CHAPTER 1

INTRODUCTION

1.1.Aim and objective of the project

As the usage of cloud computing is increasing day by day and more and more sensitive information is being centralized onto the cloud, the data must be encrypted before outsourcing. The existing search techniques allow the user to search over encrypted data using keywords but these techniques account for only exact keyword search. There is no tolerance for typos and format inconsistencies which are normal user behavior. In our work, we aim to focus on secure storage using Advanced Encryption Standard (AES) and information retrieval by performing fuzzy keyword search on this encrypted data. We are proposing the implementation of an advanced fuzzy keyword search mechanism called the Wildcard based technique. In the fuzzy keyword search technique if typos and/or format inconsistencies exist in the searching input, the server returns the closest possible results based on pre-specified similarity semantics.

1.2.Motivation

As Cloud Computing becomes prevalent, more and more sensitive information are being centralized into the cloud, such as emails, personal health records, government documents, etc. By storing their data into the cloud, the data owners can be relieved from the burden of data storage and maintenance so as to enjoy the on-demand high quality data storage service. However, the fact that data owners and cloud server are not in the same trusted domain may put the outsourced data at risk. Hence we will be focusing on secure storage using an encryption technique.

Although traditional searchable encryption schemes allow a user to securely search over encrypted data through keywords and selectively retrieve files of interest, these techniques support only *exact* keyword search. Since there is no tolerance for typos and formal inconsistencies which are normal user behavior, we were motivated to implement fuzzy keyword search technique. This technique returns the matching files when user’s searching inputs exactly

match the predefined keywords or the closest possible matching keywords when exact match fails.

1.3.Cloud Computing

Cloud computing means that instead of all the computer hardware and software you're using sitting on your desktop, or somewhere inside your company's network, it's provided for you as a service by another company and accessed over the Internet, usually in a completely seamless way. Exactly where the hardware and software is located and how it all works doesn't matter to you, the user—it's just somewhere up in the nebulous "cloud" that the Internet represents [1][15].

Fig 1.1. Cloud computing

1.4 Security in cloud computing

Cloud computing involves highly available massive compute and storage platforms offering a wide range of services. One of the most popular and basic cloud computing services is storage-as-a-service (SAAS) [11][14][15].

One of the major concerns is the privacy as cloud service is generally provided by the third party. In the following, we call the company, who uses SAAS, the database owner. We call anyone who queries the company's database, the database user. And we call the cloud servers,

which store the database, the cloud server. Now we start to clarify different types of privacy challenges during the deployment of cloud service [11][14].

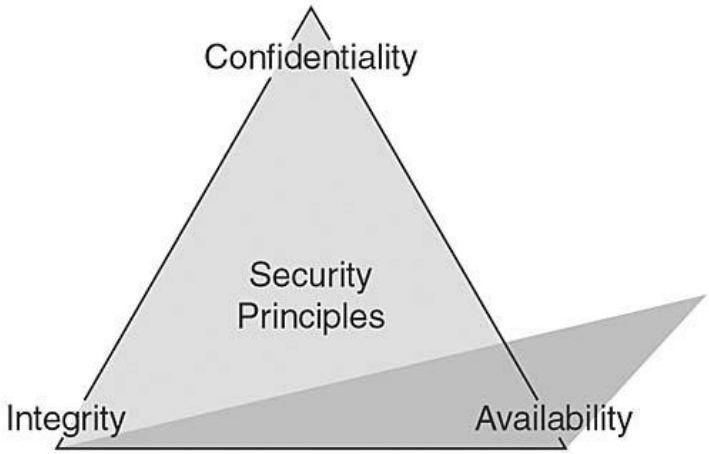


Fig 1.2. Security in cloud computing

From the perspective of the database owner, three challenges arise.

1: How to protect outsourced data from theft by hackers or malware infiltrating the cloud server? Encryption by the cloud server and authenticated access by users seems to be a straightforward solution [3].

2: How to protect outsourced data from abuse by the cloud server? A trivial solution is for the owner to encrypt the database prior to outsourcing. Subsequently, users (armed with the decryption key(s)) can download the entire encrypted database, decrypt it and perform querying in situ. Clearly, this negates most benefits of using the cloud. A more elegant approach is to use searchable encryption. Unfortunately, current searchable encryption techniques only support simple search (attribute=value), as opposed to complicated SQL, queries [3].

3: How to realize content-level fine-grained access control for users? This challenge is even harder to solve as it requires variable decryption capabilities for different users. Even trivial solution to the second challenge does not solve this challenge as it gives each user equal decryption capability (same decryption key). An ideal solution would entail the database owner issuing a given user a key that only allows the user to search and decrypt certain records [3].

From user's perspective, three more challenges arise.

1: How to query the cloud server without revealing query details? Learning user's query details means learning user's possibly sensitive search interest. In addition, by learning user queries, the cloud server gradually learns the information in the encrypted database [3].

2: How to hide query contents (e.g., values used in "attribute=value" queries) from the database owner. For the database owner to exercise access control over its outsourced data, a user should first obtain an approval from the database owner over its query contents. However, in some cases, the user may want to get the approval without revealing its query contents even to the database owner. This is the case when the user happens to be a high-level executive who is automatically qualified to search any value and is not willing to reveal query to anyone [3].

3: How to hide query contents while assuring database owner the hidden contents are authorized by some certificate authority (CA). Such challenge surfaces, for example, when the user is FBI who does not want to reveal the person it is investigating while database owner wants to get some confidence by making sure FBI is authorized by the court to do this investigation [3].

1.5 Encryption of data for security

Cryptography is the practice and study of techniques for secure communication in the presence of third parties called adversaries. More generally, it is about constructing and analyzing protocols that overcome the influence of adversaries and which are related to various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation. Modern cryptography intersects the disciplines of mathematics, computer science, and electrical engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce [8]. Cryptography is the study of Secret (crypto-) writing (-graphy) that is concealing the content of message from all except the sender and the receiver and to authenticate the correctness of message to the recipient. It is concerned with making sure that nosy people cannot read, or worse, modify messages intended for other recipients [10].

1.5.1 Data Encryption Standard (DES)

DES is based on a cipher known as the Feistel block cipher. This was a block cipher developed by the IBM cryptography researcher Horst Feistel in the early 70's. It consists of a number of rounds where each round contains bit-shuffling, non-linear substitutions (S-boxes) and exclusive OR operations. Once a plaintext message is received to be encrypted, it is arranged into 64 bit blocks required for input. If the number of bits in the message is not evenly divisible by 64, then the last block will be padded. DES performs an initial permutation on the entire 64 bit block of data. It is then split into 2, 32 bit sub-blocks, Li and R i which are then passed into 16 rounds (the subscript i in Li and Ri indicates the current round) [2][12][13].

1.5.2 Triple Data Encryption Standard (3DES)

Triple DES is simply another mode of DES operation. It takes three 64-bit keys, for an overall key length of 192 bits. In Private Encryptor, we simply type in the entire 192-bit (24 character) key rather than entering each of the three keys individually. The Triple DES DLL then breaks the user provided key into three subkeys, padding the keys if necessary so they are each 64 bits long. The procedure for encryption is exactly the same as regular DES, but it is repeated three times. Hence the name Triple DES. The data is encrypted with the first key, decrypted with the second key, and finally encrypted again with the third key [2][12][13].

1.5.3. Advanced Encryption Standard (AES)

AES emerged as a powerful replacement of DES during a competition held by National Institute of Standard and Technology (NIST). The competition was organized to develop a substitute of existing DES. Rijndael: an algorithm designed by Daemen and Rijmen was judged the best and announced to be new AES. NIST choose Rijndael, due to its simplicity and high performance. It is fast, compact, and has a very simple mathematical structure.AES is a symmetric block cipher with a block size of 128 bits. Key lengths can be 128 bits, 192 bits, or 256 bits; called AES-128, AES-192, and AES-256, respectively. AES-128 uses 10 rounds, AES-192 uses 12 rounds, and AES-256 uses 14 rounds. The main loop of AES performs the following functions:

1. SubBytes ()
2. ShiftRows ()
3. MixColumns ()

4. AddRoundKey ().

The first three functions of an AES round are designed to thwart cryptanalysis via the Methods of “confusion” and “diffusion.” The fourth function actually encrypts the data. AES formats plaintext into 16 byte (128-bit) blocks, and treats each block as a 4x4 State array. It then performs four operations in each round. The arrays contains row and column information used in the operations, especially MixColumns () and Shiftrows (). AES can be attacked using the Timing analysis Attack. This occurs when Malice (the malicious Alice) runs the Sub-Bytes method on different data and observes the time it takes for each execution [2][7][12][13].

CHAPTER 2

LITERATURE SURVEY

2.1 Related work

In this section, we mention the traditional searchable encryption technique. We have also compared the different fuzzy keyword searching techniques and justified the technique which will be used in our proposed system.

2.1.1 Searchable encryption

Traditional searchable encryption has been widely studied in the context of cryptography. In it each word in the document is encrypted independently under a special two-layered encryption construction. For each file, a Bloom filter containing trapdoors of all unique words is built up and stored on the server. To search for a word, the user generates the search request by computing the trapdoor of the word and sends it to the server. Upon receiving the request, the server tests if any Bloom filter contains the trapdoor of the query word and returns the corresponding file identifiers. To achieve more efficient search, both proposed similar "index" approaches, where a single encrypted hash table index is built for the entire file collection. In the index table, each entry consists of the trapdoor of a keyword and an encrypted set of file identifiers whose corresponding data files contain the keyword. As a complementary approach, we presented a public-key based searchable encryption scheme, with an analogous scenario to that of [6]. In their construction, anyone with the public key can write to the data stored on the server but only authorized users with the private key can search. As an attempt to enrich query predicates, conjunctive keyword search, subset query and range query over encrypted data, have also been proposed in [9]. Note that all these existing schemes support only exact keyword search, and thus are not suitable for Cloud Computing [7][16].

2.2 Fuzzy keyword search techniques

The existing searchable encryption techniques do not suit for cloud computing scenario since they support only exact keyword search. That is, there is no tolerance of minor typos and

format inconsistencies which, on the other hand, are typical user searching behavior and happen very frequently.[4][5]

The fuzzy keyword search returns the results according to the following rules:

- If the input exactly matches the pre-set keyword, the server should return the files containing the keyword.
- If typos and/or format inconsistencies exist in the searching input, the server returns the closest possible results based on pre-specified similarity semantics.[4][5]

2.2.1. Gram-based Technique

The gram of a string is a substring that can be used as a signature for efficient approximate search. While gram has been widely used for constructing inverted list for approximate string search, we use gram for the matching purpose[6][7]. We propose to utilize the fact that any primitive edit operation will affect at most one specific character of the keyword, leaving all the remaining characters untouched. In other words, the relative order of the remaining characters after the primitive operations is always kept the same as it is before the operations. With this significant observation, the fuzzy keyword set for a keyword wi with ℓ single characters supporting edit distance d can be constructed as $S_{wi,d} = \{S'_{wi,\tau}\}_{0 \le \tau \le d}$, where $S'_{wi,\tau}$ consists of all the $(\ell-\tau)$-gram from wi and with the same relative order (we assume that $d \le \ell$). For example, the gram-based fuzzy set $S_{CASTLE,1}$ for keyword CASTLE can be constructed as {CASTLE, CSTLE, CATLE, CASLE, CASTE, CASTL, ASTLE} [9][18].

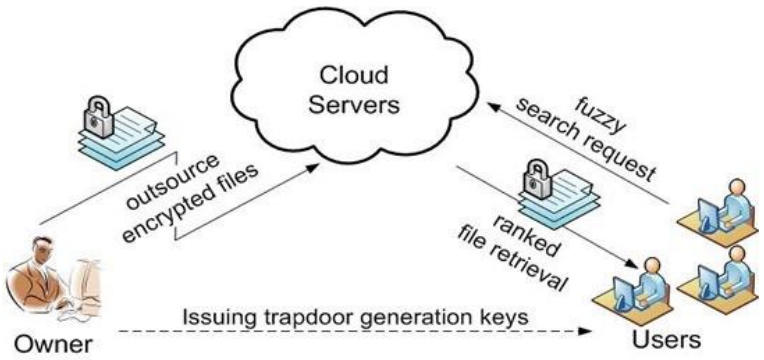


Figure 2.1 Gram based technique

2.2.2. Symbol – Based Trie – traverse Search Scheme

To enhance the search efficiency, we now propose a symbol-based trie-traverse search scheme, where a multi-way tree is constructed for storing the fuzzy keyword set {Swi,d}wi∈W over a finite symbol set. The key idea behind this construction is that all trapdoors sharing a common prefix may have common nodes. The root is associated with an empty set and the symbols in a trapdoor can be recovered in a search from the root to the leaf that ends the trapdoor. All fuzzy words in the trie can be found by a depth-first search. Assume Δ = {αi} is a predefined symbol set, where the number of different symbols is | Δ |= 2n, that is, each symbol αi∈ Δ can be denoted by n bits. [6][9][18]

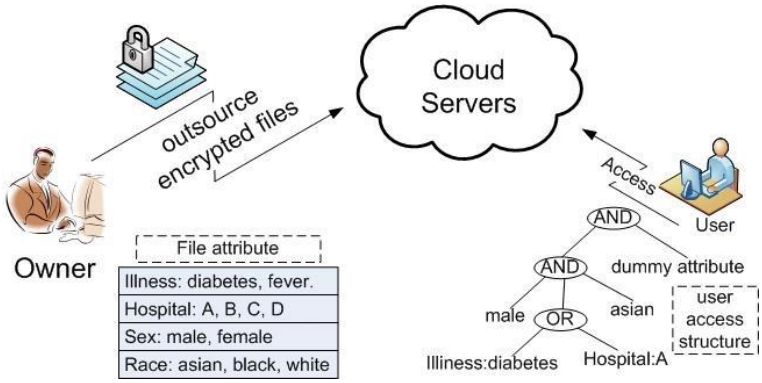


Figure 2.2 Symbol based trie traverse technique

2.2.3. Wildcard based technique

In the above straightforward approach, all the variants of the keywords have to be listed even if an operation is performed at the same position. Based on the above observation, we proposed to use an wildcard to denote edit operations at the same position. The wildcard-based fuzzy set of wi with edit distance d is denoted as Swi,d={S ′ wi,0 , S′ wi,1 , · · · , S ′ wi,d}, where S ′ wi,τ denotes the set of words w ′ i with τ wildcards. Note each wildcard represents an edit operation on wi . The procedure for wildcard-based fuzzy set construction is shown in Algorithm 1[6][9]. For example, for the keyword CASTLE with the pre-set edit distance 1, its wildcard based fuzzy keyword set can be constructed as SCASTLE,1 = {CASTLE, *CASTLE, *ASTLE, C*ASTLE, C*STLE, · · · , CASTL*E, CASTL*, CASTLE*}. The total number of variants on CASTLE constructed in this way is only 13 + 1, instead of 13 × 26 + 1 as in the above exhaustive enumeration approach when the edit distance is set to be 1. Generally,

for a given keyword wi with length ℓ, the size of Swi,1 will be only 2ℓ + 1 + 1, as compared to (2ℓ + 1) × 26 + 1 obtained in the straightforward approach.

