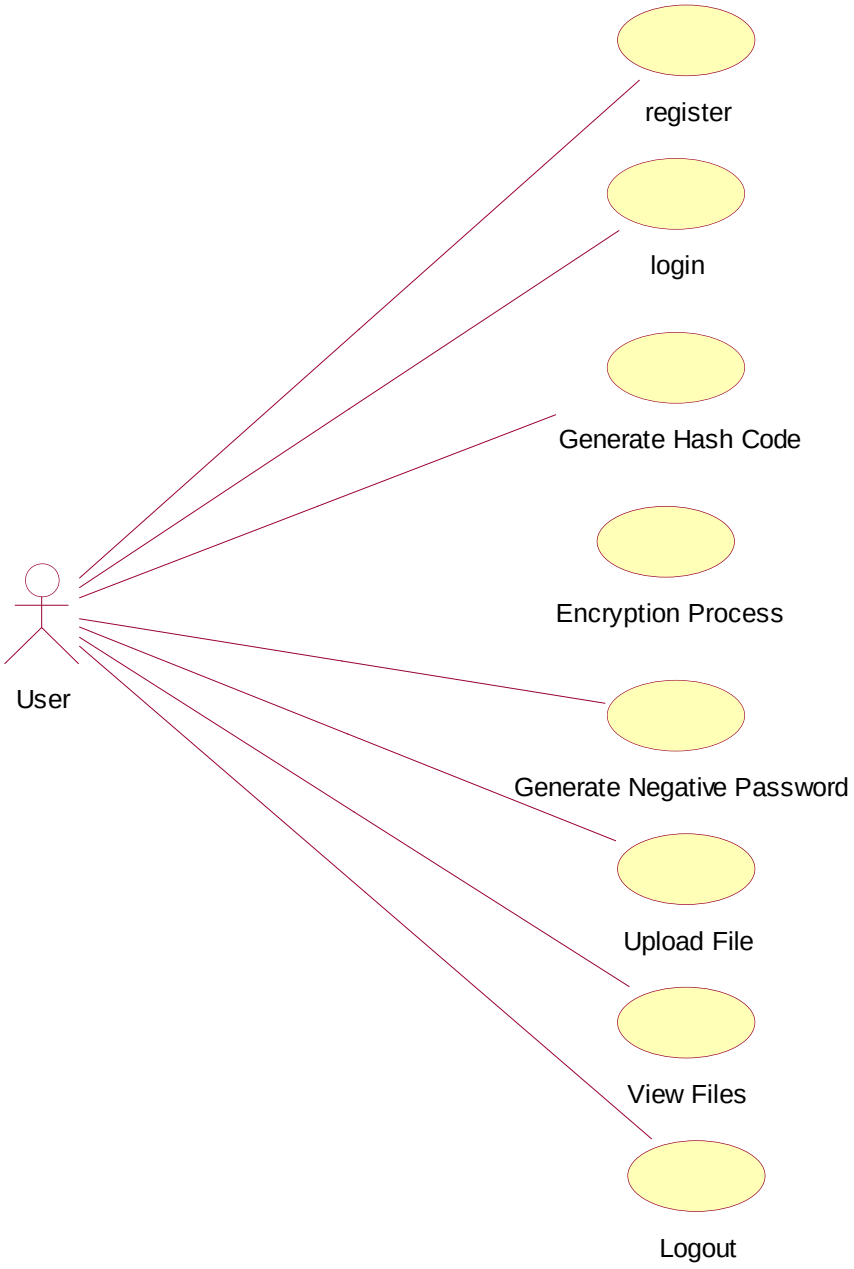
FigNo:3.5.1 Class Diagram

ADMIN USECASE



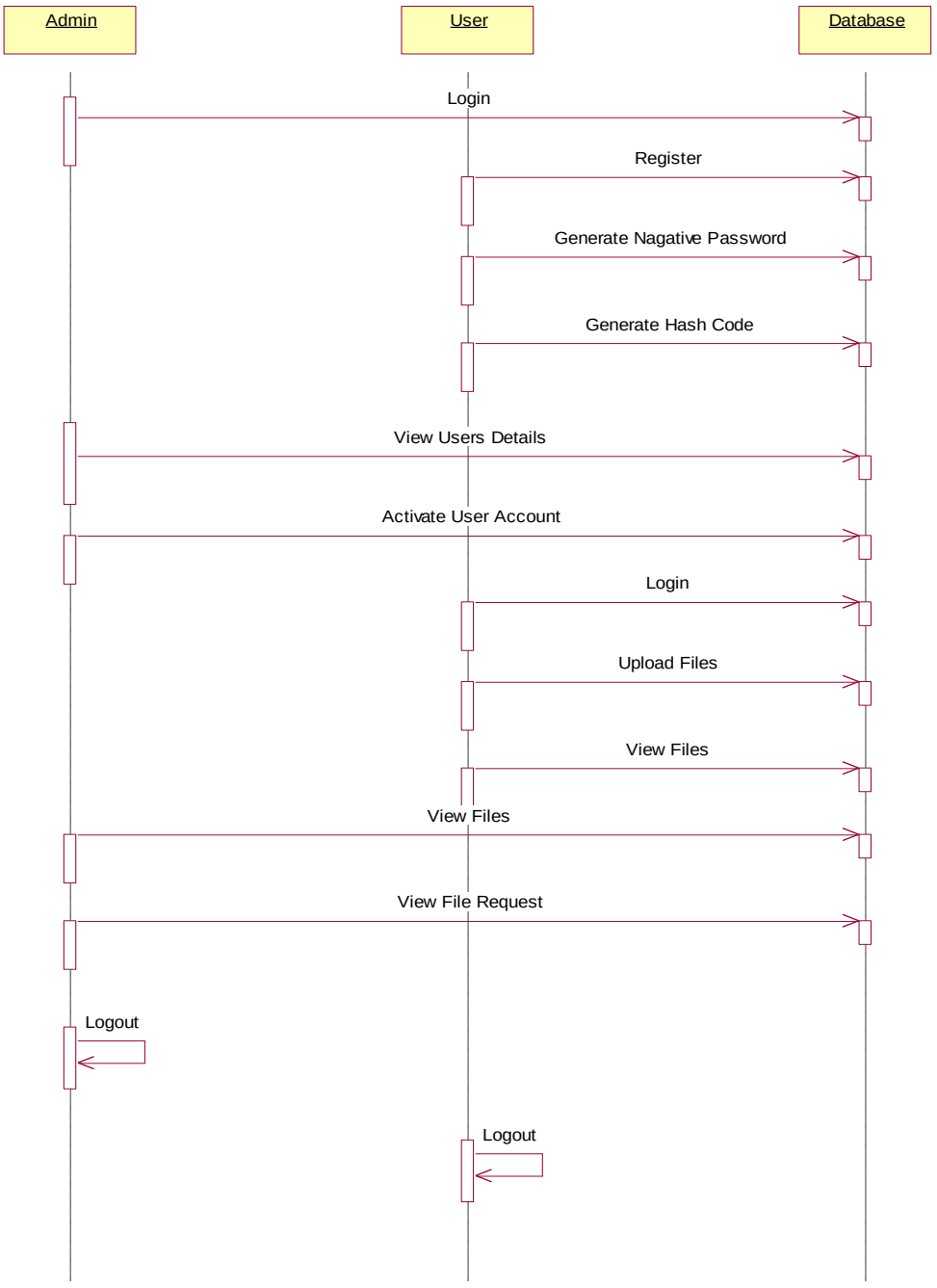
FigNo:3.5.2 Admin Use case diagram

USER USECASE



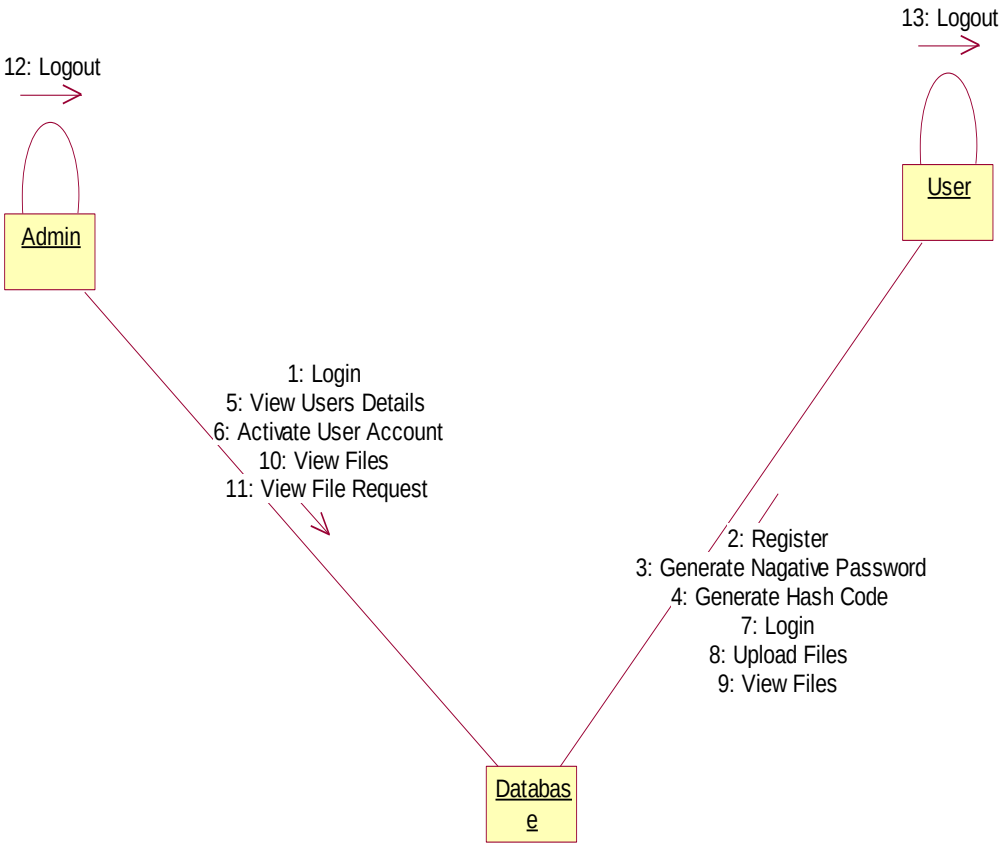
FigNo:3.5.3 User Use Case diagram

SEQUENCE DIAGRAM



FigNo:3.5.4 Sequence diagram

COLLABORATION DIAGRAM



FigNo:3.5.5 Collaboration diagram

4.TESTING

4.1 INTRODUCTION

Testing is a critical process in various fields, including software development, quality assurance, manufacturing, and more. It involves assessing the functionality, performance, and reliability of a product or system to ensure it meets specific requirements and standards. The primary goal of testing is to identify defects, errors, or shortcomings within the product and provide feedback to improve its overall quality.

In software development, testing helps identify and rectify issues that could lead to software crashes, security vulnerabilities, or user dissatisfaction. There are several types of testing, each serving a specific purpose.

4.2 TYPES OF TESTING

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.

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.

Functional test

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

Functional testing is centered on the following items:

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

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

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

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

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.

System Test

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

White Box Testing

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

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most

other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

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 field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

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

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

4.3 TESTING METHODOLOGY

Functional Testing:

- Test the basic functionality of the "Encrypted Negative Password" authentication process.
- Ensure that the negative password is properly handled during the authentication process.

Security Testing:

- Conduct security testing, including vulnerability assessments and penetration testing.
- Test for potential vulnerabilities like encryption weaknesses, password recovery mechanisms, and data leakage.

Negative Testing:

- Perform negative testing to validate the system's behavior when unexpected inputs or scenarios are encountered.
- Test cases might include entering incorrect encrypted passwords, manipulating encrypted data, or attempting unauthorized access.

Performance Testing:

- Assess the performance of the authentication system under various load conditions.
- Determine whether the encryption and decryption processes introduce any significant performance bottlenecks.

User Acceptance Testing (UAT):

- Involve actual users or representatives to participate in UAT.
- Gather feedback on the user experience, ease of use, and any issues encountered during the authentication process.

Regression Testing:

- Perform regression testing whenever changes are made to the authentication method or underlying systems to ensure no unintended issues arise.