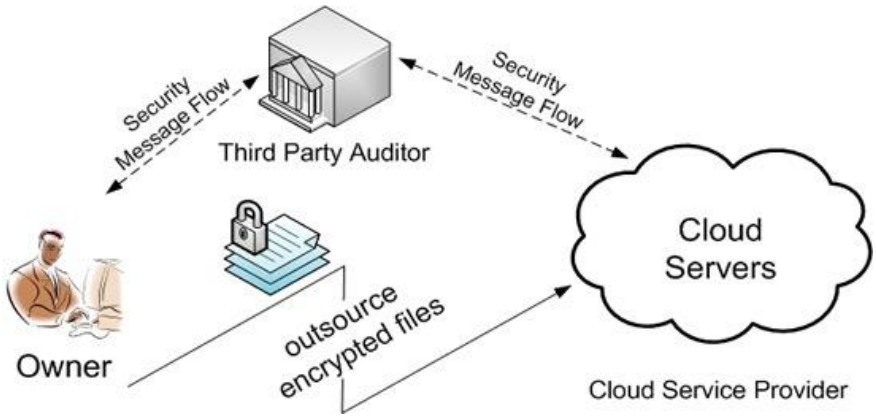


Fig 2.3. wildcard based technique

Even though Gram-Based technique is an efficient technique for constructing fuzzy set which is based on grams, the major drawback is that it has limited flexibility to include all the fuzzy sets. In the Symbol-based Trie-Traverse technique, the key idea is that all trapdoors sharing a common prefix may have common nodes. But the drawback of this construction is that storage required for the tree is quite large. Thus we have used the Wildcard-Based technique in which larger the pre-set edit distance, the more storage overhead can be reduced.

CHAPTER 3

PROPOSED SYSTEM

3.1 Problem Definition

To securely search over encrypted data, searchable encryption techniques have been developed in recent years. Searchable encryption schemes usually build up an index for each keyword of interest and associate the index with the files that contain the keyword. By integrating the trapdoors of keywords within the index information, effective keyword search can be realized while both file content and keyword privacy are well-preserved[1].

Although allowing for performing searches securely and effectively, the existing searchable encryption techniques do not suit for cloud computing scenario since they support only exact keyword search. That is, there is no tolerance of minor typos and format inconsistencies which, on the other hand, are typical user searching behavior and happen very frequently. As common practice, users may search and retrieve the data of their respective interests using any keywords they might come up with. It is quite common that users' searching input might not exactly match those pre-set keywords due to the possible typos. This significant drawback of existing schemes signifies the important need for new techniques that support searching flexibility, tolerating both minor typos and format inconsistencies[17][18].

Therefore, in this project, we consider a cloud data system consisting of cloud server and data user. Encrypted data files are stored in a cloud server and the cloud server provides the search service for the authorized users over the encrypted files. The files can be uploaded in two ways:

- Public: It will be accessed by any user who logs into the system.
- Private: It will require the user to specify a password during uploading. This password will be hashed and stored into the database. It will be required while downloading the private file by the user.

An authorized user types in a keyword to selectively retrieve data files. The cloud server maps the search request to a set of data files. Each file is indexed based on a file ID and linked to a

set of fuzzy keywords. The fuzzy keyword search returns the results according to the following rules:

- If the input exactly matches the pre-set keyword, the server should return the files containing the keyword.
- If typos and/or format inconsistencies exist in the searching input, the server returns the closest possible results based on pre-specified similarity semantics.

The search should be conducted in a secure manner that allows data files to be securely retrieved while revealing as little information as possible to the cloud server. In this project, we propose a method which ensures efficient yet privacy-preserving fuzzy keywords search services over encrypted cloud data.[5]

3.1.2 AES - Encryption Algorithm

For the AES algorithm, the length of the Cipher Key, K, is 128, 192, or 256 bits. The key length is represented by Nk = 4, 6, or 8, which reflects the number of 32-bit words (number of columns) in the Cipher Key[19].

For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by Nr, where Nr = 10 when Nk = 4, Nr = 12 when Nk = 6, and Nr = 14 when Nk = 8[19].

For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations: 1) byte substitution using a substitution table (S-box), 2) shifting rows of the State array by different offsets, 3) mixing the data within each column of the State array, and 4) adding a Round Key to the State[19].

3.1.3 Wild-Card Based Technique - Searching Algorithm

In the above straightforward approach, all the variants of the keywords have to be listed even if an operation is performed at the same position. Based on the above observation, we propose to use a wildcard to denote edit operations at the same position. [7][9].

3.2 Fuzzy keyword search scheme over encrypted data

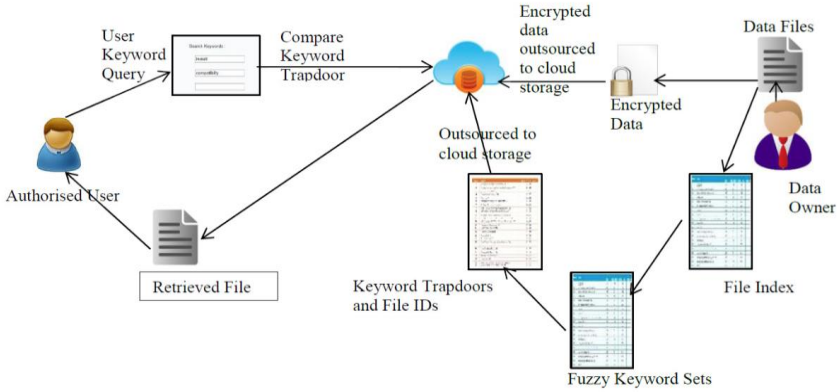


Fig 3.1. System architecture

1) We begin by constructing the fuzzy keyword set S for each keyword with edit distance d. The edit distance between two words w1 and w2 is the number of operations required to transform one of them into the other. The three primitive operations are 1) Substitution: changing one character to another in a word; 2) Deletion: deleting one character from a word; 3) Insertion: inserting a single character into a word.
2) The intuitive way to construct the fuzzy keyword set of wi is to enumerate all possible words with edit distance d and list them. For example, the following is the listing variants after a substitution operation on the first character of keyword CASTLE: {AASTLE, BASTLE,DASTLE, ⋯ , YASTLE, ZASTLE}. Based on the resulted fuzzy keyword sets, the fuzzy search over encrypted data is conducted.
3) To build an index for keywords, the data owner computes fuzzy set for each keyword and the corresponding file id. The index table and encrypted data files are outsourced to the cloud server for storage.
4) To search with w, the authorized user computes the trapdoor Tw of w and sends it to the server.
5) Trapdoors of Keywords: Trapdoors of the keywords can be realized by applying a one-way function f, which is: Given a keyword wi and a secret key sk, we can compute the trapdoor of wi as Twi = f(sk, wi).
6) Upon receiving the search request Tw, the server compares it with the index table and returns all the possible encrypted file identifiers according to the fuzzy keyword search. The user decrypts the returned results and retrieves relevant files of interest.[5]

3.3. System specification

3.3.1. ASP.NET

ASP.NET is a development framework for building web pages and web sites with HTML, CSS, JavaScript, server side scripting and Programming Language. ASP stands for Active Server Pages.

.NET is language independent, which means we can use any .NET supported language to make .NET applications. The most common languages for writing ASP.NET applications are C# and VB.NET. While VB.NET is directly based VB (Visual Basic), C# is introduced with the .NET framework, and is therefore comparatively a new language. Some people call C# as the .NET language, but according to Microsoft, we can create same applications, no matter we use C# or VB.NET. The two languages are not very different. So, if we get comfortable using one, the other must be very easy to learn. In the proposed system, we have used C#.

ASP.NET Framework is used to create dynamic websites, web applications and web services. It is built on .NET Framework.

ASP.NET Framework has various capabilities like Hosting Model, Site/Service Management, Protocol Abstraction, Security, Caching capability, Routing and Model Binding etc.

3.3.2 Visual Studio

The Visual Studio IDE offers a set of tools that help you to write and modify the code for your programs, and also detect and correct errors in your programs. Using Visual Studio you can build Windows Store apps, desktop apps, mobile apps, ASP.NET web apps, and web services. We used Visual Studio 2015 for developing the website and publishing it on azure.

3.3.3 ADO.NET

ADO.NET consists of a set of objects that expose data access services to the .NET environment. It is a data access technology from Microsoft .Net Framework which provides communication between relational and non-relational systems through a common set of components. Note that 'ADO' in ADO.NET stands for ActiveX Data Object.

System.Data namespace is the core of ADO.NET and it contains classes used by all the data providers. ADO.NET is designed to be easy to use. The Visual Studio software provides several wizards and other features that we can use to generate ADO.NET data access code.

3.3.3.1 Data Providers

The two key components of ADO.NET are Data Providers and DataSet. The Data Provider classes are meant to work with different kinds of data sources. They are used to perform all data-management operations on specific databases. The DataSet class provides mechanisms for managing data when it is disconnected from the data source.

A data provider contains Connection, Command, DataAdapter, and DataReader objects. These four objects provide the functionality of Data Providers in the ADO.NET.

3.3.4 Microsoft SQL Server

Microsoft SQL Server is a Relational Database Management System developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet).

3.3.4.1 Connection String

A connection string provides the information that a provider needs to communicate with a particular database. It includes parameters such as the name of the driver, server and database, along with the security information such as user name and password.
Microsoft SQL Server Connection String example is:
connetionString="Data Source=ServerName;Initial Catalog=Databasename; User ID=UserName;Password=Password"

3.3.4.2 Web.config and ConnectionString

Connection strings can be stored in the Web.config file and the configuration entries can be referenced in data source controls. ConnectionStrings element can also be created within the configuration element. This can be done by creating a child element and placing the connection strings there.

The Close() method in SqlConnection Class is used to close the Database Connection. The Close method rolls back any pending transactions and releases the connection from the SQL Server Database.

3.3.6 State Management

ASP.NET Session state provides a place to store values that persist across the page requests. Values stored in Session are stored on the server and remain in memory until they are explicitly removed or until the Session expires. Hence, a Session state is a collection of objects, tied to a session stored on a server.

3.3.7 Cryptography

One of the biggest challenges when dealing with the security and encryption for a system, is the determination of the correct ciphering paradigm. In .NET, there is a copious amount of libraries available for use in the System.Cryptography namespace.

3.3.7.1 Encryption using AES

Firstly the original text i.e. clear text is converted into bytes.For the AES algorithm to perform encryption, a Key and IV (Initialisation Vector) are generated using the derived bytes and the symmetric key.

Using MemoryStream and CryptoStream the clear text is encrypted and written to a byte array. Finally the byte array is converted to Base64String and returned which is the final outcome i.e. the corresponding encrypted text.

3.3.7.2 Decryption using AES

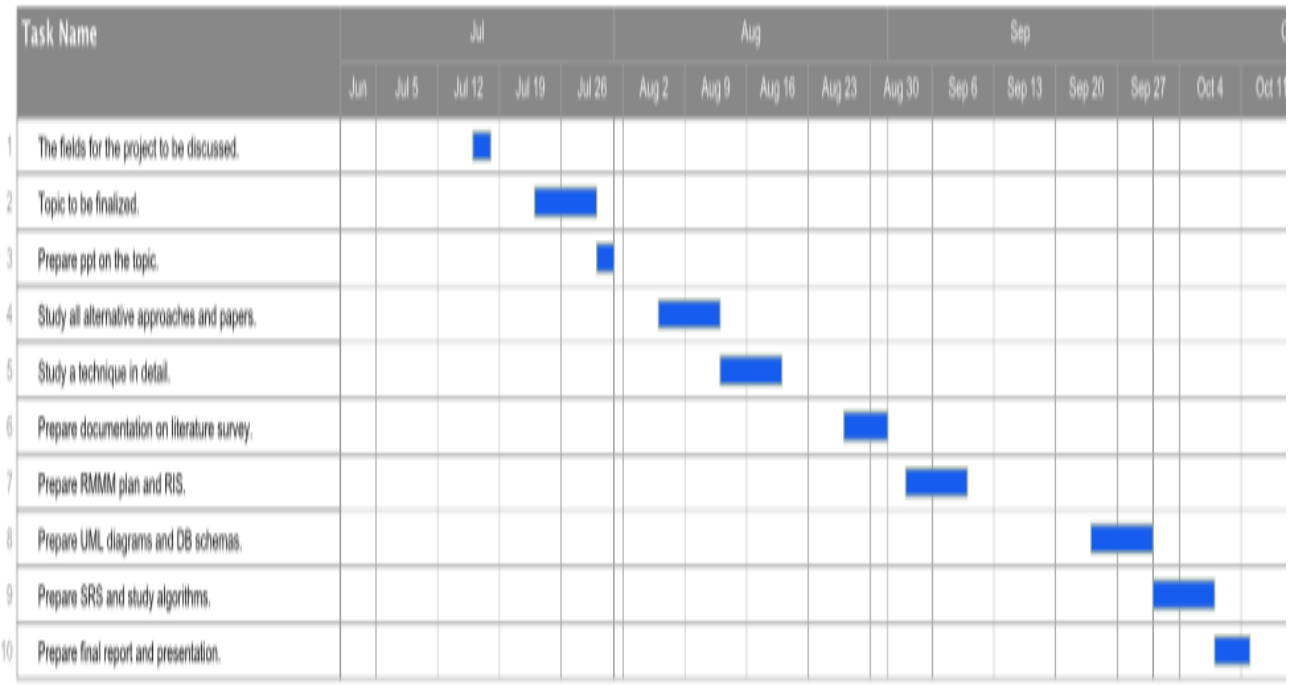
Firstly the encrypted text i.e. cipher text is converted into bytes. Similar to the encryption process,a Key and IV are generated using the derived bytes and the symmetric key.

Using MemoryStream and CryptoStream the cipher text is decrypted and written to a byte array. Finally the byte array is converted to Base64String and returned, which is the decrypted original text.

CHAPTER 4

ANALYSIS AND DESIGN

4.1. Timeline chart



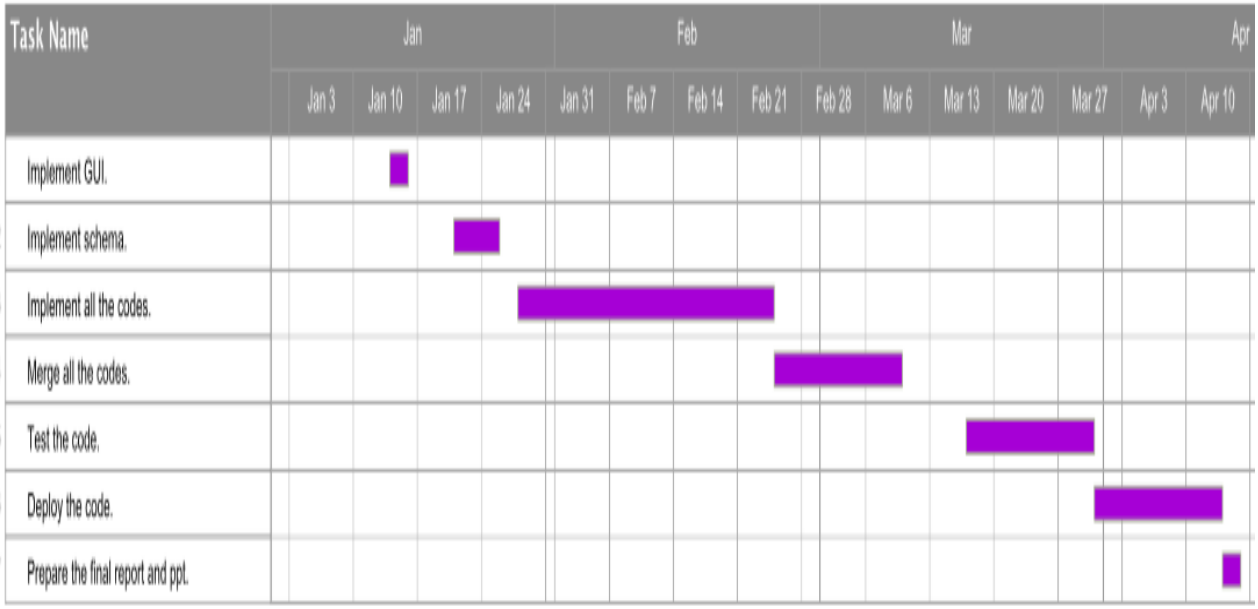


Fig 4.1Timeline Chart

4.2. Risk analysis

The RMMM plan for the project is as given below:

4.1.1. Risk Table

Risks	Probability	Impact
Deviation From Time Line Chart	30%	-1
Changes in Requirement	20%	2
Poor Comments in Code	20%	4

Impact Values:

1 – Catastrophic

2 – Critical

3 – Marginal

4 – Negligible

4.1.2. RMMM

1. Changes in Requirements

Mitigation

In order to prevent this from happening, meetings (formal and informal) will be held with the project mates and the guide on a routine business. This insures that the product we are producing is up to the mark.

Monitoring

The meetings with the project mates should ensure that the project mates and our guide understand each other and the requirements for the product.

Management

Should the development team come to the realization that their idea of the product requirements differs from those of the project demand, the guide should be immediately notified and whatever steps necessary to rectify this problem should be taken. Preferably a meeting should be held between the development team and the guide to discuss at length this issue.

2. Deviation From Time Line Chart

Mitigation

The cost associated with a late delivery is critical. A late delivery will result in receiving a failing grade for the course. Steps have been taken to ensure a timely delivery by gauging the scope of project based on the delivery deadline.

Monitoring

A schedule has been established to monitor project status. Falling behind schedule would indicate a potential for late delivery. The schedule will be followed closely during all development stages.

Management

Late delivery would be a catastrophic failure in the project development. If the project cannot be delivered on time the development team will not pass the course. If it becomes apparent that the project will not be completed on time, the only course of action available would be to request an extension to the deadline.

3. Poor Comments in Code

Mitigation

Poor code commenting can be minimized if commenting standards are better expressed. While standards have been discussed informally, no formal standard yet exists. A formal written standard must be established to ensure quality of comments in all code.

Monitoring

Reviews of code, with special attention given to comments will determine if they are up to standard. This must be done frequently enough to control comment quality. If they are not done comment quality could drop, resulting in code that is difficult to maintain and update.

Management

Should code comment quality begin to drop, time must be made available to bring comments up to standard. Careful monitoring will minimize the impact of poor commenting. Any problems are resolved by adding and refining comments as necessary.

4.3. Database Schema

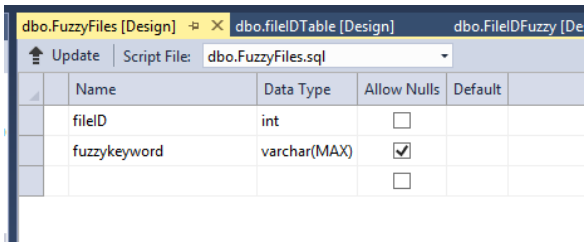


Fig. 4.2. database schema for table FuzzyFiles

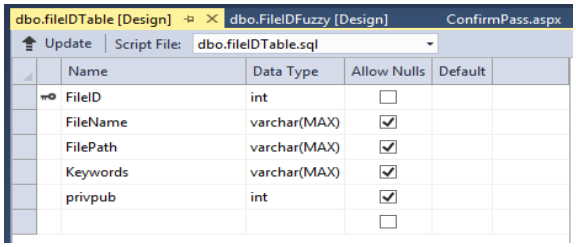


Fig. 4.3. database schema for table fileIDTable

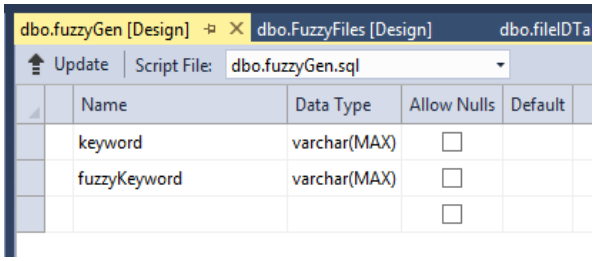


Fig. 4.4. database schema for table FuzzyGen

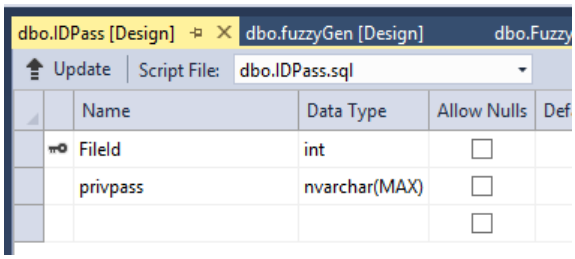


Fig. 4.5. database schema for table IDPass

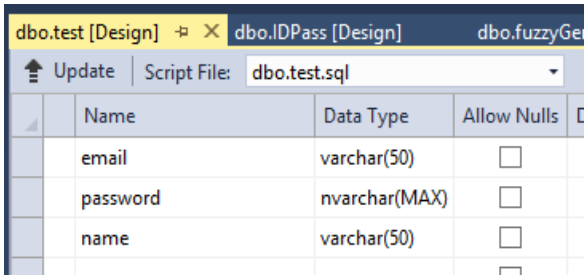


Fig. 4.6. database schema for table test

4.4. Sequence diagram

The sequence diagram for the project is given below:

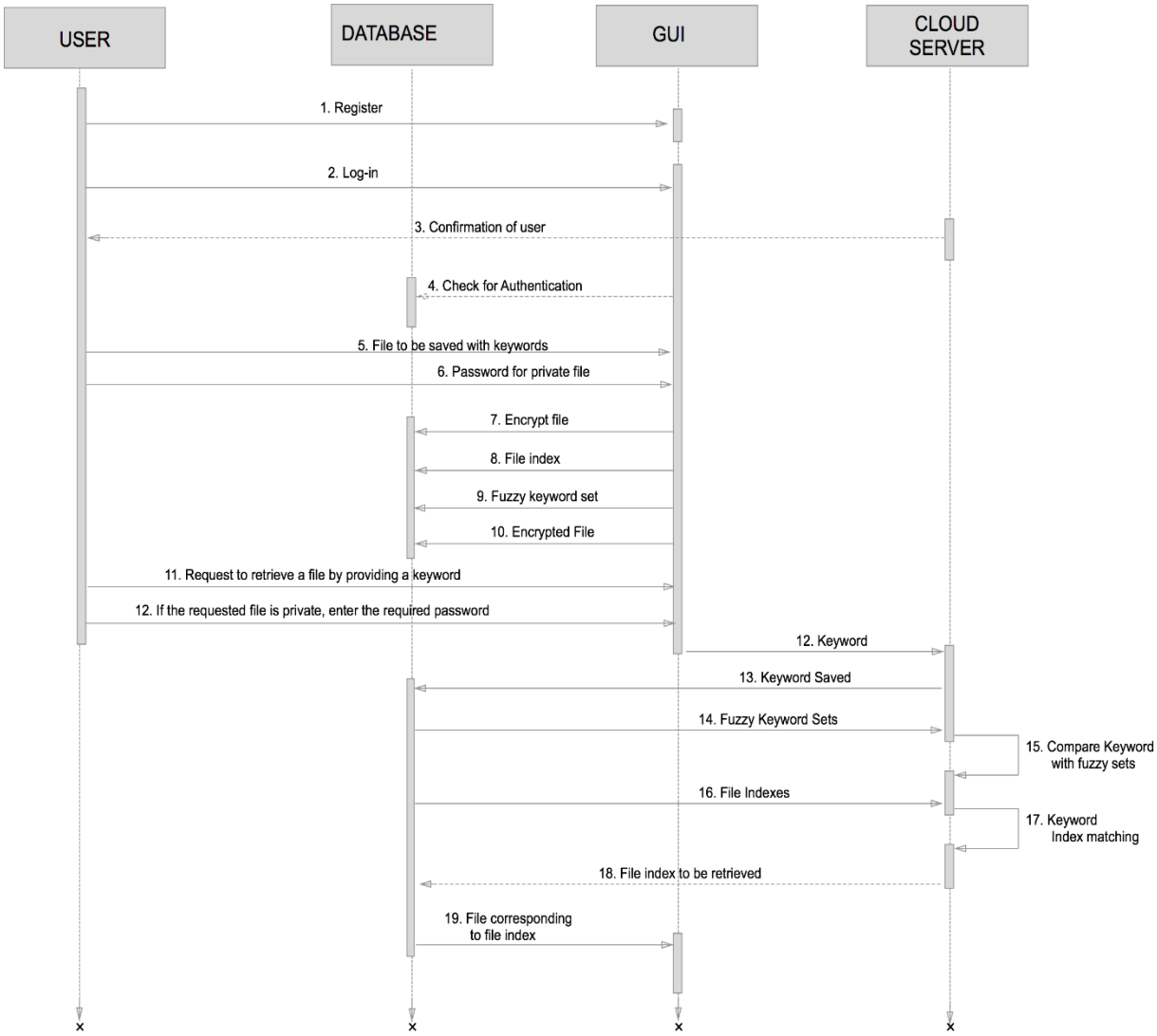


Fig 4.7. Sequence diagram

The user will first register and use the username and password to log into the system. The user will search a keyword for which a table will be created containing the fuzzy keywords matching to the searched keyword. Fuzzy keyword sets for the files saved will be made already by the data owner. The fuzzy set of the searched keyword will be matched with the fuzzy set of the existing files and the matching ones will be returned to the user.

4.5. Component diagram

The component diagram for the project is given below:

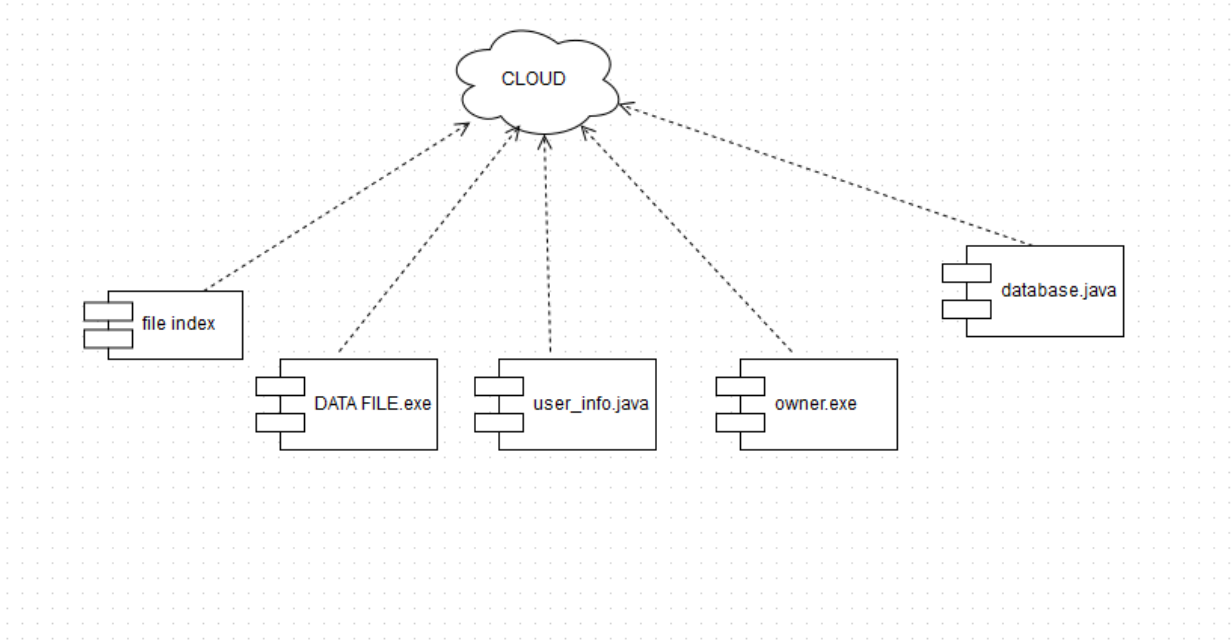


Fig 4.8. Component diagram

Our project includes the following files:

- File index to store the index of the files.
- Data file that will contain the data of the users.
- user_info contain the username and passwords of the authenticated users.
- Database and Owner.

4.6. Deployment diagram

The deployment diagram for the project is given below:

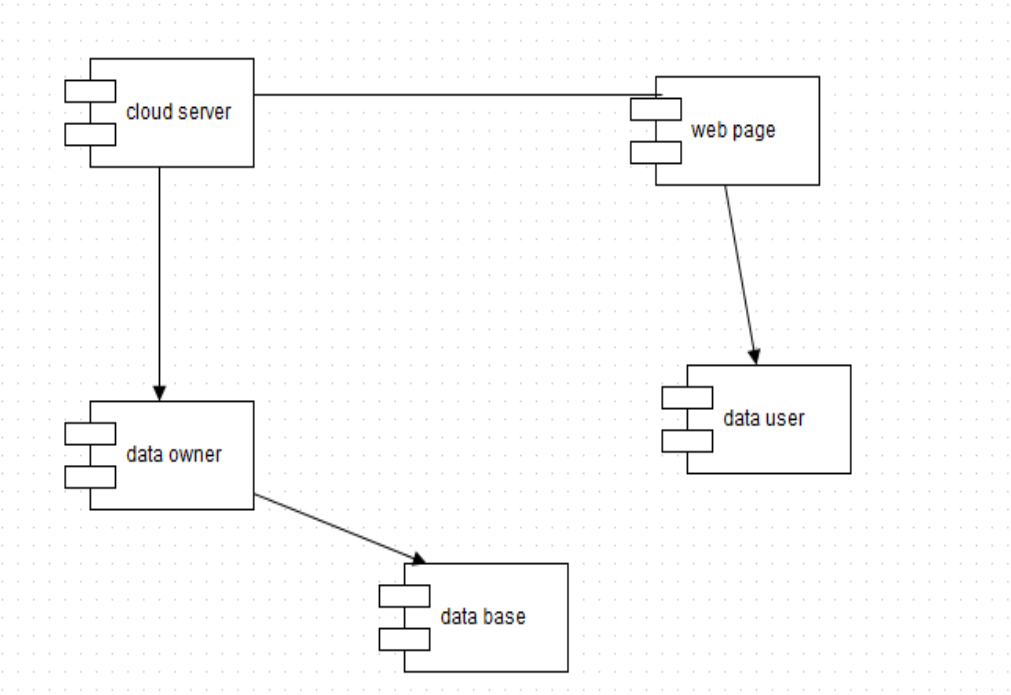


Fig 4.9. Deployment diagram

Our project uses cloud server where the files are stored, database, data owner which manages the file indexing, data user and web page through which the user communicates with the system.