4.4 TEST CASES

USER REQUIREMENTS:

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

Home

Use case ID	Authentication by Encrypted Negative Password
Use case Name	Home button
Description	Display home page of application
Primary actor	User
Precondition	User must open application
Post condition	Display the Home Page of an application
Frequency of Use case	Many times
Alternative use case	N/A
Use case Diagrams	N/A
Attachments	N/A

Table 4.4.1: Home

Registration Form

Use case ID	Authentication by Encrypted Negative Password
Use case Name	Registration
Description	It displays the credential form
Primary actor	User
Precondition	User Must have Email ID and Phone
Post condition	User get the popup Message" U successfully Registered"
Frequency of Use case	One time
Alternative use case	N/A
Use case Diagrams	N/A
Attachments	N/A

Table 4.4.2: Registration

Login Form

Use case ID	Authentication by Encrypted Negative Password
Use case Name	Login Form
Description	Display Login form to the User
Primary actor	User
Precondition	User must have username &password
Post condition	User send the response to the admin
Frequency of Use case	Many times
Alternative use case	N/A
Use case Diagrams	N/A
Attachments	N/A

Table 4.4.3: Login

Administrator

Use case ID	Authentication by Encrypted Negative Password
Use case Name	admin
Description	View details of all the user files
Primary actor	Admin
Precondition	Must open the application home page
Post condition	View files by login
Frequency of Use case	Many times
Alternative use case	N/A
Use case Diagrams	N/A
Attachments	N/A

Table 4.4.4: Administrator

User

Use case ID	Authentication by Encrypted Negative Password
Use case Name	User
Description	User uploads the files and send the response & key to admin
Primary actor	User
Precondition	User must be login
Post condition	View files
Frequency of Use case	Many times
Alternative use case	N/A
Use case Diagrams	One
Attachments	Files

Table 4.4.5: User

TEST CASES

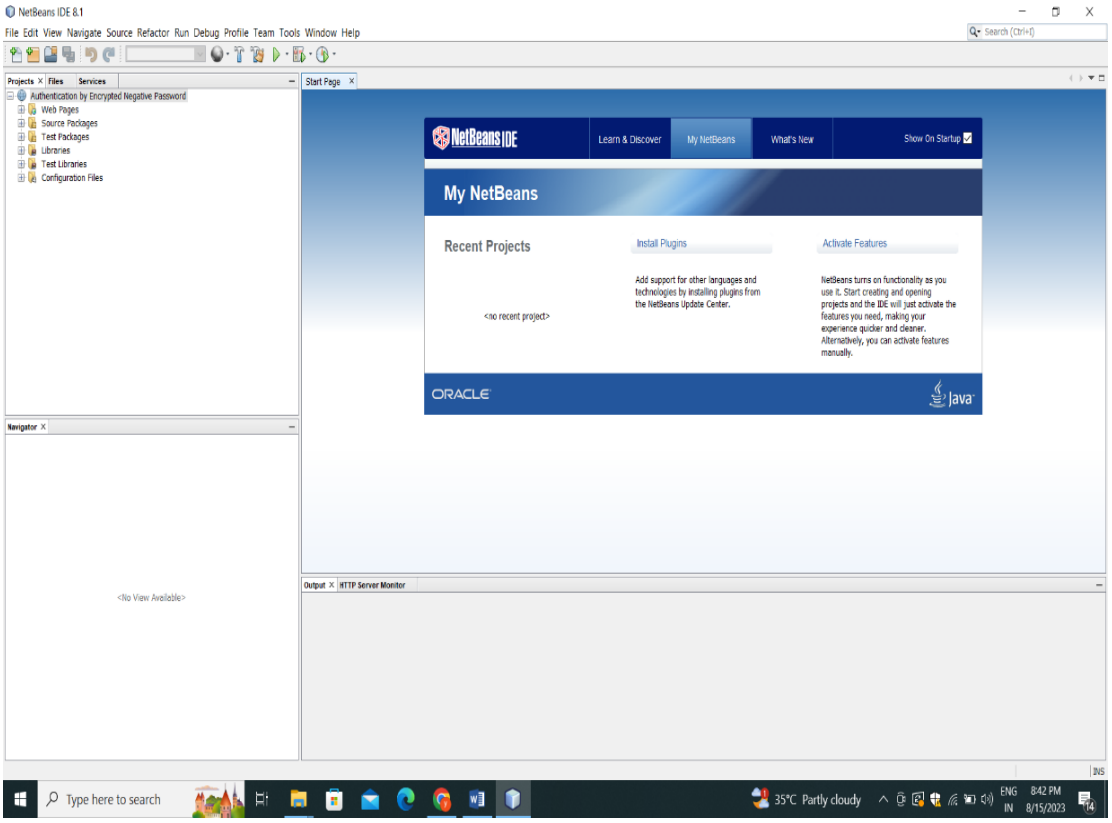
Test Case	Test Case Name	Description	Status
Case 1	User registration	User has to register by giving his details successfully	Pass
Case 2	ENP generation	User generates his three level security passwords	Pass
Case 3	User Login	User cannot login without being activated his account by admin	Pass
Case 4	Admin authorization	Admin cannot get access without response from user	Pass
Case 5	User authentication	User gets credentials by client for authentication	Pass

Table 4.4.6: Test Cases

5.IMPLEMENTATION

5.1 Sample Outputs

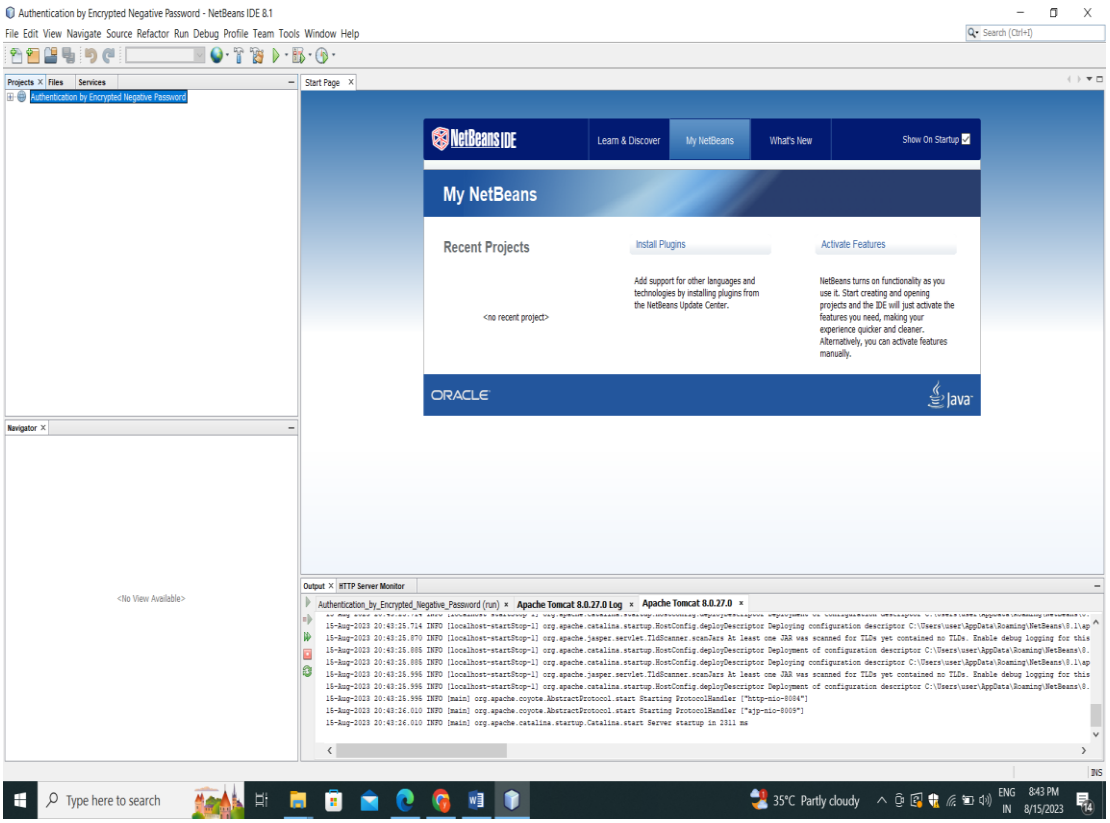
NETBEANS



Screen:5.1.1 NETBEANS

Description: Open & Run the net beans application.

RUN PROJECT FIE



Screen:5.1.2 RUN PROJECT FIE

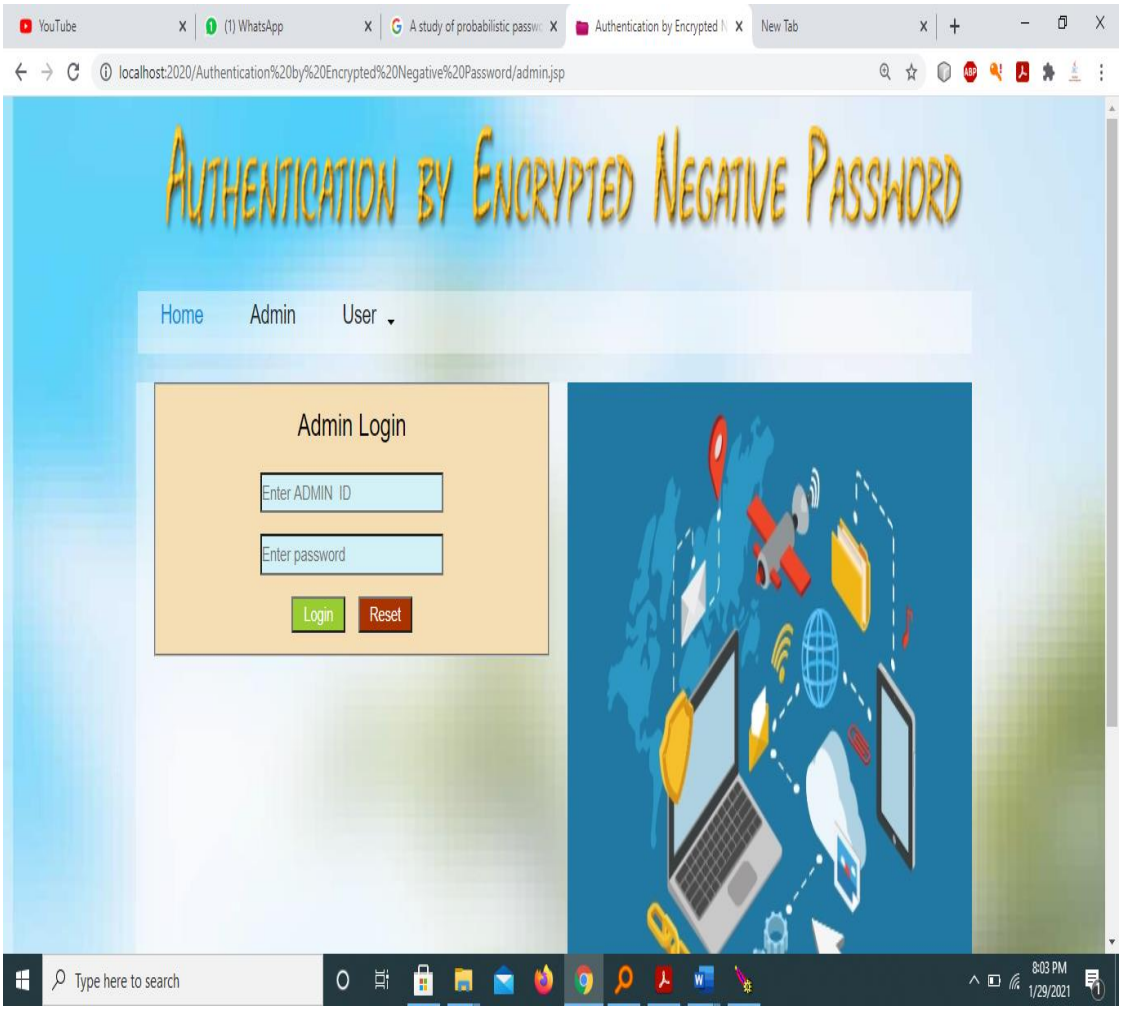
Description: open and run the project file.

HOME PAGE

Screen:5.1.3 HOME PAGE

Description: This is Home page which contains the modules like ‘admin’ & ‘user’.

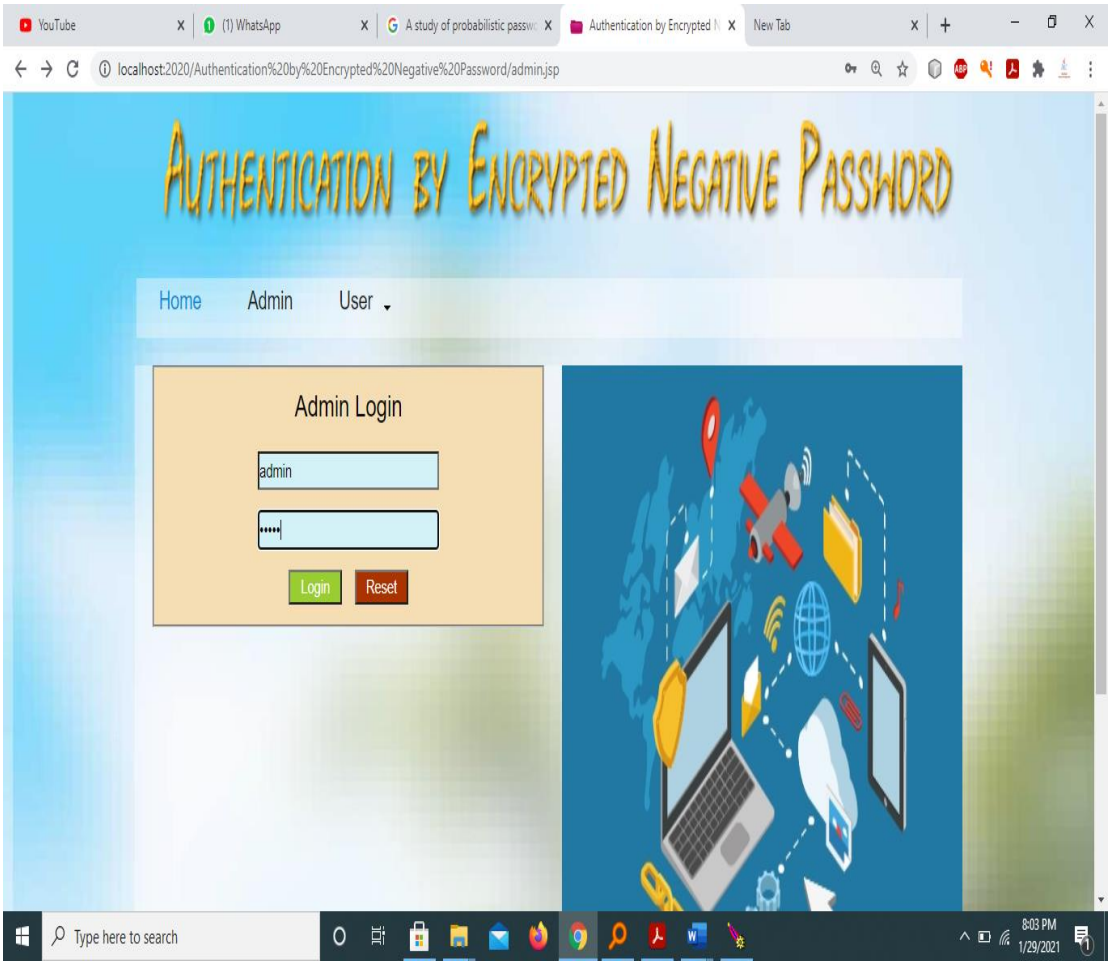
ADMIN LOGIN PAGE



Screen:5.1.4 ADMIN LOGIN PAGE

Description: Admin will login to the application using credentials.