4.7. Data Flow Diagram

The user will use the system for registration, login, uploading a file or searching a file. The username, password, email-id of the user are stored in the test table. The password is stored after being hashed in the table. The generated fuzzy keywords are stored along with the file-id in the FuzzyFiles table. The keywords of the file and their corresponding keywords are stored in the FuzzyGen table. The file path, keywords entered at the time of upload, file-id are stored in the FieldTable. If the file uploaded is kept as private then password must be provided by the user to access it at the time of downloading, this password will be stored in the hashed form with the corresponding file id.

The data flow diagram for the project is given below:

Fig 4.10. DFD Level 0

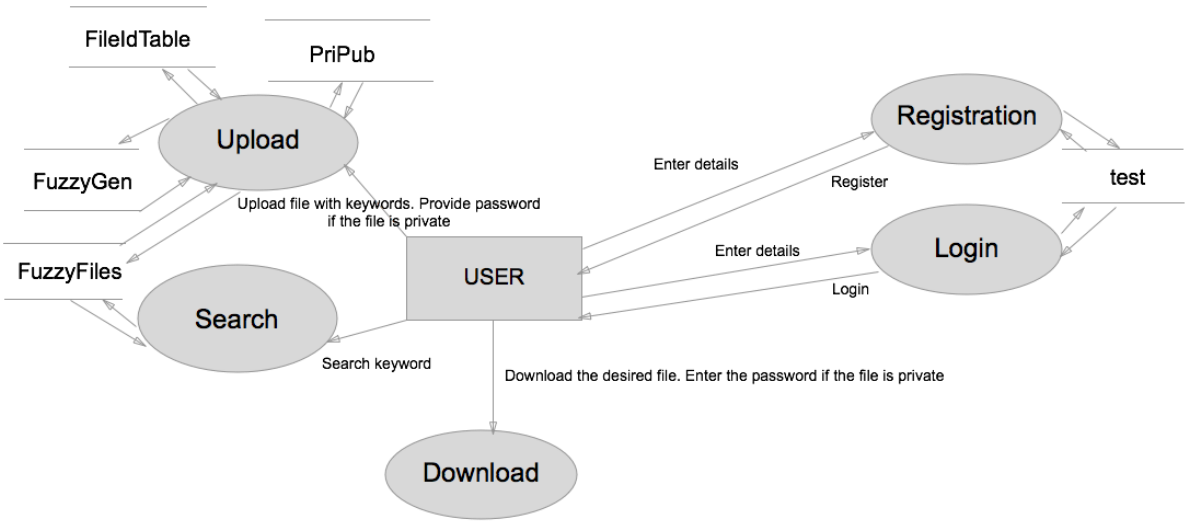


Fig 4.11..DFD Level 1

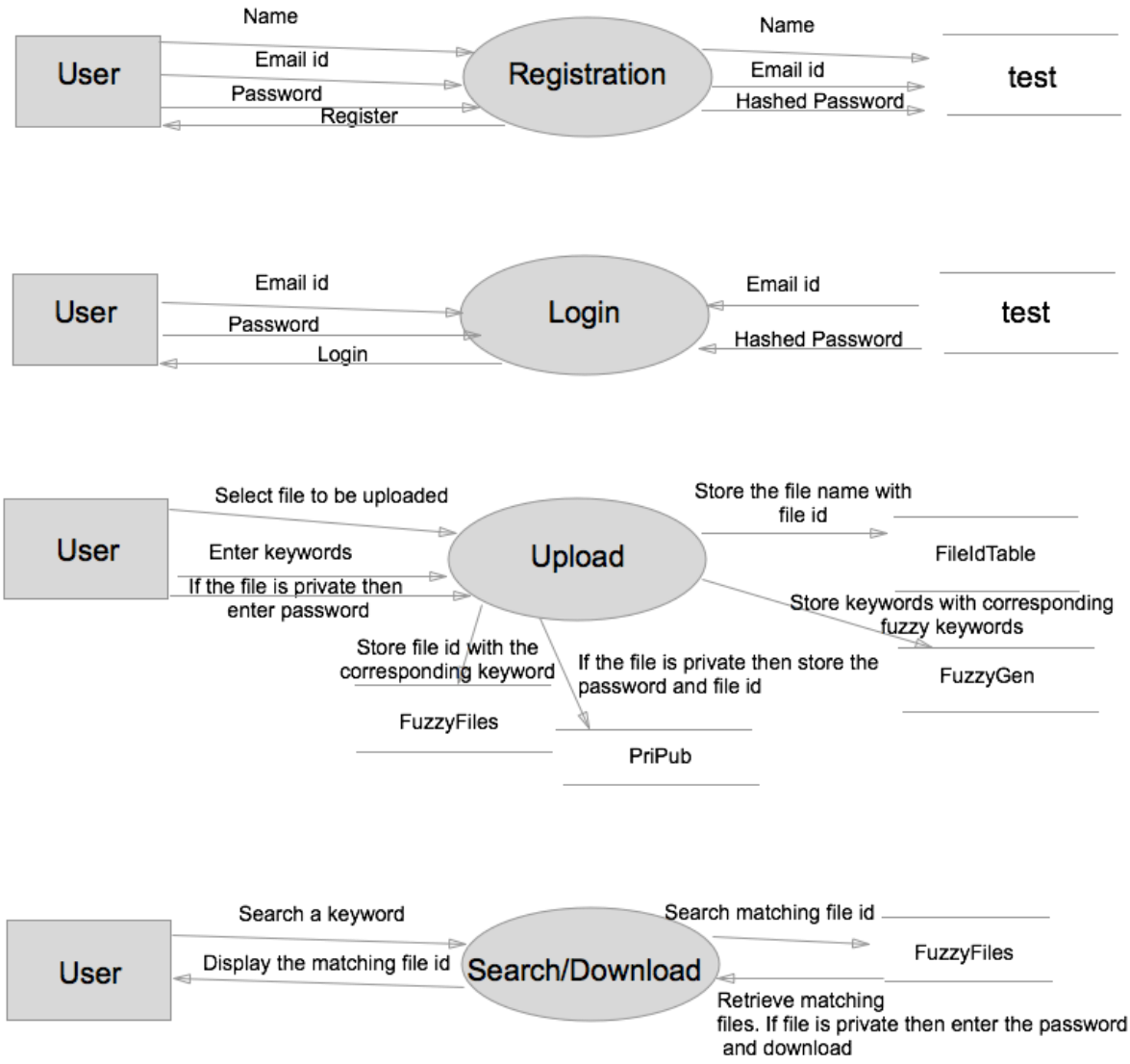


Fig 4.12. DFD Level 2

4.9. UseCase Diagram

The user registers on the system by entering his/her username, email-id, password. For accessing the system, the user needs to login by entering his/her email-id and password. After logging in the user can upload, search for a file. For uploading a file, the user needs to upload a file along with keywords that can be used for searching the file later. For searching a file, the user needs to enter a keyword and the files with the exact match or near match are returned. The user can download the required file by clicking on the download option.

The use case diagram for the project is given below:

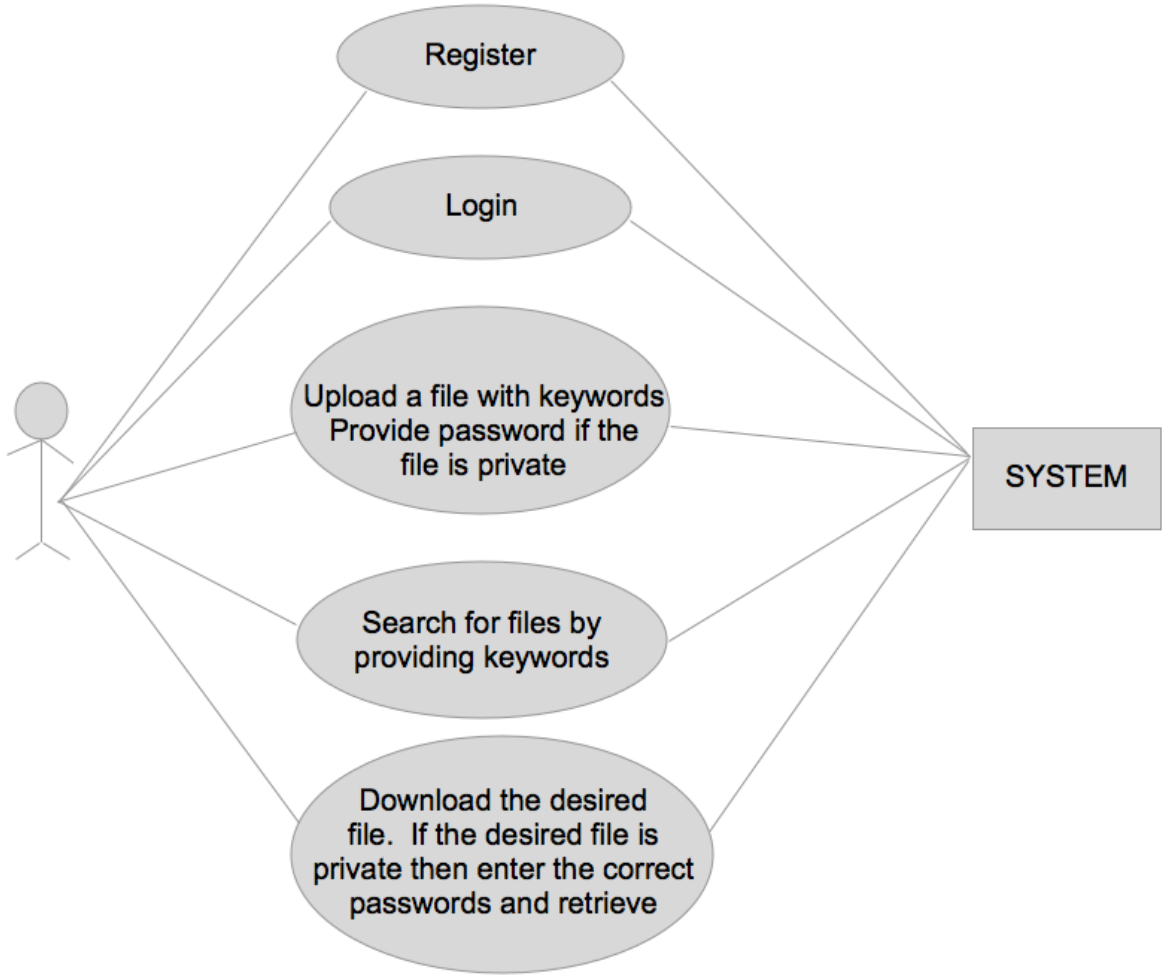


Fig 4.13. UseCase Diagram

4.10. Entity-Relationship Diagram

The username, email-id and password of users are stored in the database where email-id is the primary key. The user can upload a file. The file path, corresponding keywords and file-id of the files are stored in the database where file-id acts as the primary key. Fuzzy keywords of the entered keywords with the file are generated and stored in the database with the file-id. When the user searches for the keyword, the searched keyword is matched with the generated fuzzy keywords of the keywords entered at the time of uploading a file and the matching file is returned.

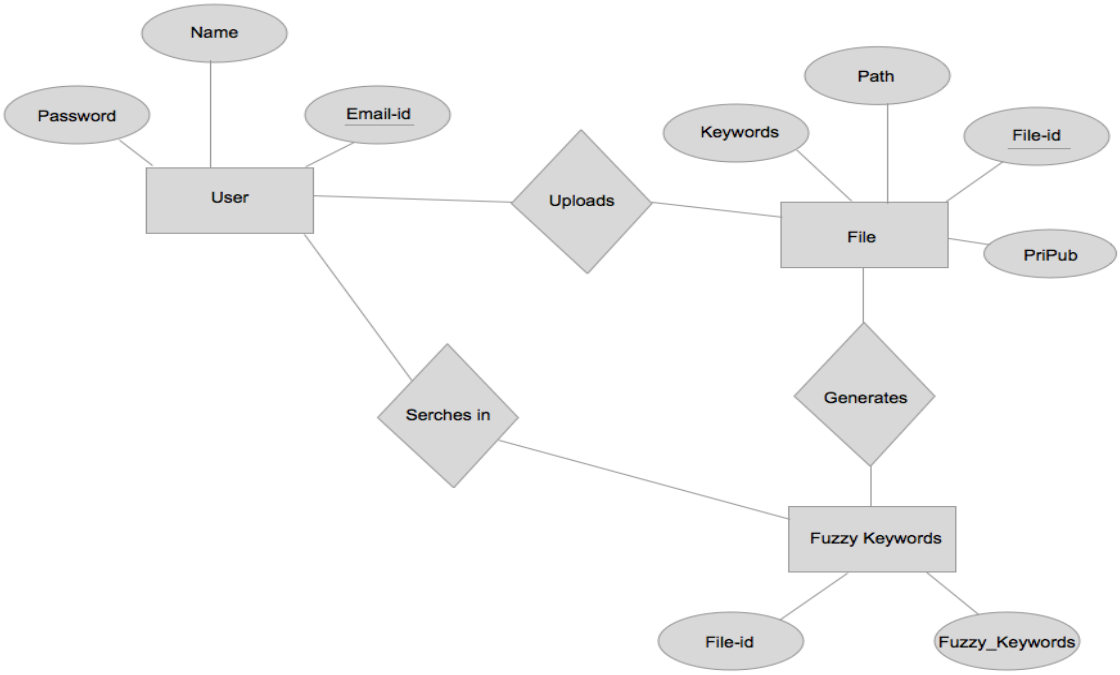


Fig 4.14. ER Diagram

CHAPTER 5

IMPLEMENTATION

5.1. AES Encryption

For authentication Encryption is needed while uploading the file by user. So for encryption we used AES algorithm of 128 key size. When user will upload the file, file will be stored in the encrypted format.