ADMIN LOGIN



Screen:5.1.5 ADMIN LOGIN

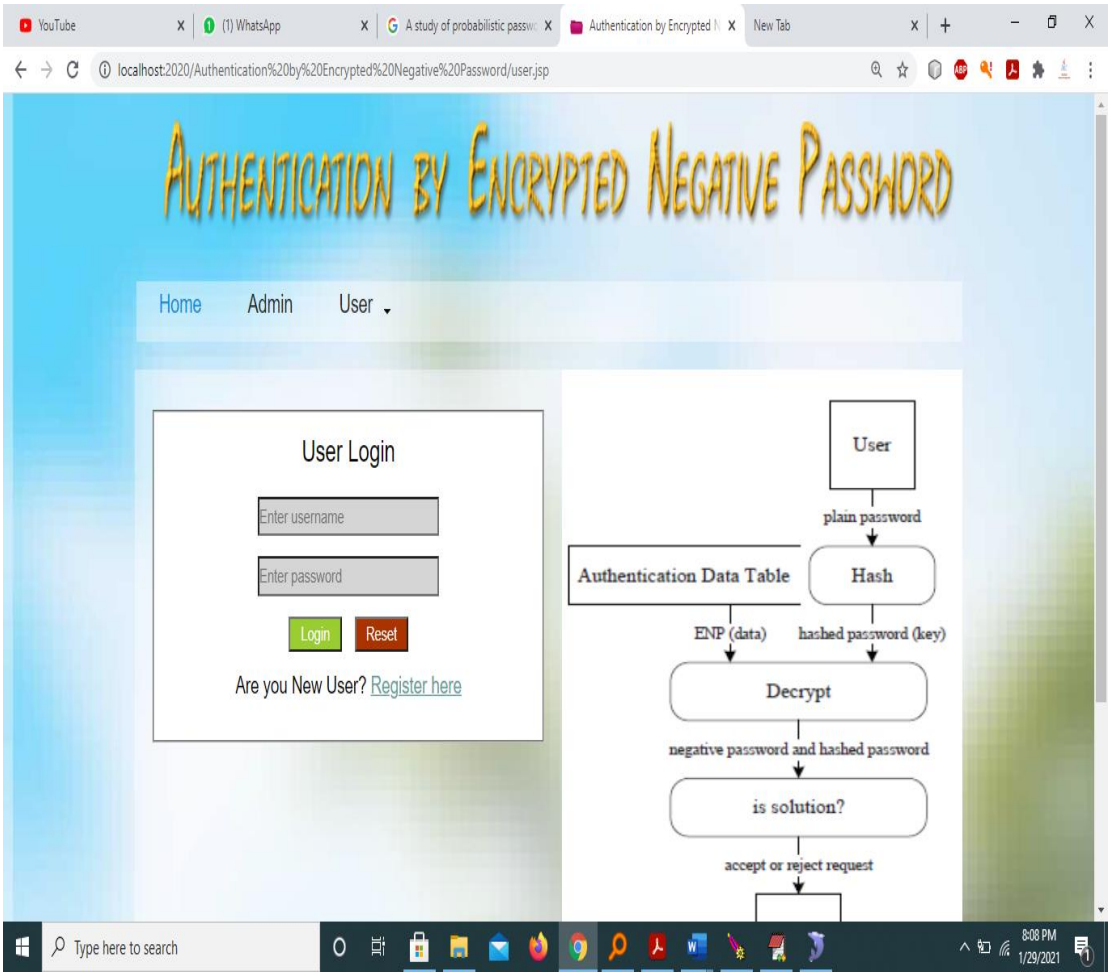
Description: Admin entered his credentials and will get access.

ADMIN HOME PAGE

Screen:5.1.6 ADMIN HOME PAGE

Description: These are the functions that admin contains in his home page.

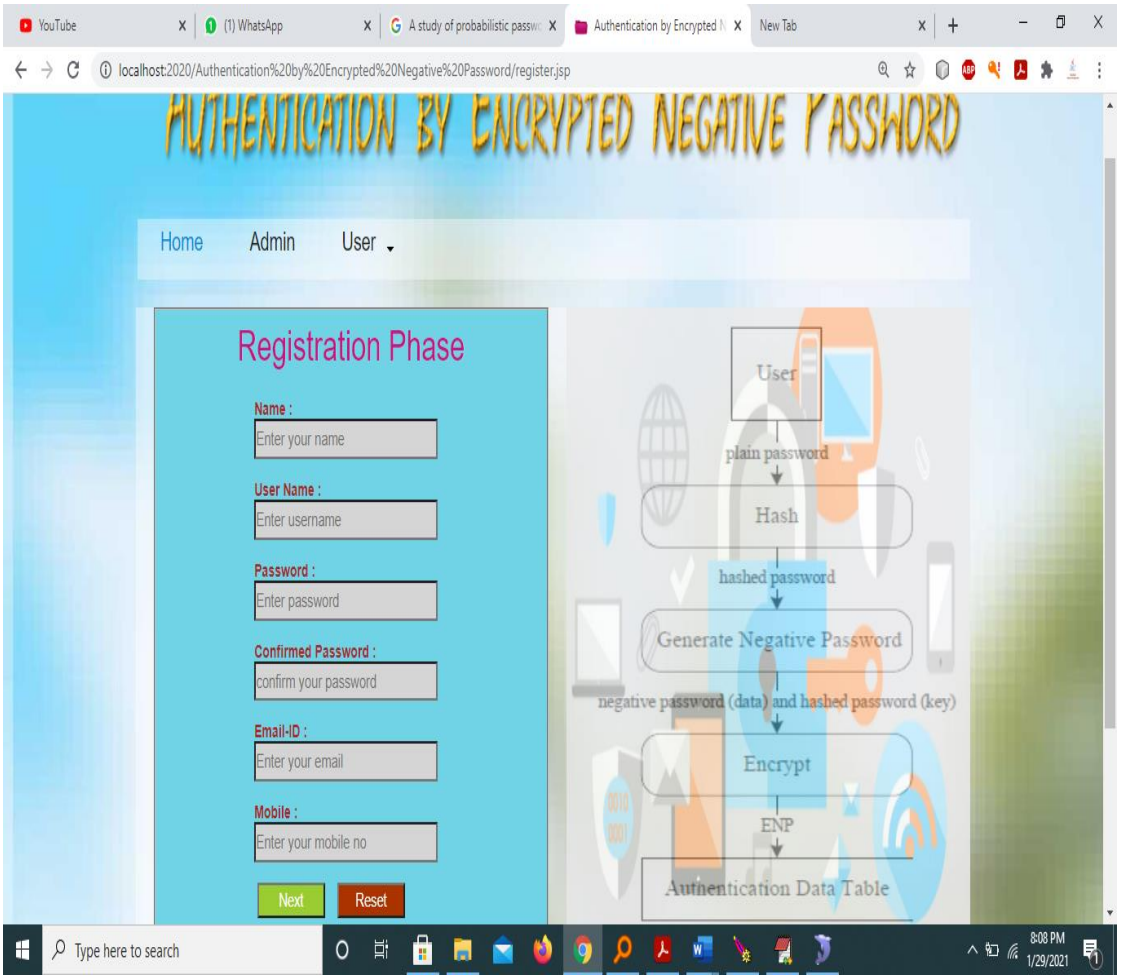
USER LOGIN



Screen:5.1.7 USER LOGIN

Description: This is user login frame user has to login with his user name & password.

REGISTRATION



Screen:5.1.8 REGISTRATION

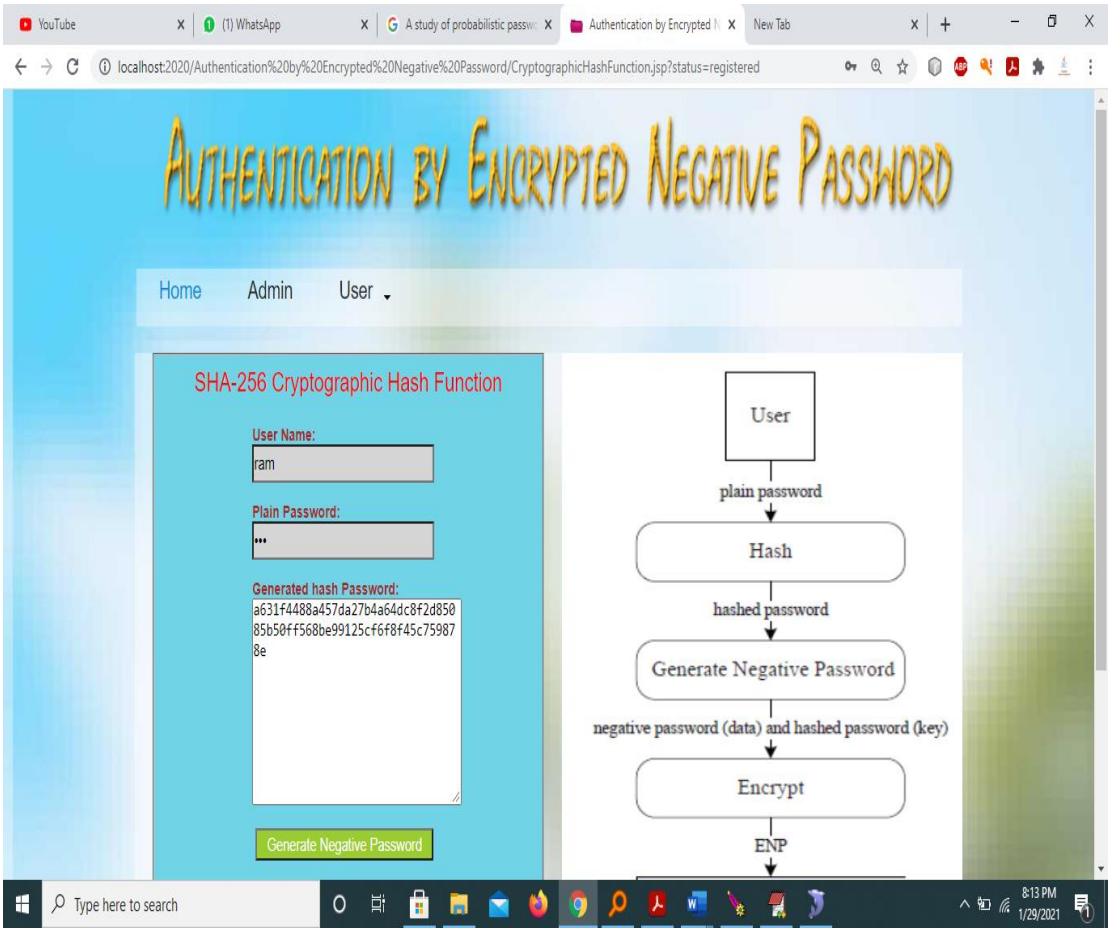
Description: After user clicks on register this page will open.

REGISTRATION

Screen:5.1.9 REGISTRATION

Description: User registers by giving his details like Name, Password, Contact num, Email-ID.