Function call for AES

1. create array encryptedBytes and assign null to it
2. create array saltBytes of size 8 bytes and assign 1 to 8 numbers
3. Initializes a new instance of the MemoryStream class as ms
    3.1 Initializes a new instance of the RijndaelManaged class as AES.
        3.1.1 KeySize = 256
            BlockSize = 128
        3.1.2 AES.Key = key.GetBytes(AES.KeySize / 8);
        3.1.3 AES.IV = key.GetBytes(AES.BlockSize / 8);
        3.1.3 select AES encryption mode as CBC
        3.1.4 Initializes a new instance of the CryptoStream class with a ms,
            AES.CreateEncryptor(), CryptoStreamMode.Write.
            3.1.4.1 cs.Write(bytesToBeEncrypted, 0, bytesToBeEncrypted.Length);
            3.1.4.2 cs.Close();
        3.1.5 convertms list to array and assign it to encryptedBytes
4. returnencryptedBytes

5.2. Upload

For storing the file onto the server first it need to be selected and then upload it. Uploading all the files means storing them on cloud. Whoever is uploading the file he must be authenticated. He must provide all the credentials.

function call wild with string p and int function

1. assign value of p to new string a
2. initializei,j,k=0,l,m,v
3. create char array b and assign a to it
4. l = length of a
5. v = (2 * l + 2) * l + (2 * l + 1)
6. create char array str with length of l*(l+1)+l+10
7. create 2D array s of length (l+1,l)
8. fori=0 to l+1
    - 8.1 k-0
    - 8.2 for j=0 to l
    - 8.2.1 s[i, j] = b[k++]
9. fori=0 to l
    - 9.1 for j=0 to l
        - 9.1.1 if(i==j)
            - 9.1.1.1 s[i, j] = '*'
10. k=0
11. fori=0 to l+1
    - 11.1 for j=0 to l
        - 11.1.1 str[k] = s[i, j]
        - 11.1.2 k++
    - 11.2 str[k] = ' '
    - 11.3 k++
12. create new string stri
13. splitstri by '\0'
14. query for inserting keyword,fuzzyKeyword into table fuzzyGen
15. trim string stri
16. split string with ' ' and assign to new string w
17. for z=0 to w.length
    - 17.1 String s1 = w[z]
    - 17.2 query for inserting fileID,fuzzykeyword into table FuzzyFiles

function for encryption

function for btnUpload_Click

1. String s="x"
2. int x
3. selecting maximum fileID from table fileIDTable
4. if (s == "NULL")
    4.1 x = 101
5. else
    5.1 x++
6. String s1, s2, s3, s4, s5
7. s1 = TextBox1.Text.ToString()
8. s2 = TextBox2.Text.ToString()
9. s3 = TextBox3.Text.ToString()
10. s4 = TextBox4.Text.ToString()
11. s5 = TextBox5.Text.ToString()
12. if (s1==" ")
    12.1 s1=" "
12. if (s2==" ")
    12.1 s2=" "
12. if (s3==" ")
    12.1 s3=" "
12. if (s4=" ")
    12.1 s4=" "
12. if (s5==" ")
    12.1 s5=" "
13. query for inserting FileID,FileName,FilePath,Keywords into table fileIDTable
14. cmd.Parameters.AddWithValue("@ID", x);
15. cmd.Parameters.AddWithValue("@Name", file);
16. cmd.Parameters.AddWithValue("@Path", "/Files/" + file);
17. cmd.Parameters.AddWithValue("@keyword", s1 + " " + s2 + " " + s3 + " " + s4 + " " + s5)
18. string filename = Server.MapPath("/Cloud_Fuzzy1/Files/" + file)
19. string password = "abcd1234"
20. byte[] bytesToBeEncrypted = File.ReadAllBytes(filename);
21. byte[] passwordBytes = Encoding.UTF8.GetBytes(password)
22. passwordBytes = SHA256.Create().ComputeHash(passwordBytes)

```
23. byte[] bytesEncrypted = AES_Encrypt(bytesToBeEncrypted, passwordBytes);
24. stringfileEncrypted = Server.MapPath("/Cloud_Fuzzy1/Files/" + file);
25. File.WriteAllBytes(fileEncrypted, bytesEncrypted)
26. s1 = TextBox1.Text.ToString()
27. s2 = TextBox2.Text.ToString()
28. s3 = TextBox3.Text.ToString()
29. s4 = TextBox4.Text.ToString
30. s5 = TextBox5.Text.ToString()
31. char[] z = s5.ToCharArray()
32. int j = s5.Length
33. if(s1!= "")
        33.1 wild(s1,x)
34. if(s2!= "")
        34.1 wild(s2,x)
35. if(s3!= "")
        35.1 wild(s3,x)
36. if(s4!= "")
        36.1 wild(s4,x)
37. if(s5!= "")
        37.1 wild(s5,x)
```

5.3. Search Algo

When user wants to search any file he must be authorized. He can perform searching on any file system. While searching he can do any typing errors. So for getting the right result matching need to be done. Keywords that are entered by users will be matched with fuzzy keywords generated by wild card based technique.

```
1. public partial class users_Search : System.Web.UI.Page
        1.1 constring= ConfigurationManager.ConnectionStrings ["ConnectionString"].Con-
        nectionString;
        1.2 public static SqlConnection con = new SqlConnection(constring);
        1.3 List<int>termsList = new List<int>();
        1.4 protected void Page_Load(object sender, EventArgs e)
                1.4.1 if (!IsPostBack)
```

1.4.1.1 if (Session["StudentUser"] == null)

1.4.1.1.1 Response.Redirect("~/Login.aspx");

1.4.1.2 else

1.4.1.2.1 Response.ClearHeaders();

` 1.4.1.2.2 Response.AddHeader("Cache-Control", "no-cache, no-store, max-age=0, must-revalidate");

1.4.1.2.3 Response.AddHeader("Pragma", "no-cache");

1.5 protected void ButtonClick(object sender, EventArgs e)

1.5.1 int[] x = new int[150];

1.5.2 Boolean flag;

1.5.3 String inp, variable;

1.5.4 String[] fuzz = new String[1000];

1.5.5 String[] colltrue = new String[1000];

1.5.6 inti=0, j,k=0,n,p=0;

1.5.7 inp = TextBox1.Text.ToString();

1.5.8 SqlCommand command = new SqlCommand("select fuzzykeyword from FuzzyFiles", con);

1.5.9 SqlDataReadermyreader;

1.5.10 myreader = command.ExecuteReader();

1.5.10.1 k = 0;

1.5.11 Response.Write(" ");

1.5.12 while (myreader.Read())

1.5.12.1 variable = myreader.GetString(0);

1.5.12.2 fuzz = variable.Split(' ');

1.5.12.3 j = inp.Length;

1.5.12.4 foreach (string words in fuzz)

1.5.12.4.1 flag = stringmatch(words, inp);

1.5.12.4.2 if (flag == true)

1.5.12.4.2.1 colltrue[k] = words;

1.5.12.4.2.2 k++;

1.5.13 p = 0;

1.5.14 foreach (String collword in colltrue)

1.5.14.1 con.Open();

1.5.14.2 String s1 = "select distinct (fileID) from FuzzyFiles where fuzzykeyword = ';

1.5.14.3 s1 = string.Concat(s1, "'");

1.5.14.4 s1 = string.Concat(s1, collword);

1.5.14.5 s1 = string.Concat(s1, "'");

1.5.14.6 SqlCommand cmd1 = new SqlCommand(s1, con);

1.5.14.6 SqlDataReader myreader1;

1.5.14.7 myreader1 = cmd1.ExecuteReader();

1.5.14.8 while (myreader1.Read())

1.5.14.8.1 int z = myreader1.GetInt32(0);

1.5.14.8.2 if (!x.Contains(z))

1.5.14.8.2.1 x[p] = z;

1.5.14.8.2.2 p++;

1.5.14.8.2.3 print myreader1.GetInt32(0)

1.5.14.9 con.Close();

1.5.15 int[] terms = termsList.ToArray();

1.5.16 Session["Item"] = termsList;

1.5.17 Response.Redirect("Result.aspx");

1.5.18 Response.Write(" ");

1.6 public Boolean stringmatch(string words, string inp)

1.6.1 inti;

1.6.2 Boolean flag = true;

1.6.3 if (words.Length<= (inp.Length + 1) && words != null && words != "NULL" && words != " ")

1.6.3.1 if (words.Length == inp.Length)

1.6.3.1.1 for (i = 0; i<inp.Length; i++)

1.6.3.1.1.1 if (words[i] == '*')

1.6.3.1.1.1.1 continue;

1.6.3.1.1.2 else

1.6.3.1.1.2.1 if (words[i] != inp[i])

1.6.3.1.1.2.1.1 flag = false;

1.6.3.1.1.2.1.2 break;

1.6.3.1.2 return flag;

1.6.3.2 intfindex = 0, uindex = 0;

1.6.3.3 for (i = 0; i<words.Length; i++)

1.6.3.3.1 if (words[findex] == '*')

1.6.3.3.1.1 findex++;

1.6.3.3.1.2 tinue;

1.6.3.3.2 else

1.6.3.3.2.1 if (words[findex] != inp[uindex])

1.6.3.3.2.1.1 flag = false;

1.6.3.3.2.1.2 break;

1.6.3.3.3 findex++;

1.6.3.3.3.1 uindex++;

1.6.3.4 return flag;

1.6.4 return false;

1.7 public void printRes(int res)

1.7.1 termsList.Add(res);

CHAPTER 6

RESULT

6.1. Snapshots

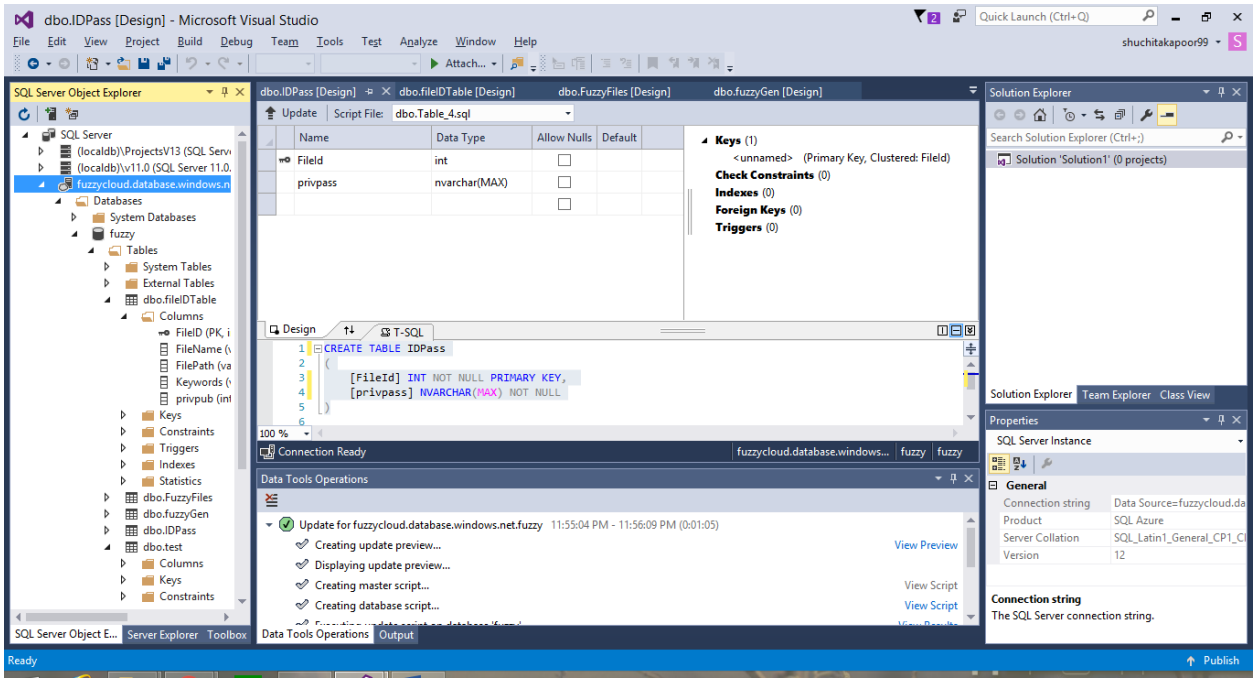


Fig 6.1 Tables added to the Azure database

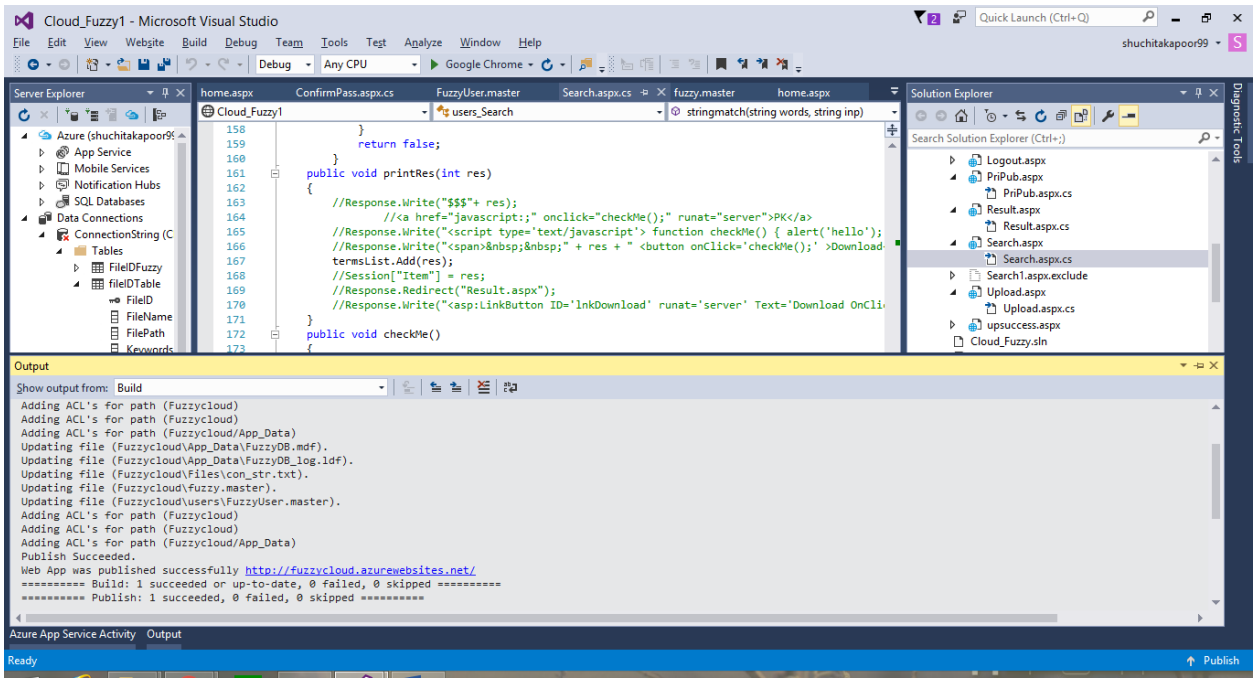


Fig 6.2. Publishing website to Microsoft Azure.