Generating Hash Password



Screen:5.1.10 Generating Hash Password

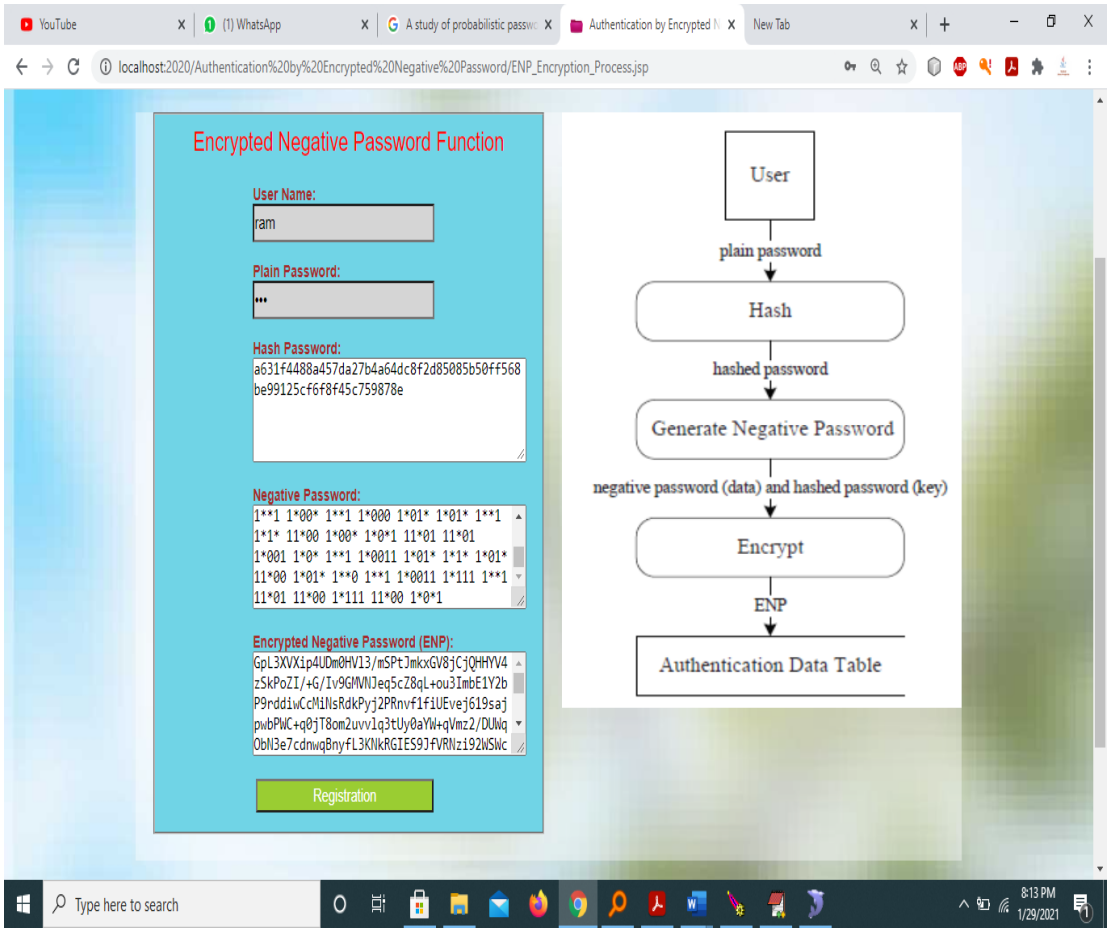
Description: User Generates the thee level security authentication like Hash password, Negative password, Encrypted Negative Password.
In this state we are generating hash password.