Fig 6.3. Home page of the website

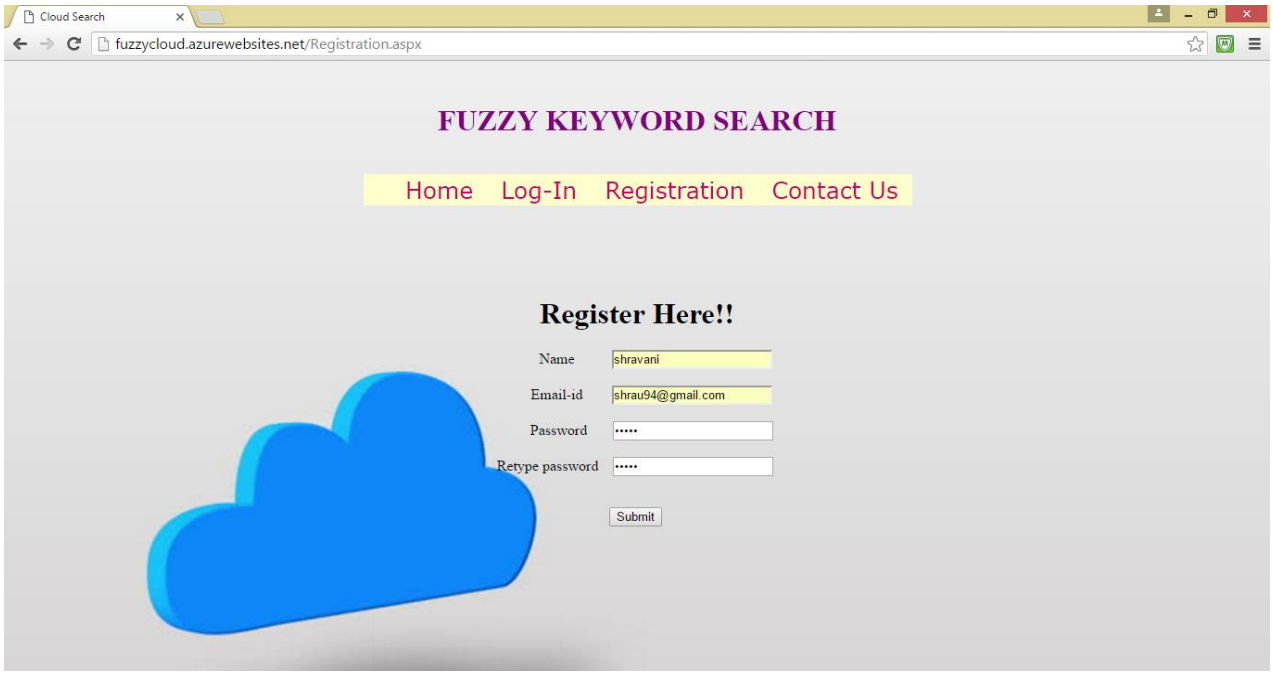


Fig 6.4. Registration page of the website. Here the new user registers by providing name, email-id and password. A user must login in order to upload and search a file.

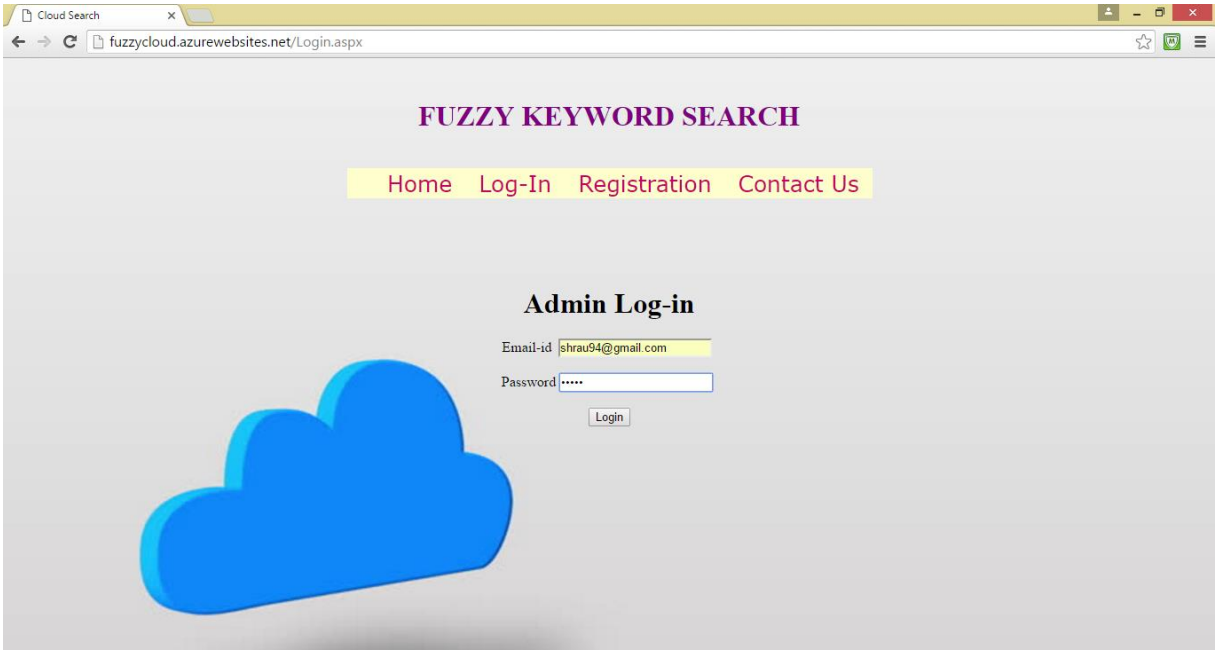


Fig 6.5. Login page.

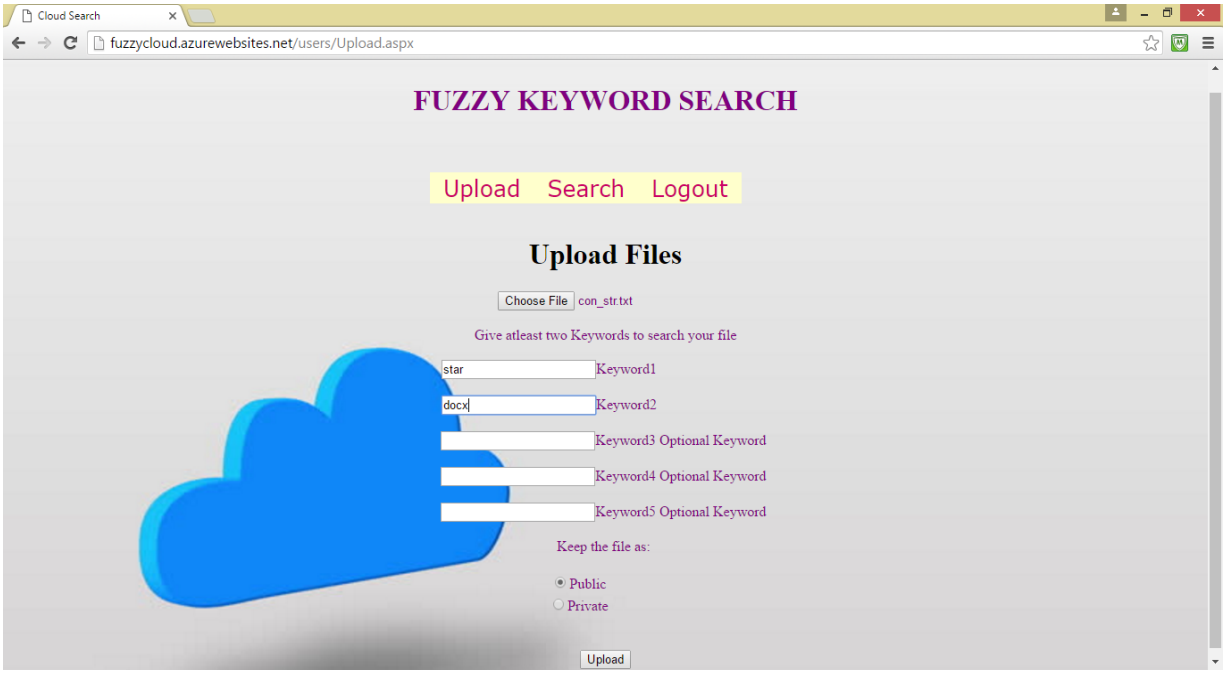


Fig 6.6. Upload page where the user can upload website with the required keywords and also specify the access. Here the access is specified as public.

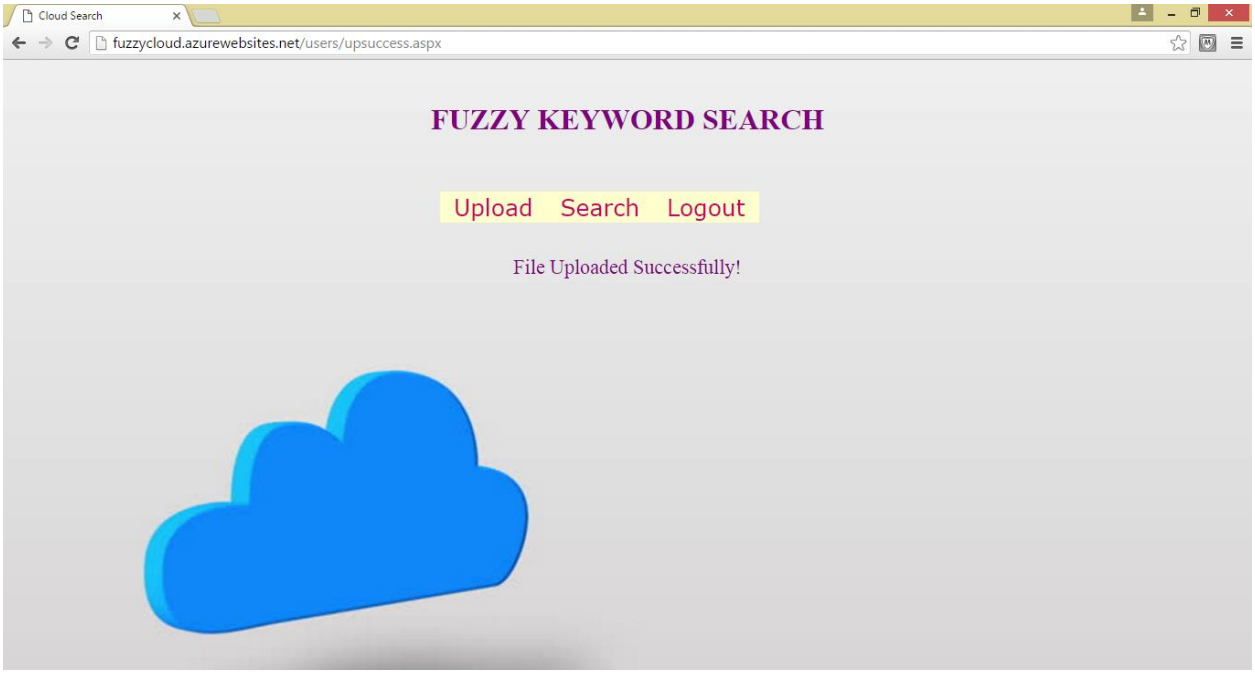


Fig 6.7. Successful upload of the file

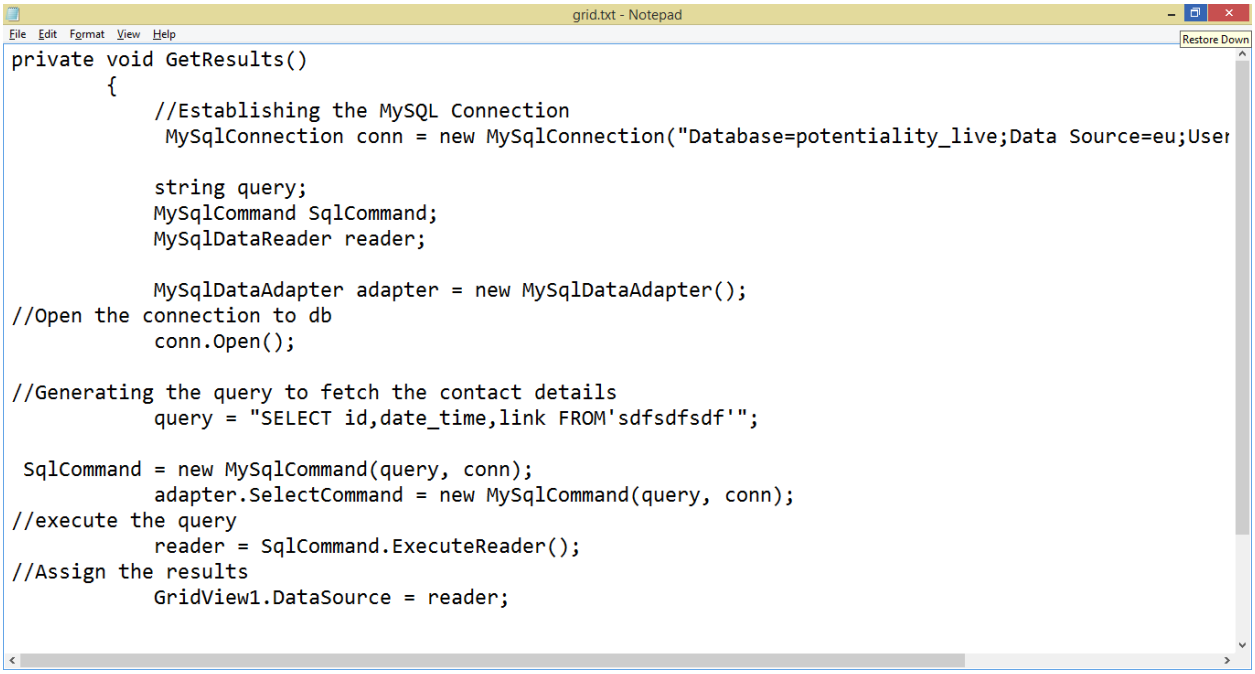


Fig 6.8. grid.txt before encryption

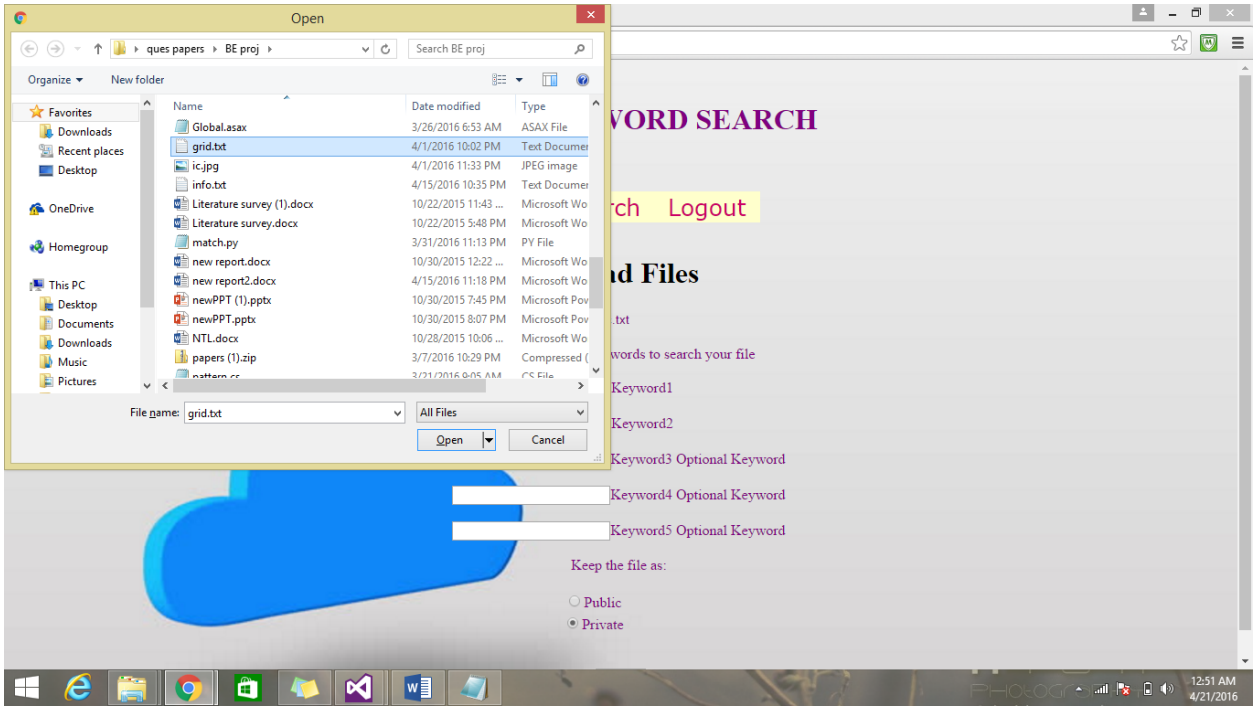


Fig 6.9. A file name grid.txt is uploaded

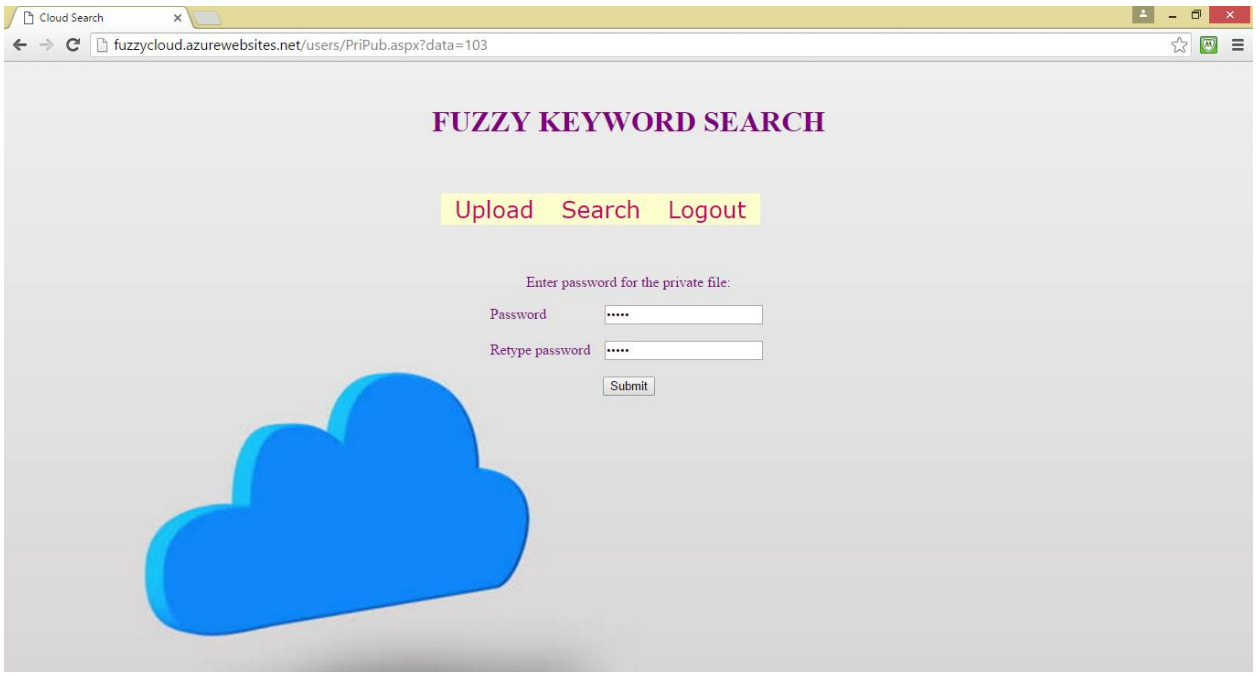


Fig 6.10. If the file type is kept private then the user needs to provide password for accessing later

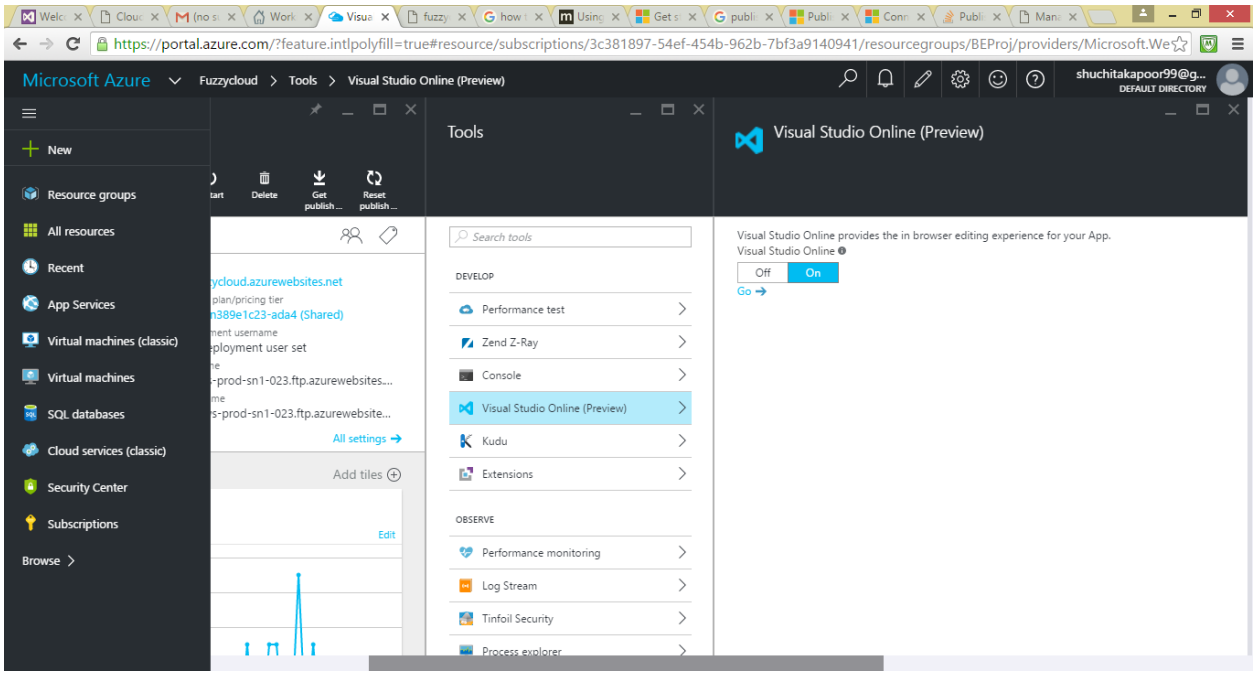


Fig 6.11. Opening the website on Azure.

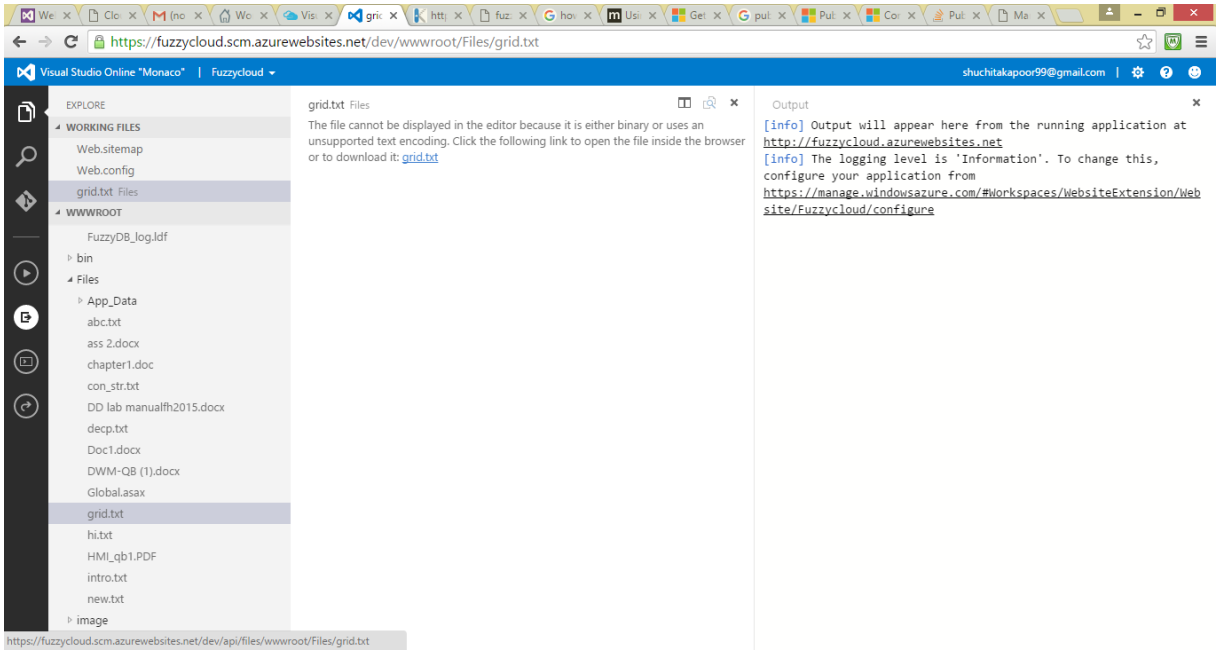


Fig 6.12. Opening the uploaded file on Azure.