NEGATIVE PASSWORD GENERATION

Screen:5.1.11 NEGATIVE PASSWORD GENERATION

Description: Now the user generates negative password through hash password

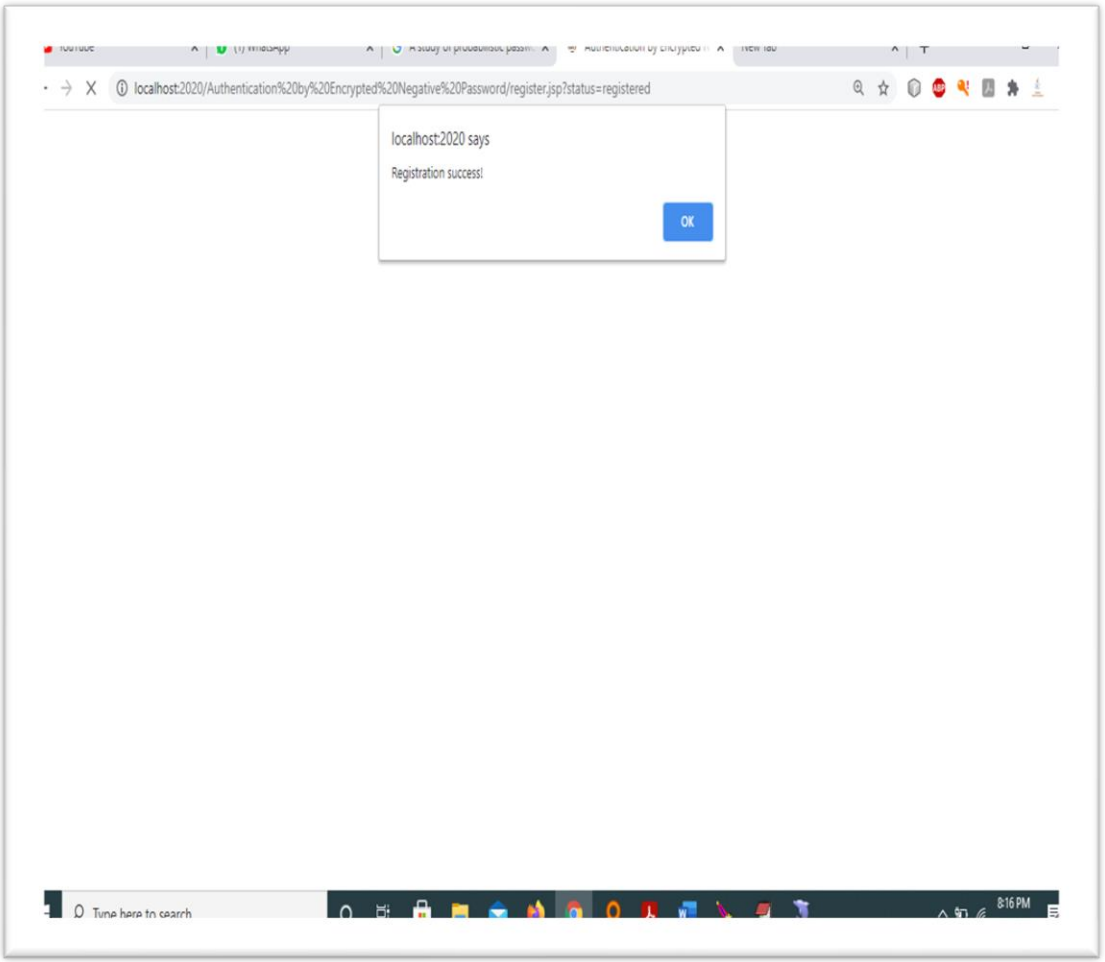
ENP GENERATION



Screen:5.1.12 ENP GENERATION

Description: Now user generates ENP through NP

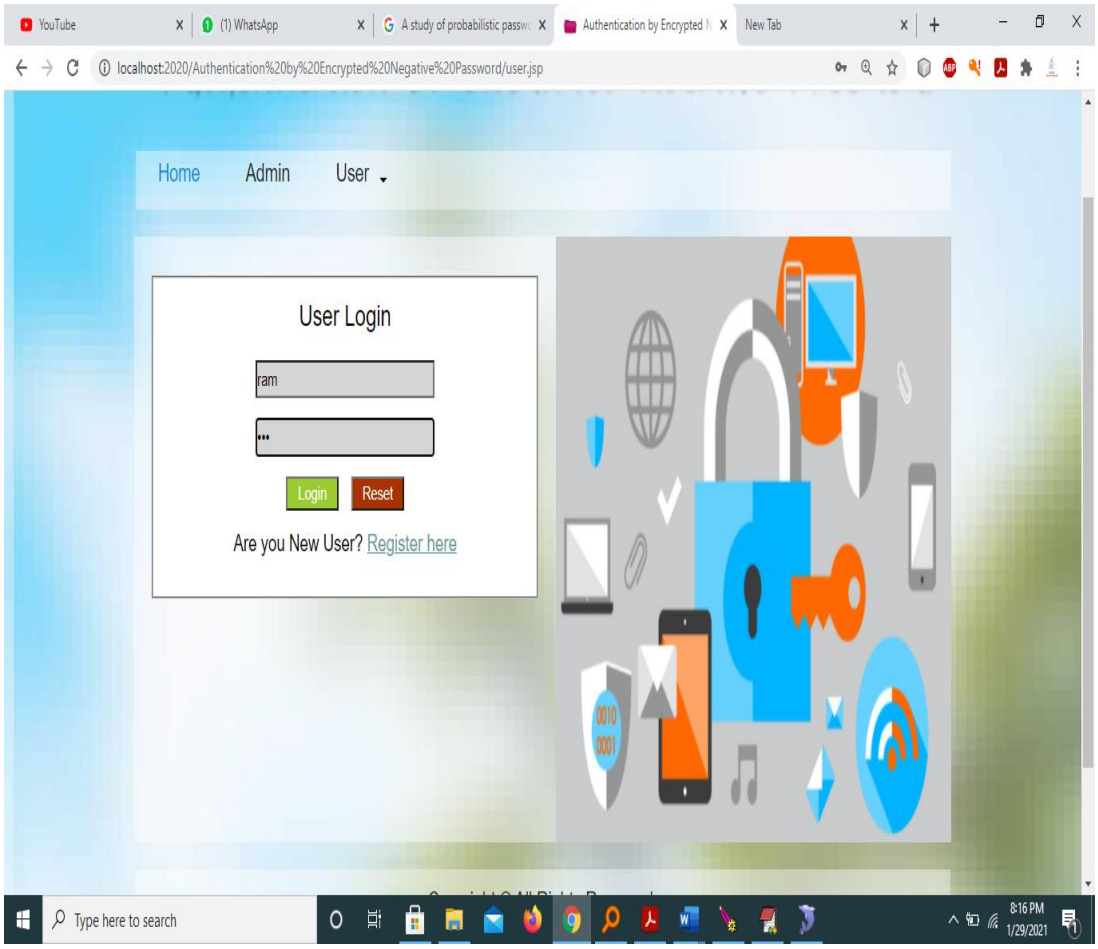
USER REGISTRATION



Screen:5.1.13 USER REGISTRATION

Description: User registration has successfully completed

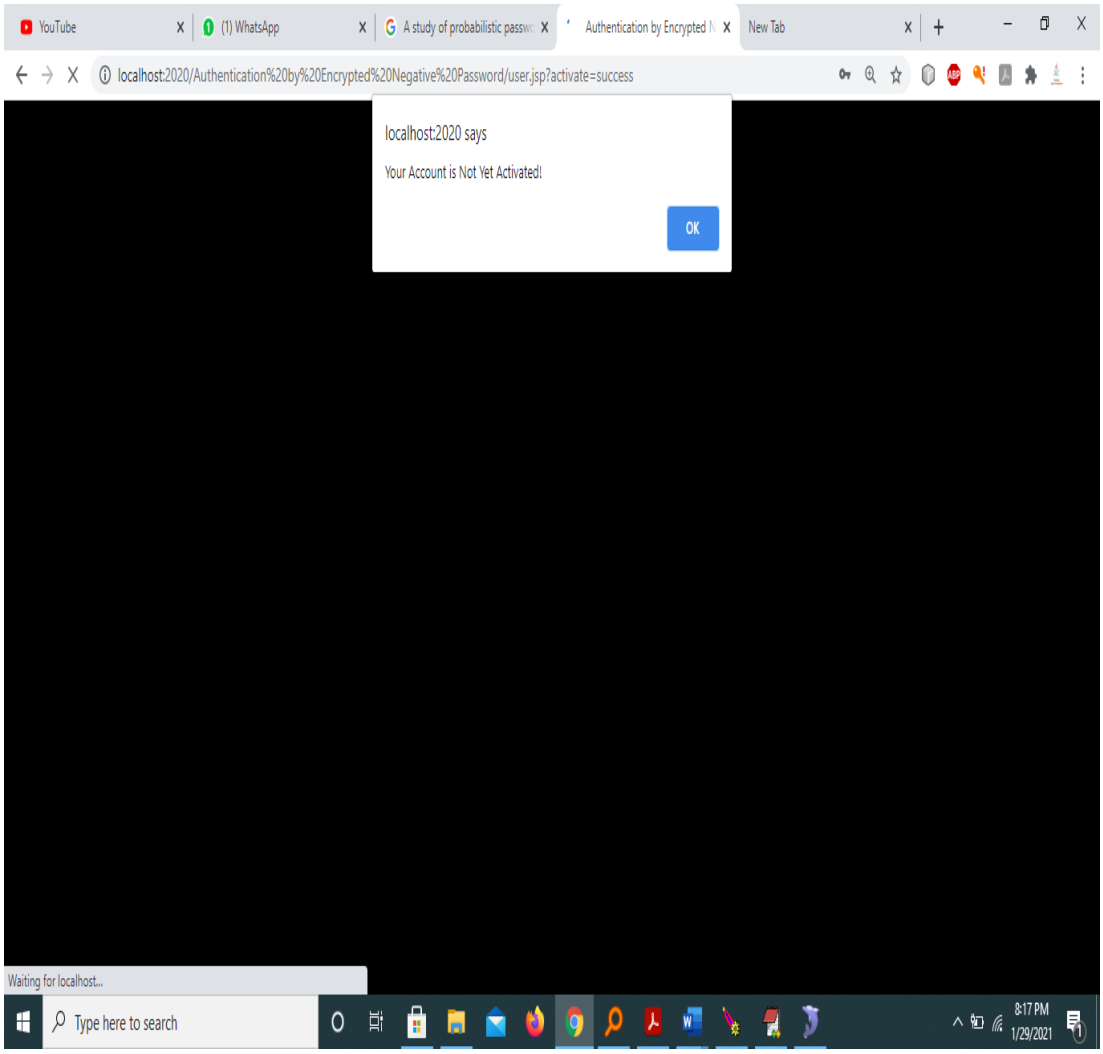
LOGIN



Screen:5.1.14 LOGIN

Description: After registration user trying to login

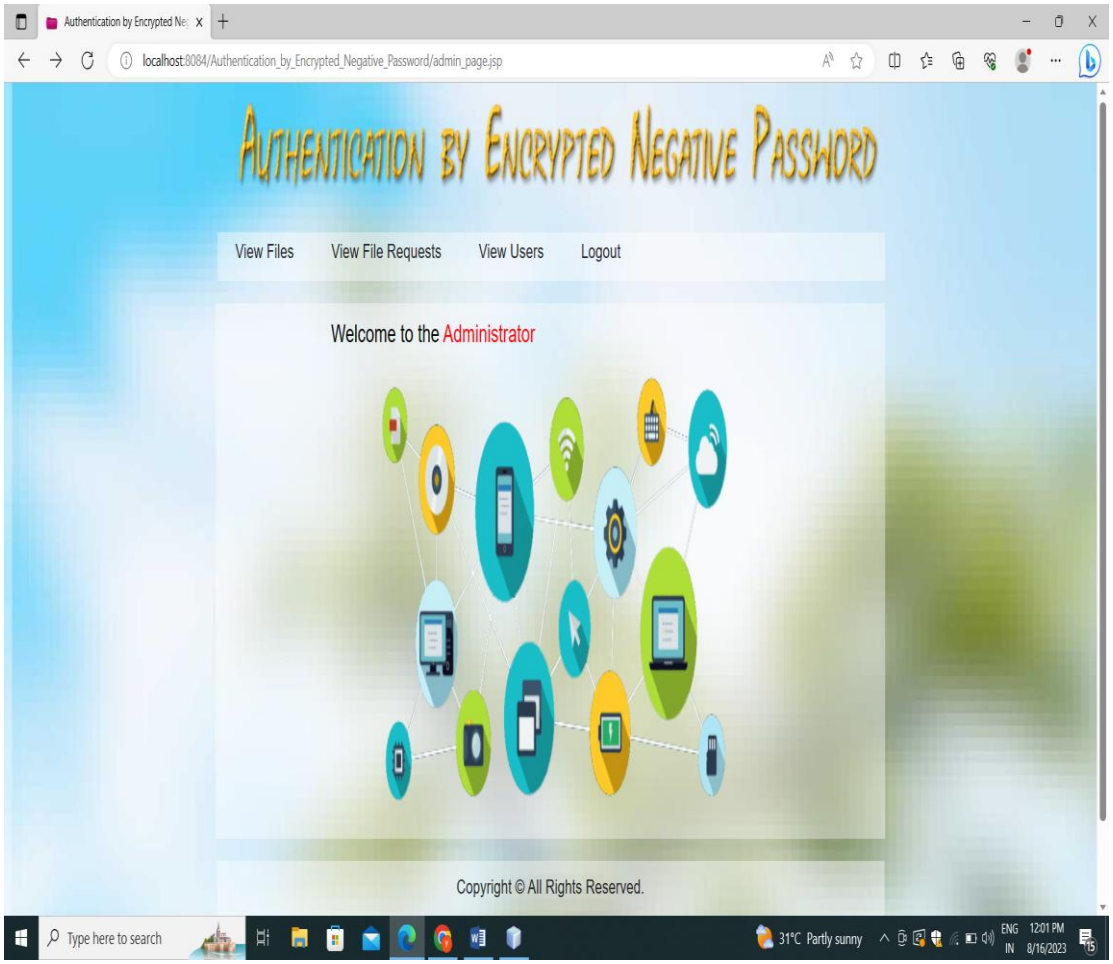
LOGIN



Screen:5.1.15 LOGIN

Description: User access denied because of account is not activated

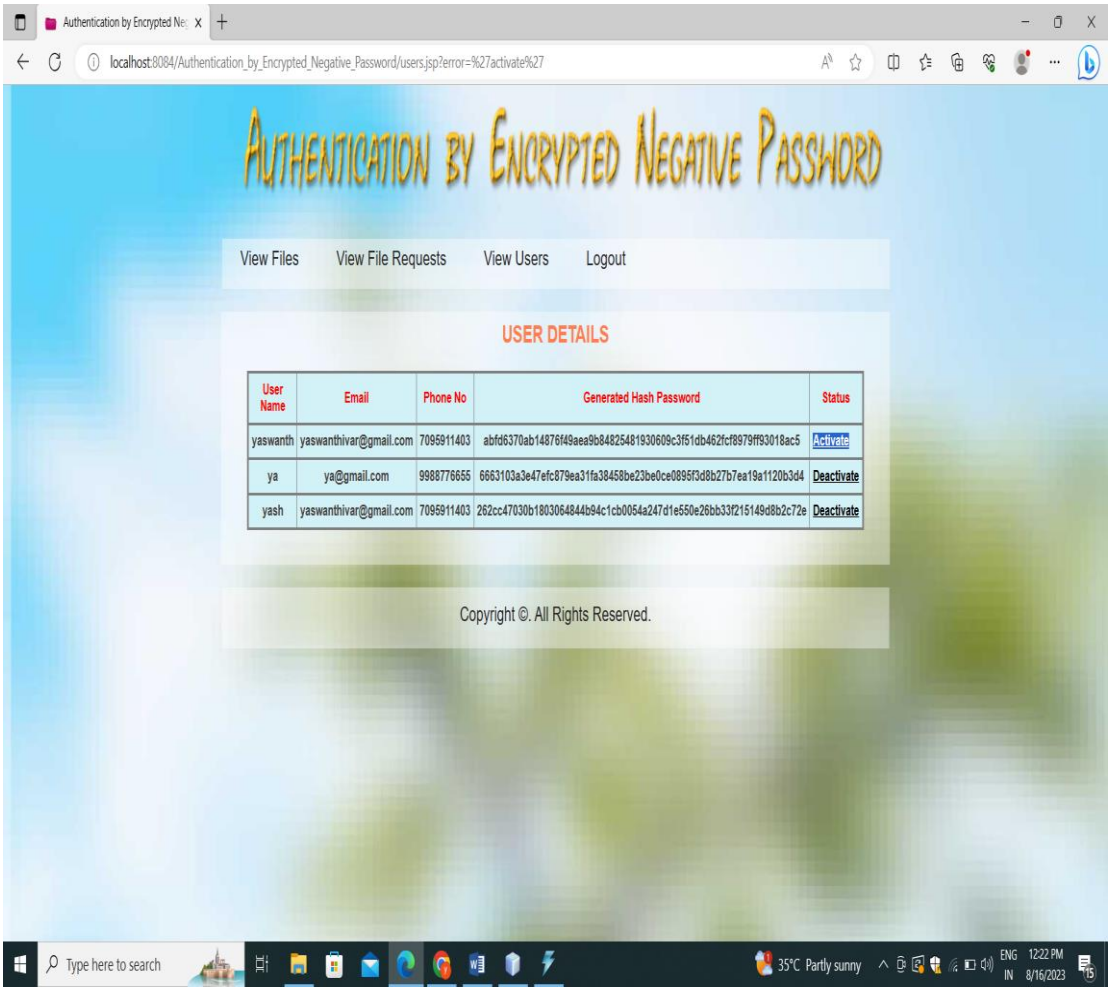
ADMIN HOME PAGFE



Screen:5.1.16 ADMIN HOME PAGFE

Description: To activate user account first login to admin account

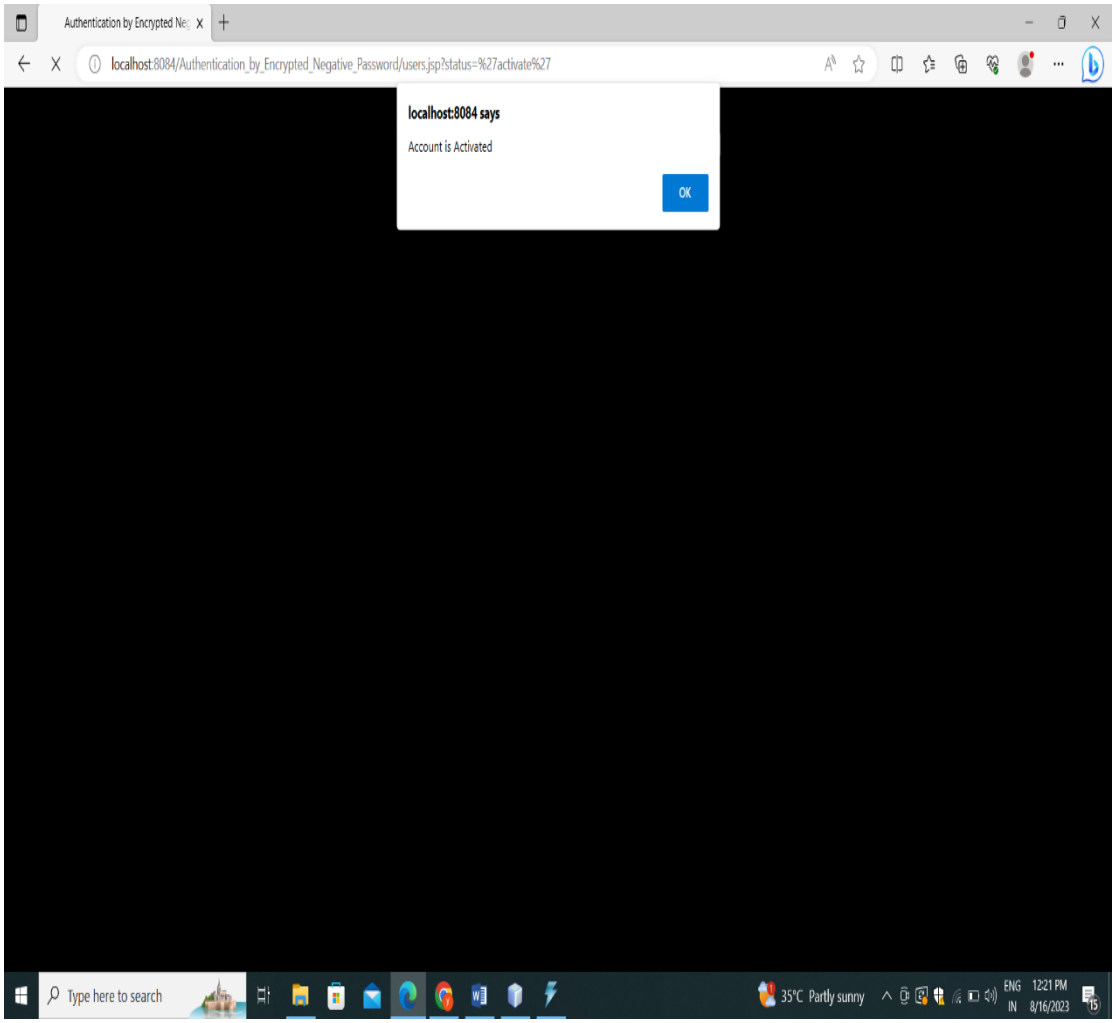
ACCOUNT STATUS



Screen:5.1.17 ACCOUNT STATUS

Description: User attempted account is not in active state so admin changing it to active state.

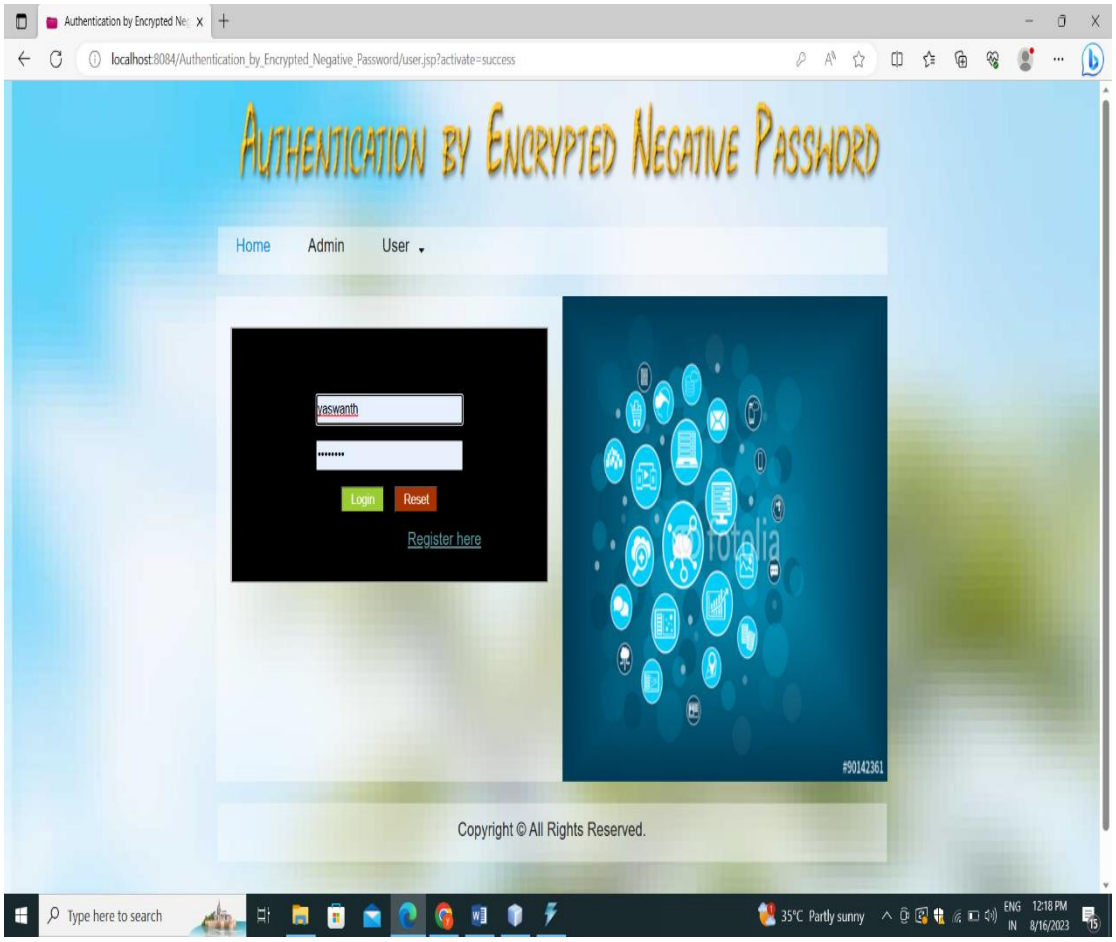
ACCOUNT ACTIVATION



Screen:5.1.18 ACCOUNT ACTIVATION

Description: After activating the acc it will popups the notification like the acc is activated

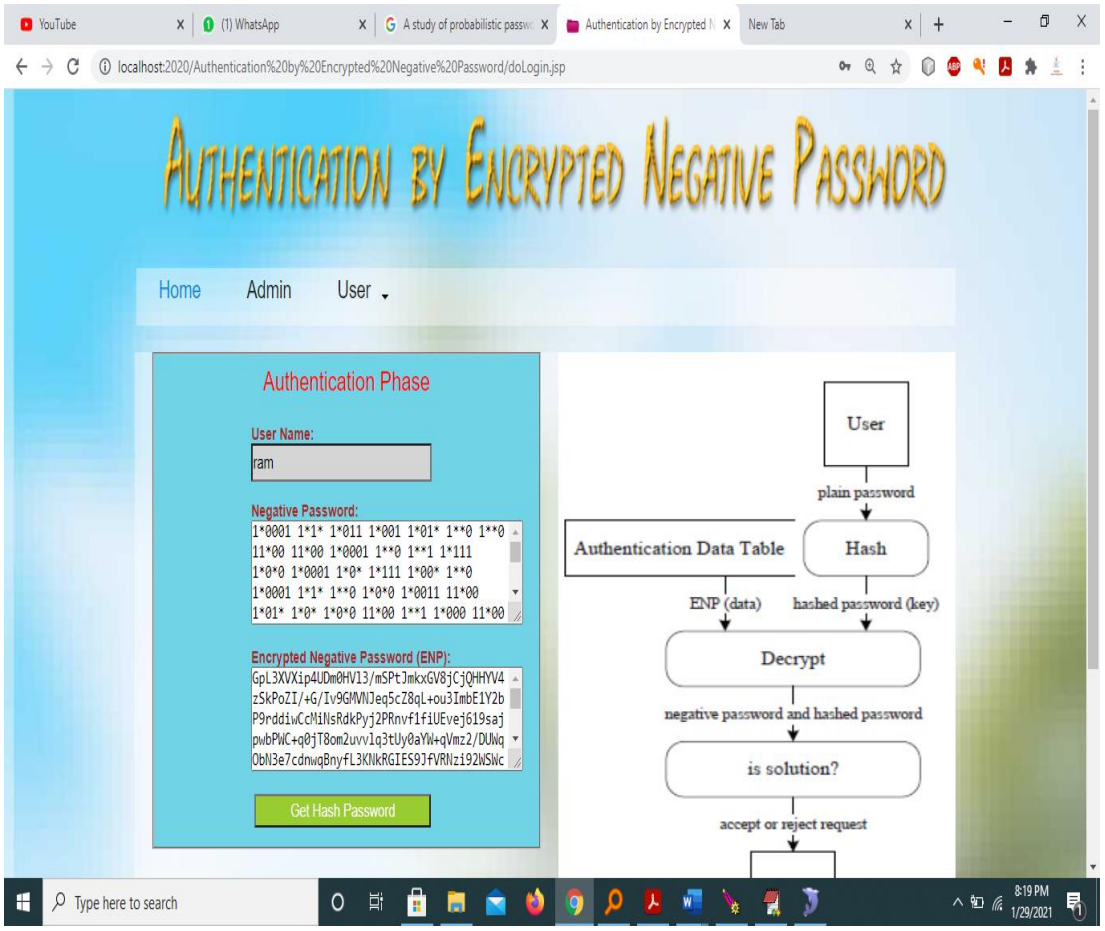
USER LOGIN



Screen:5.1.19 USER LOGIN

Description: After activating the acc user trying to login again

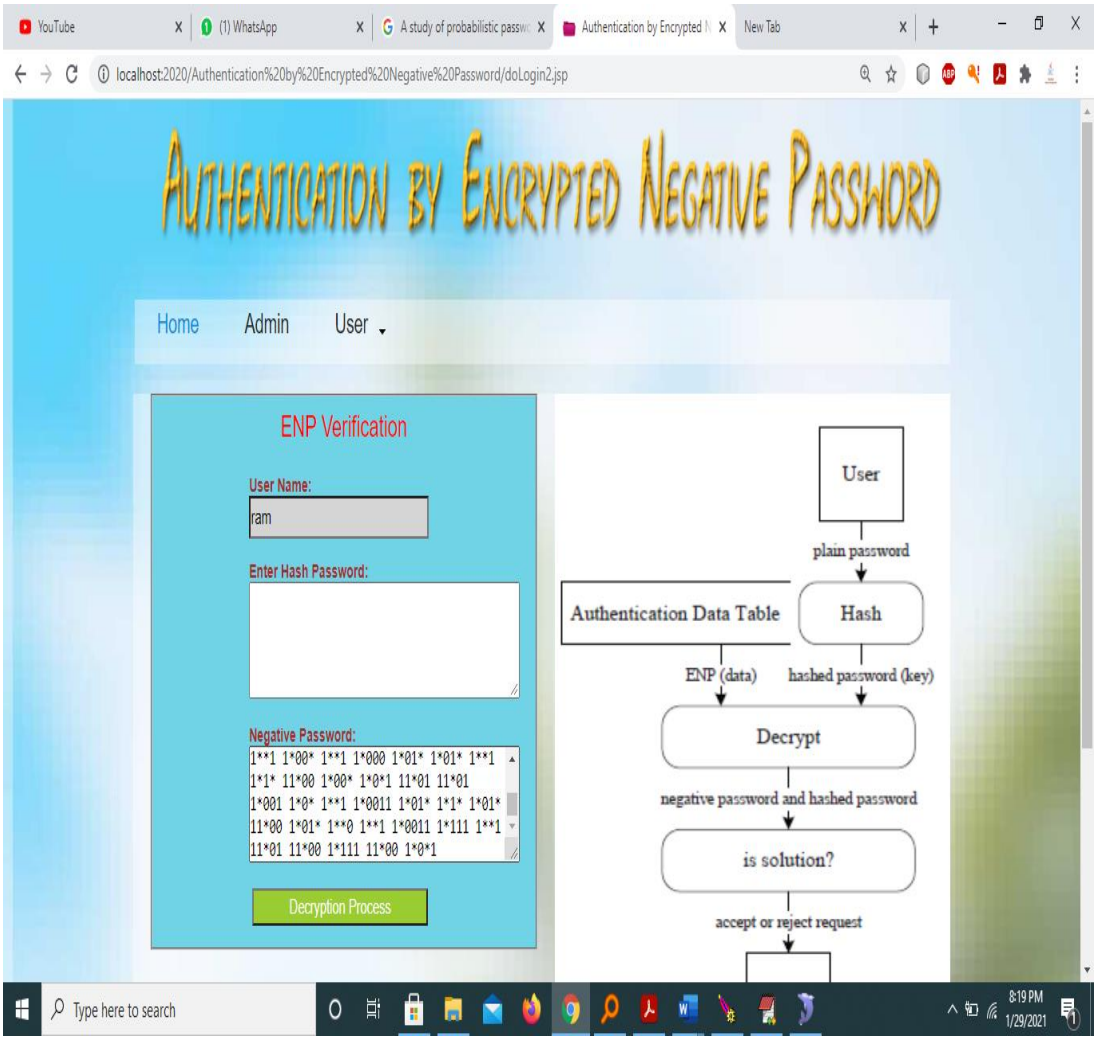
USER AUTHENTICATION



Screen:5.1.20 USER AUTHENTICATION

Description: User generating hash password

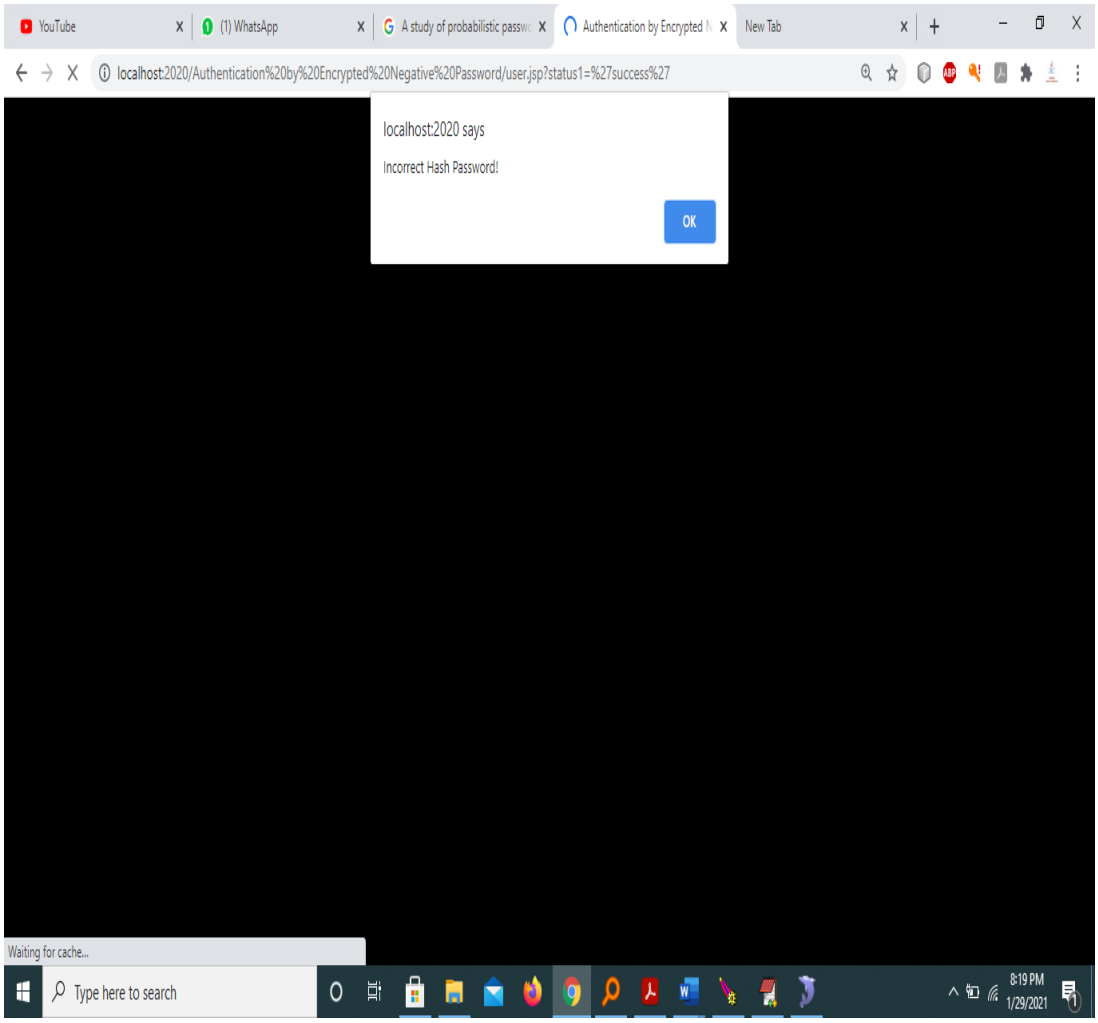
HASH PASSWORD



Screen:5.1.21 HASH PASSWORD

Description: To authenticate the account user need to provide hash password. If user Provides incorrect hash then result will be incorrect hash password.

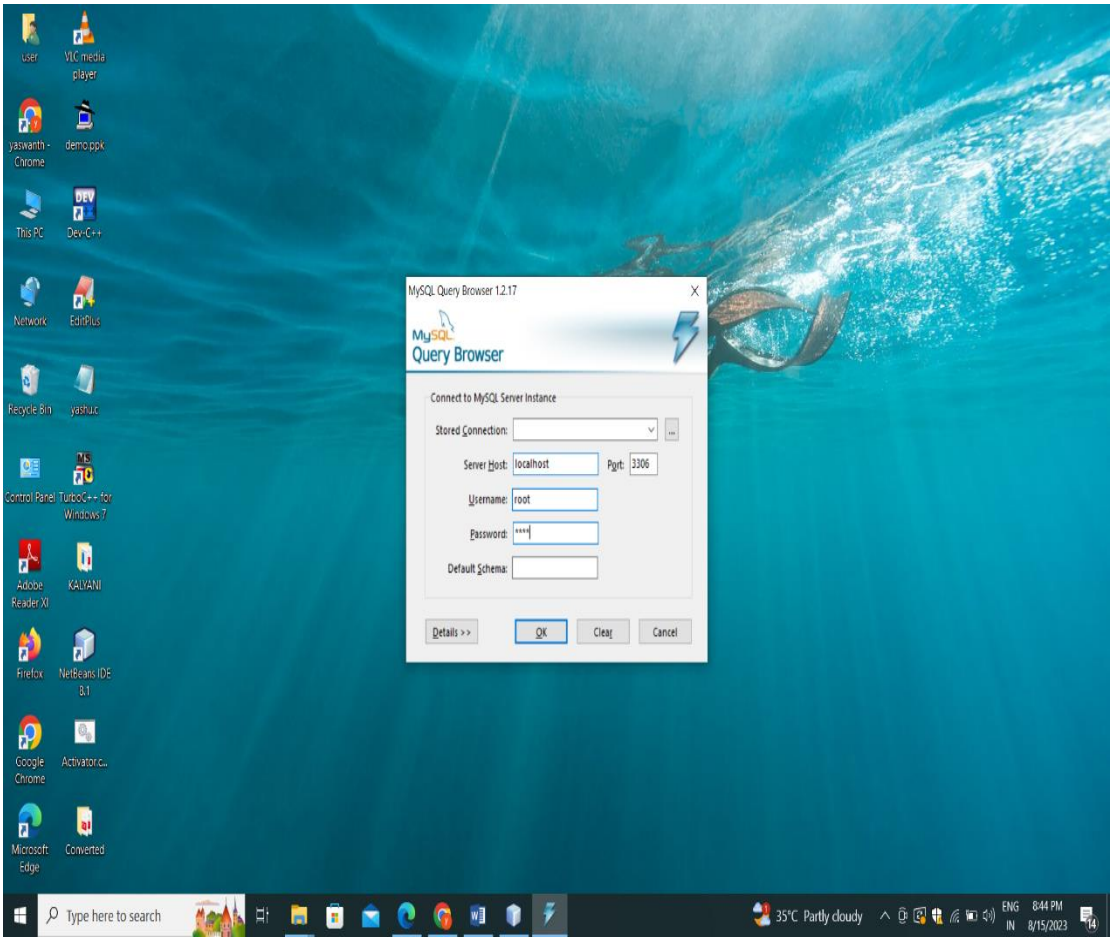
LOGIN FAILED



Screen:5.1.22 LOGIN FAILED

Description: When user provides incorrect hash password this will be the result.

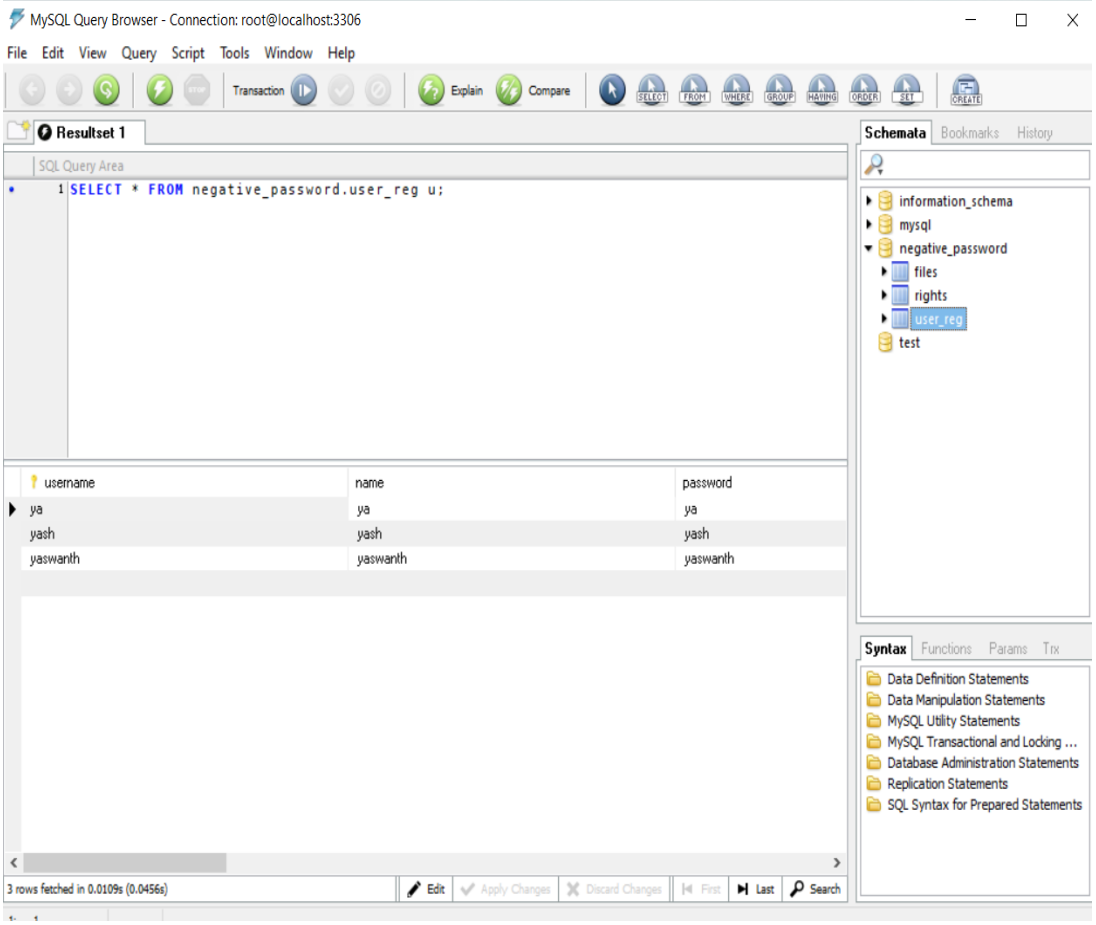
MY SQL



Screen:5.1.23 MY SQL

Description: User login to MySQL by using credentials.

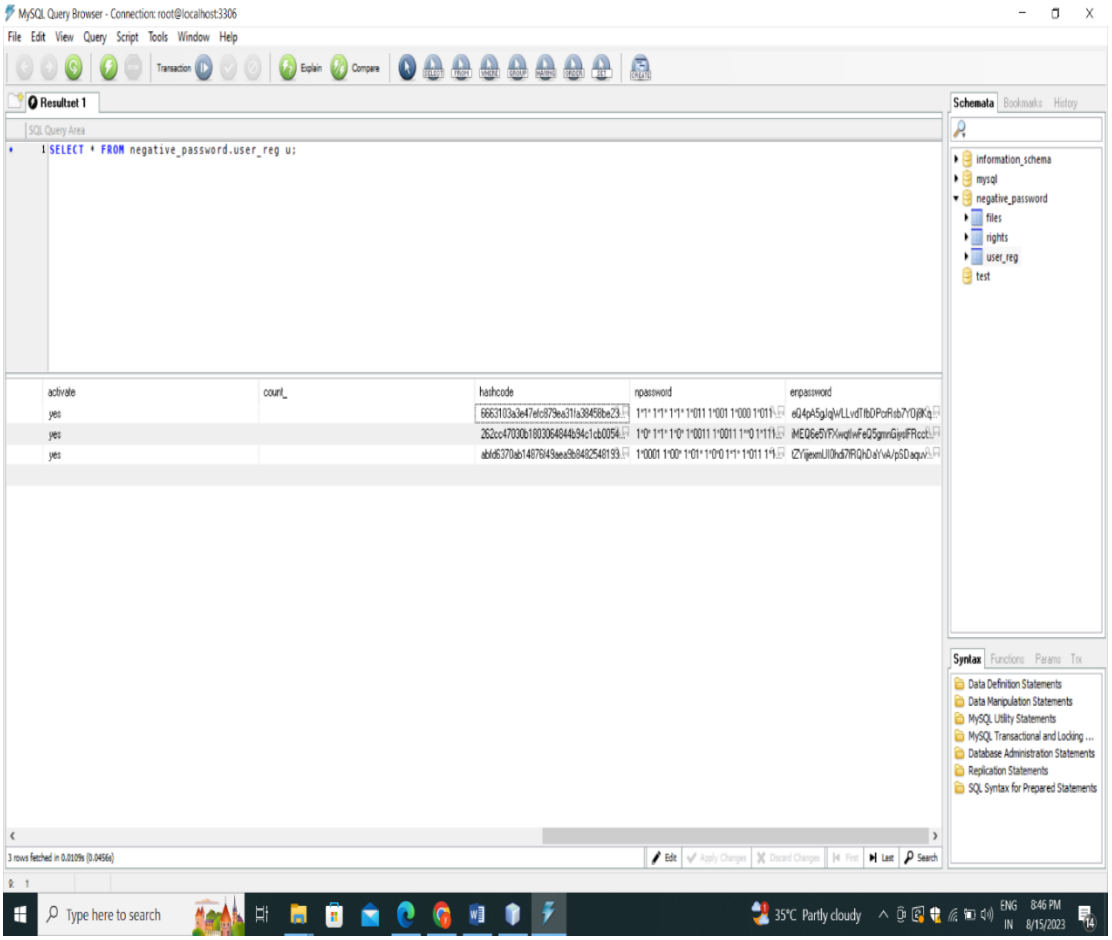
MYSQL AUTHENTICATION TABLE



Screen:5.1.24 MYSQL AUTHENTICATION TABLE

Description: Get the user data from server

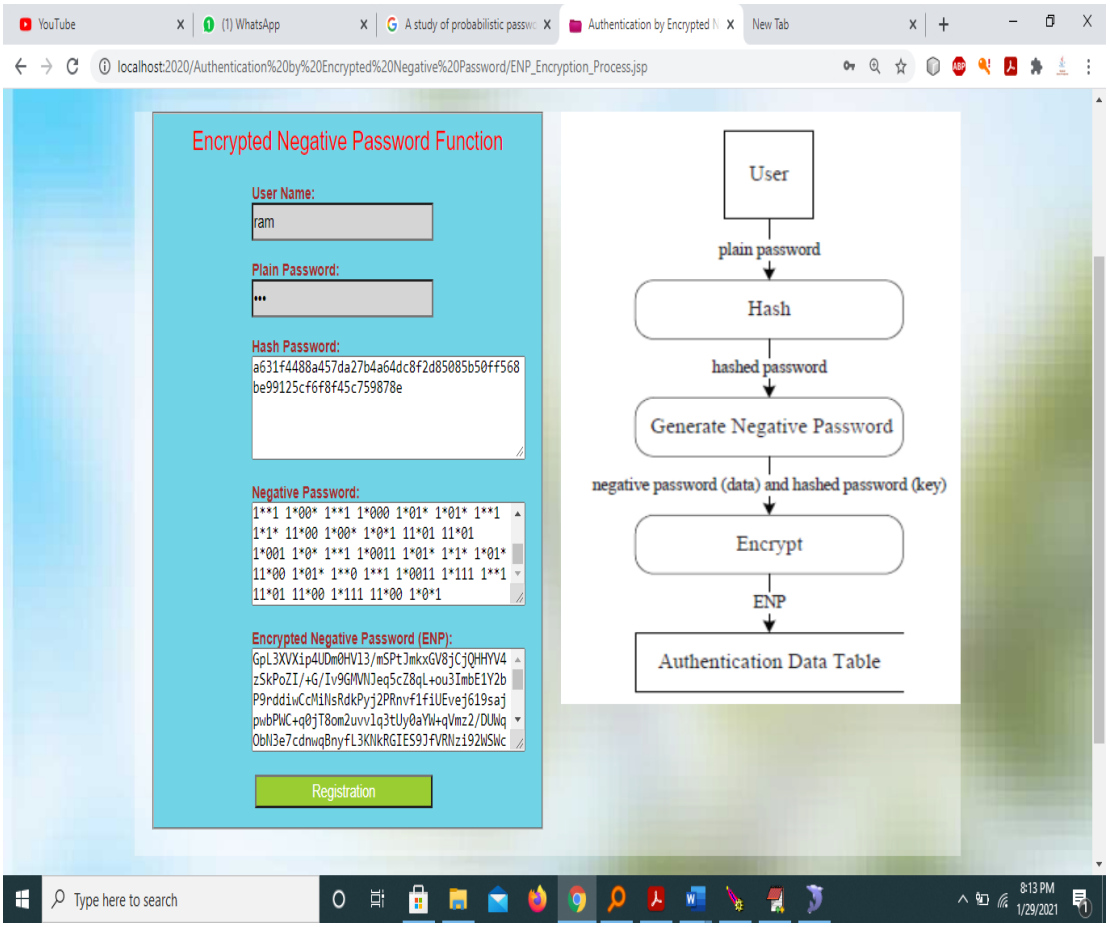
USER'S DETAILS



Screen:5.1.25 USER'S DETAILS

Description: Get the credentials from server and copy them into user login.

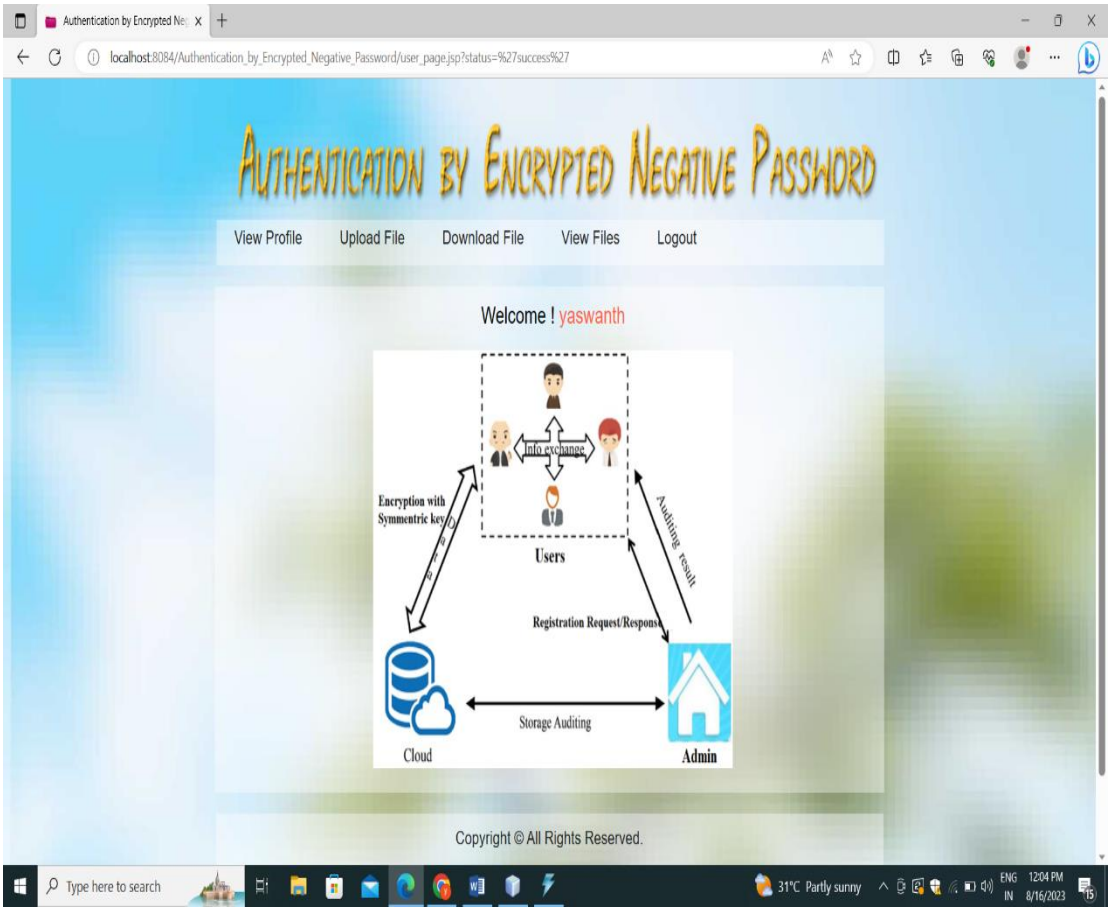
AUTHENTICATING



Screen:5.1.26 AUTHENTICATING

Description: Copy the credentials and login to account.

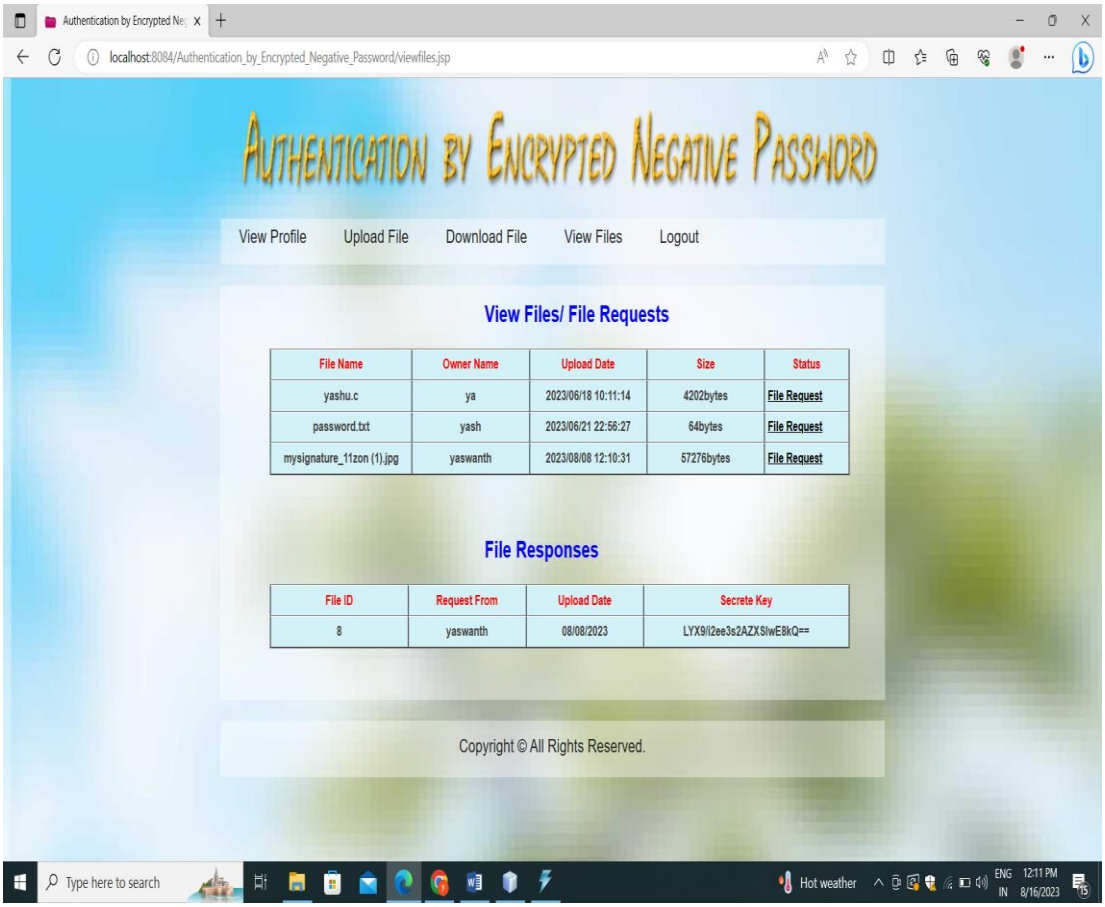
USER HOME



Screen:5.1.27 USER HOME

Description: This is user home page.

FLE RESPONSE/REQUEST



Screen:5.1.28 FLE RESPONSE/REQUEST

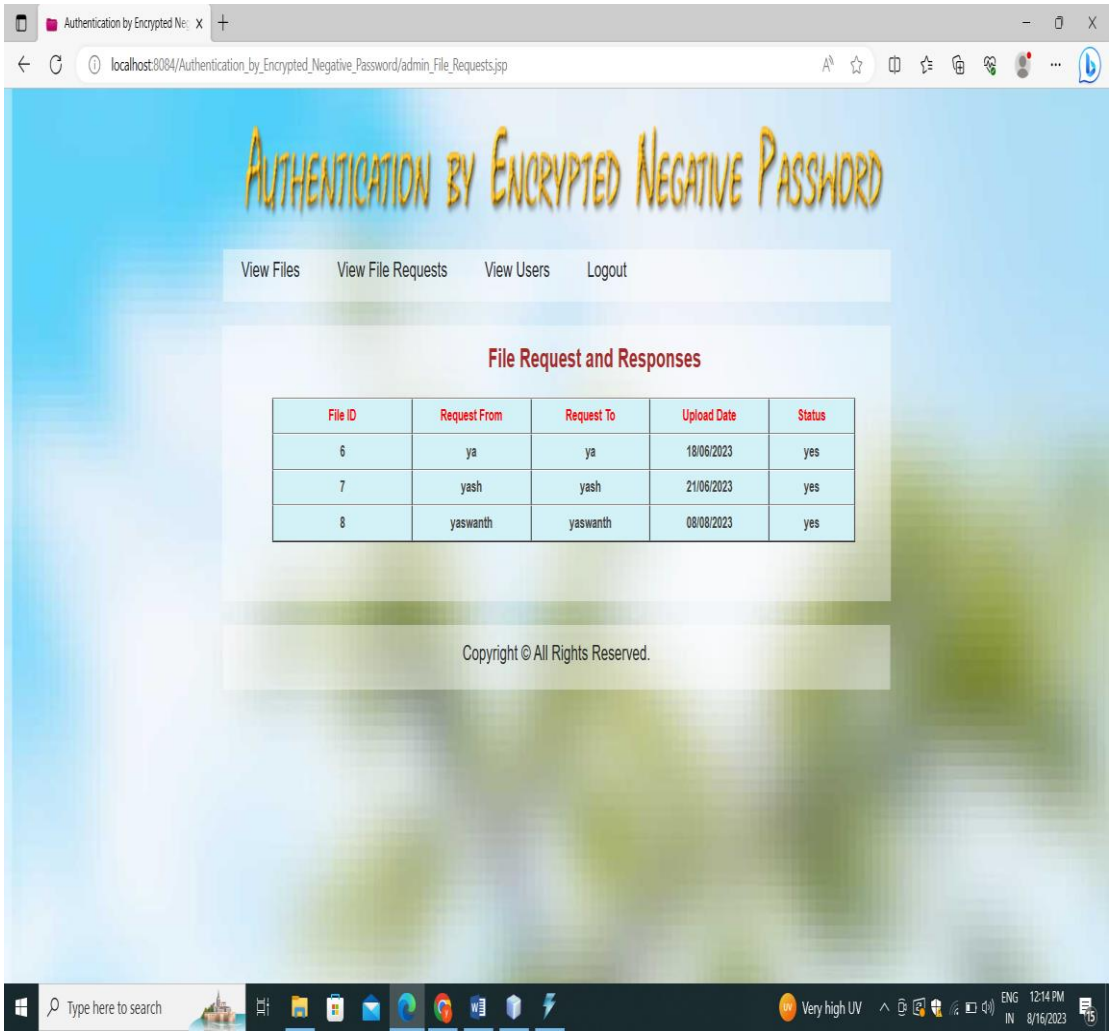
Description: User can send the response to the admin to view and access the files.

ADMIN HOME

Screen:5.1.29 ADMIN HOME

Description: These are the functions that admin has.

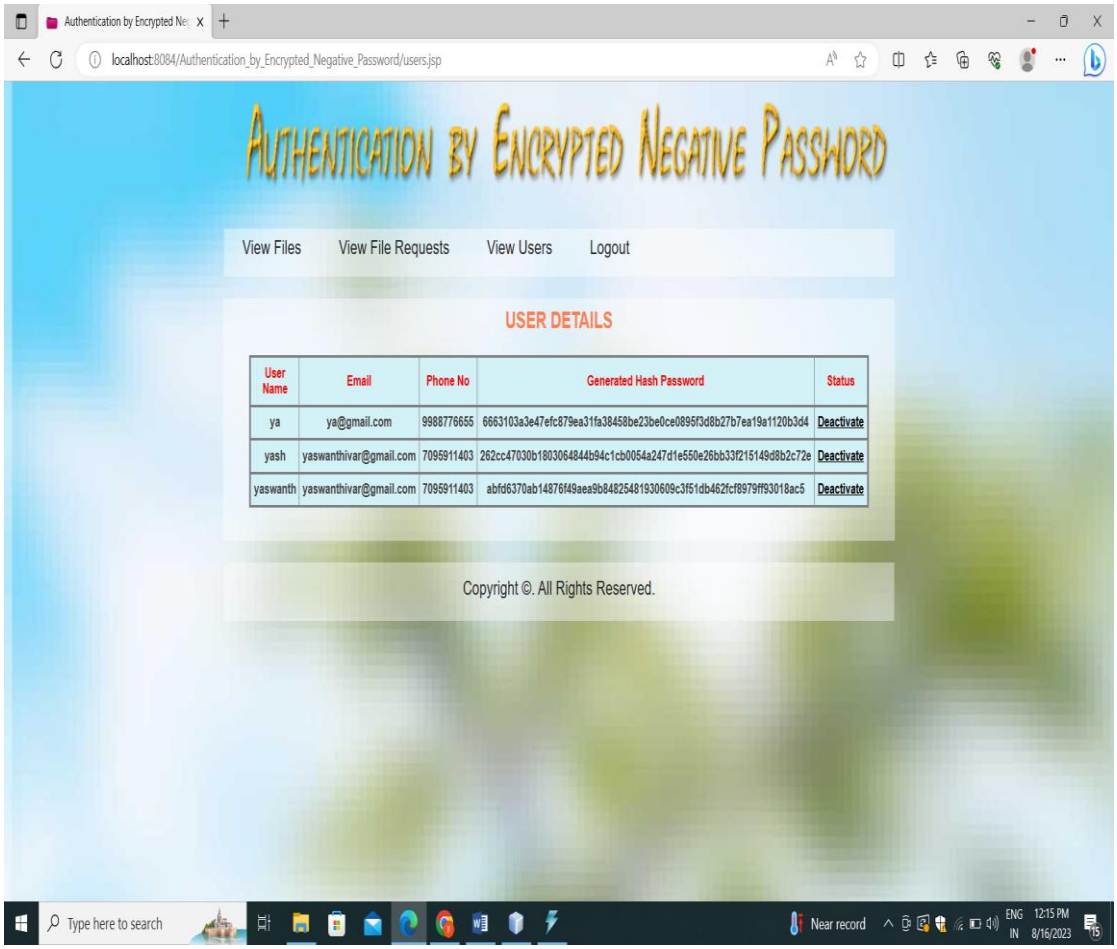
RESPONSES



Screen:5.1.30 RESPONSES

Description: Admin can check users file Id, Date & Status etc.

USER DETAILS



Screen:5.1.31 USER DETAILS

Description: Admin can check the details of users like Name, Email, Phone No, Password and admin can be able change the account status.