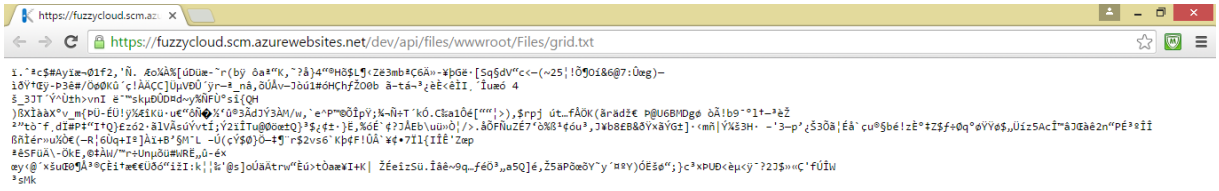


Fig 6.13. Uploaded file is stored after encryption

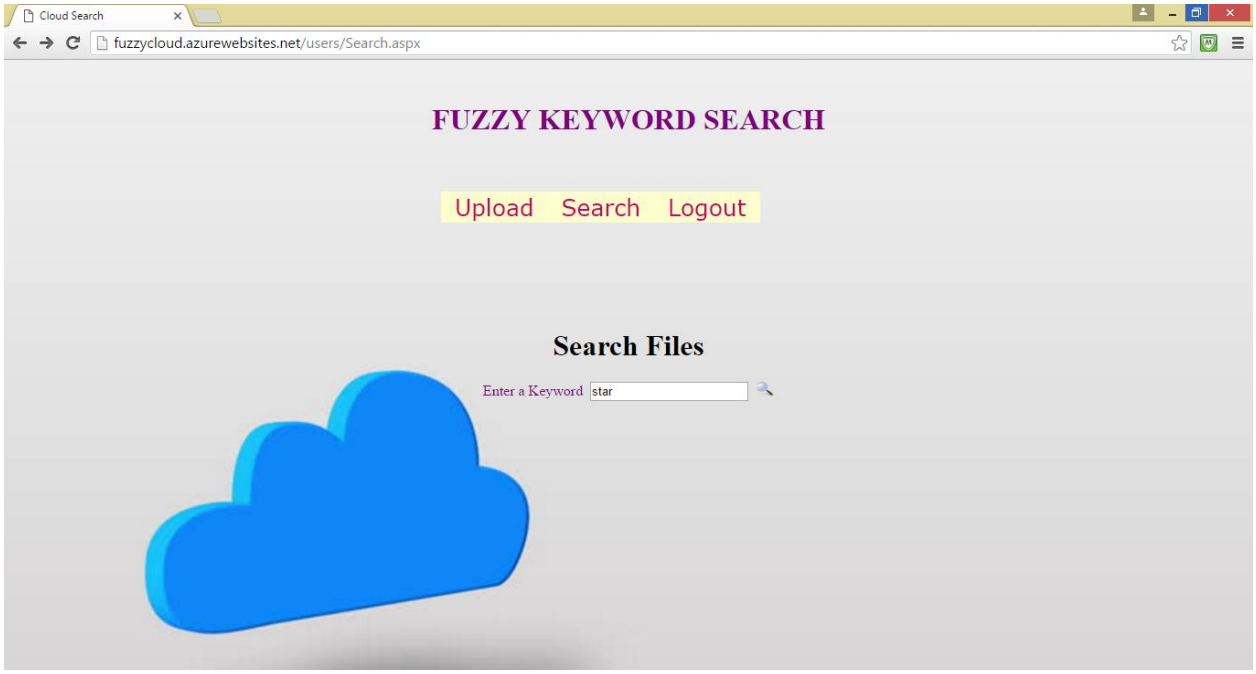


Fig 6.14. Search page where the user needs to enter keyword to search for file

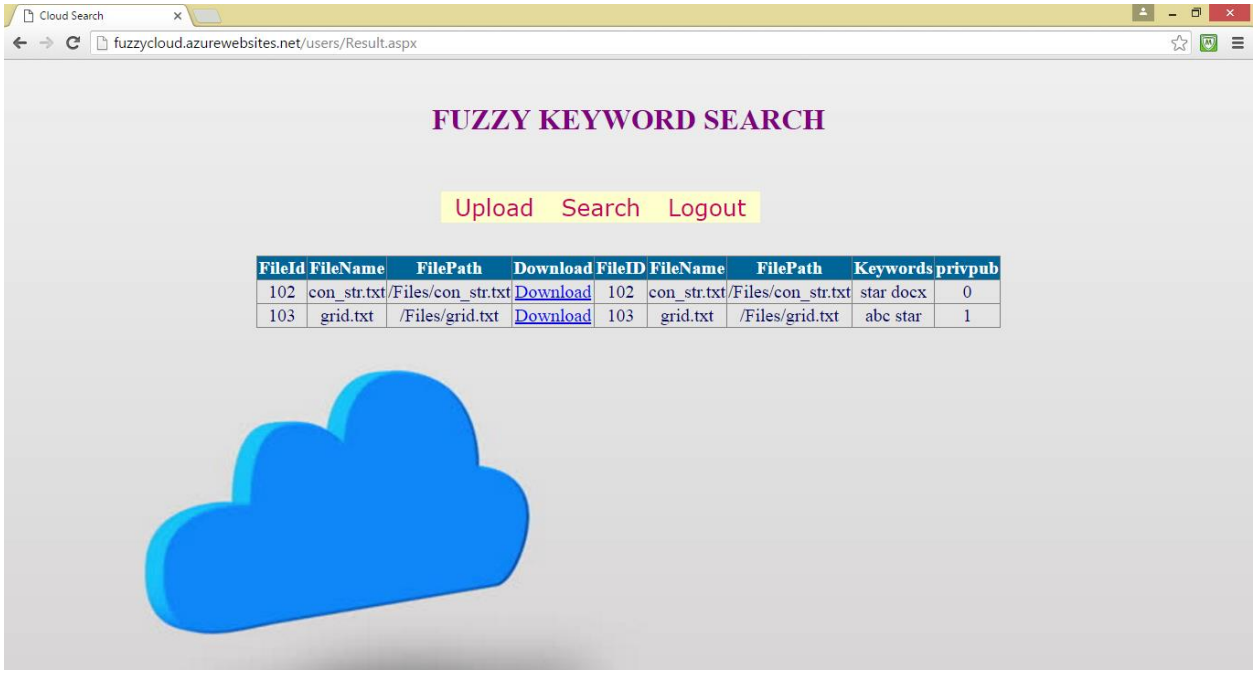


Fig 6.15. Result page where the matching files are displayed

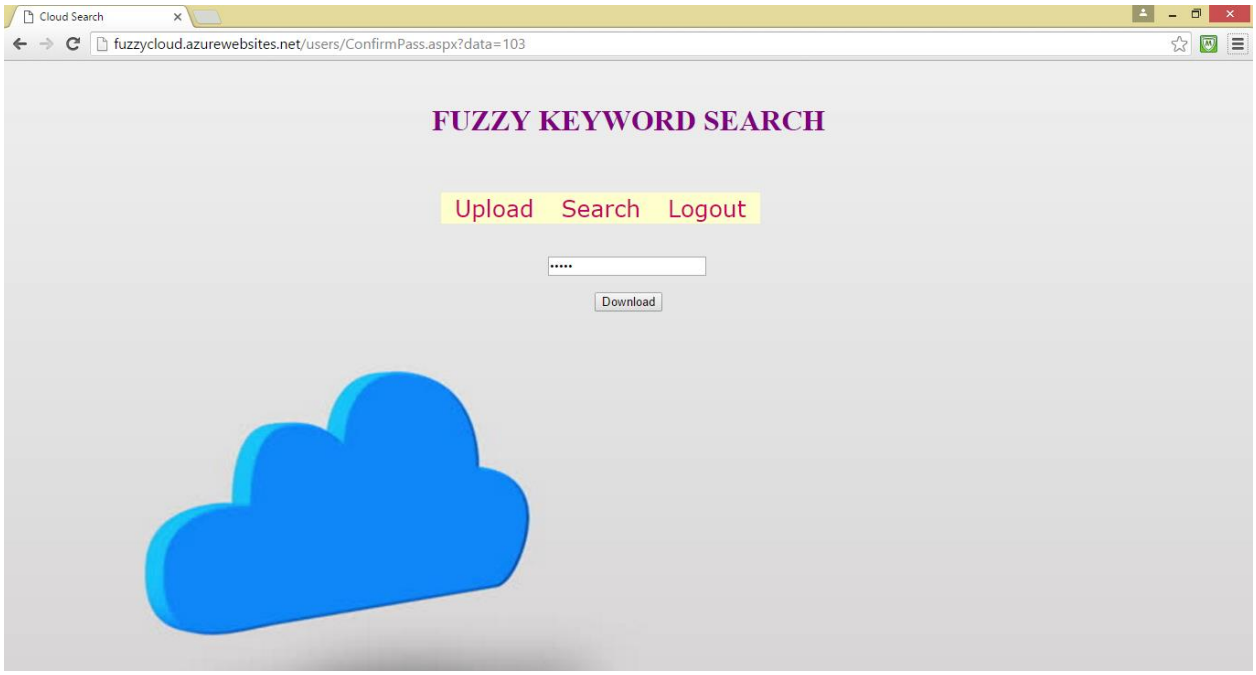


Fig 6.16. If the file is private the user needs to enter the correct password

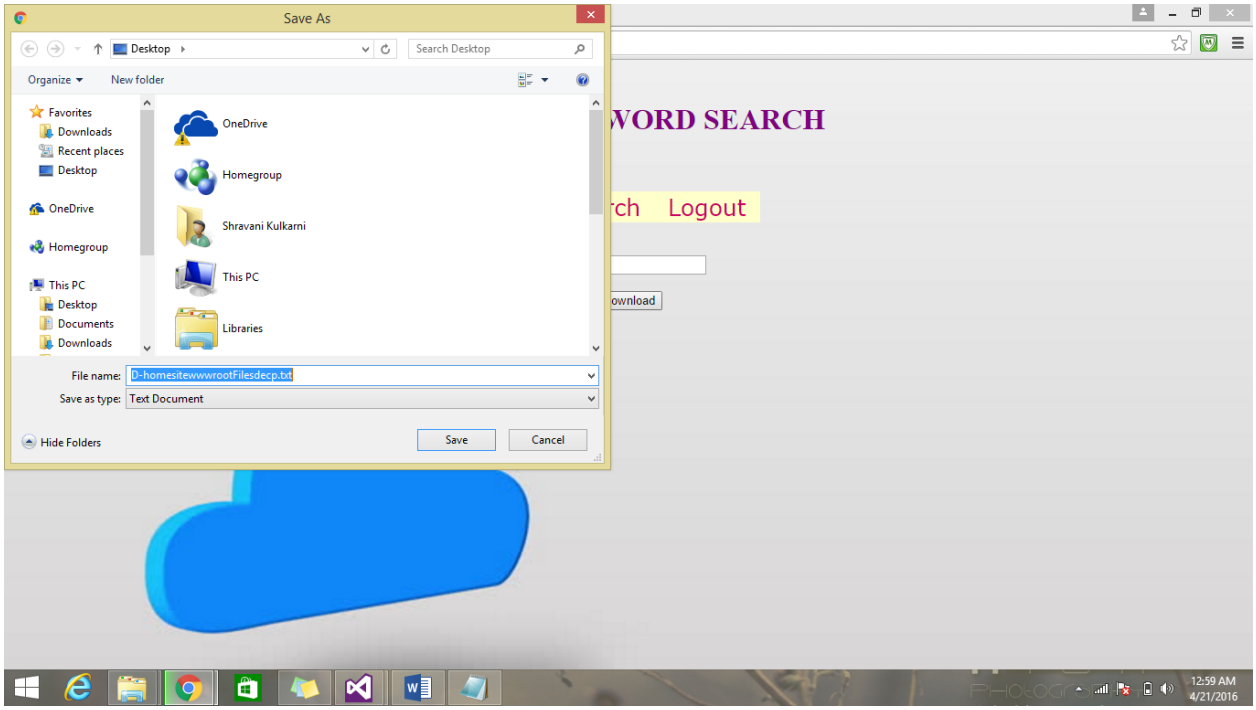


Fig 6.17. Window to save the file in the desired location

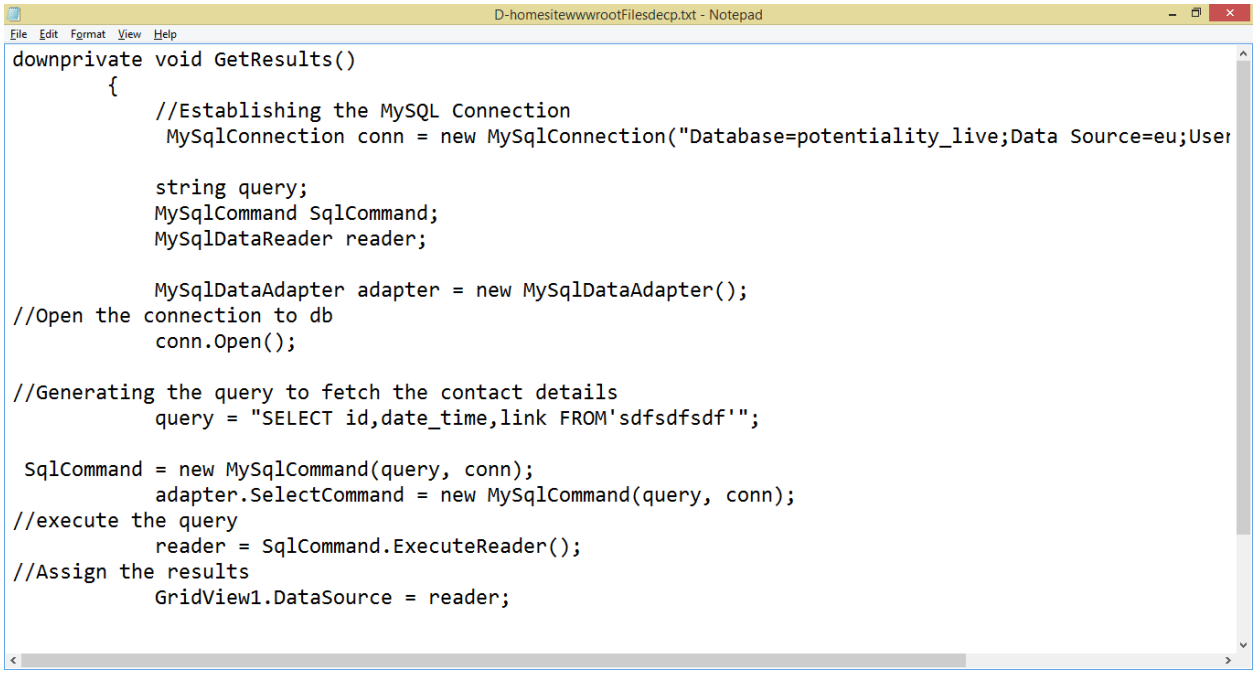


Fig 6.18. Downloaded file

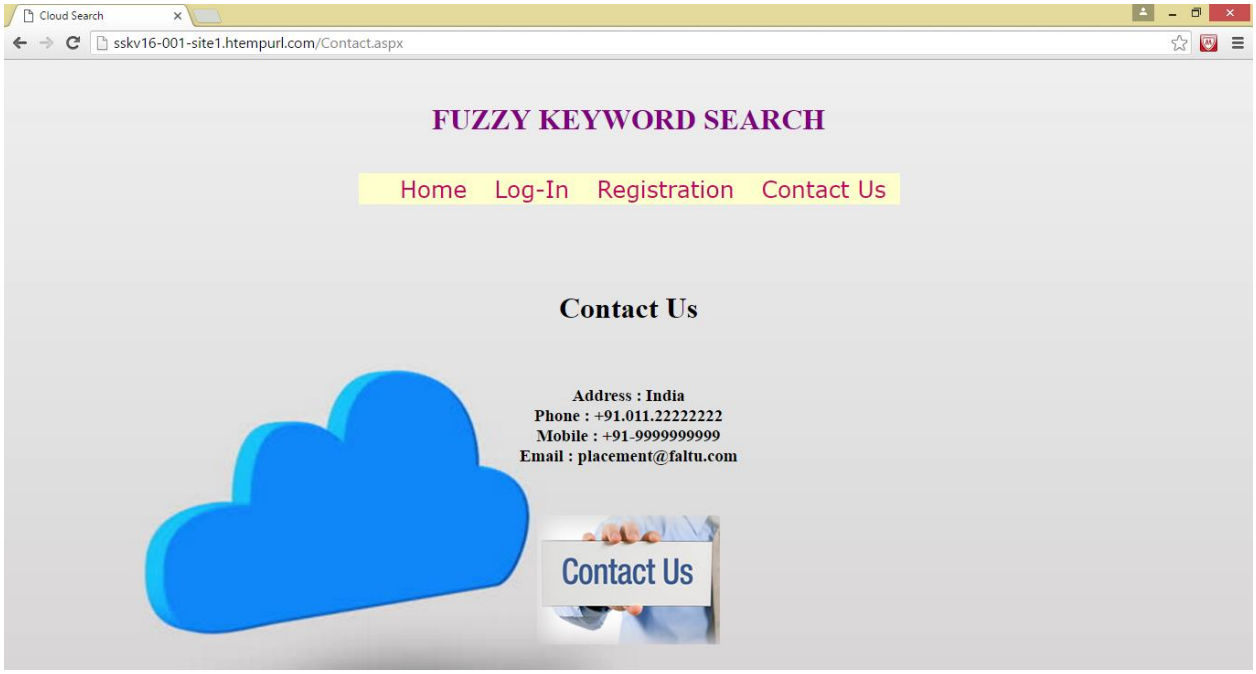


Fig 6.19. Contact page of the website

CHAPTER 7

CONCLUSION

We made a privacy-preserving fuzzy search for achieving effective usage of remotely stored encrypted data in cloud computing. We designed an advanced search mechanism (i.e. Wild-card Based Technique) for constructing storage efficient fuzzy-keyword sets based on the similarity metric edit distance. Based on the fuzzy-keyword sets, we proposed a fuzzy key-word search technique. The fuzzy keyword search returns the results according to the follow-ing rules:

- If the input exactly matches the pre-set keyword, the server should return the files containing the keyword.
- If typos and/or format inconsistencies exist in the searching input, the server returns the closest possible results based on pre-specified similarity semantics.

The files are uploaded as private or public. Public means that it will be accessed by any user who logs into the system. And private means that it will require the user to specify a password during uploading. This password will be hashed and stored into the database. It will be re-quired while downloading the private file by the user.

References

[1] Dr Narendra kekokar, “Implementation of Fuzzy Keyword Search over Encrypted Data in Cloud Computing” in ICATA, 2015 .

[2] Jawahar Thakur and Nagesh Kumar, “DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis” in ISSN 2250-2459, Volume 1, Issue 2, December 2011.

[3] Jeff Naruchitparames, “Enhancing Data Privacy and Integrity on Cloud”.

[4] “Fuzzy Keyword Search over Encrypted Data in Cloud Computing” by Jin Li and Quan Wang in Mini-Conference at IEEE INFOCOM 2010.

[5] “Fuzzy Keyword Search over Encrypted Data in Cloud Computing” by C Anuradha.

[6] “A survey on Data Retrieval techniques in Cloud Computing” by S. Balasubramaniam in Journal of Convergence Information Technology, November 16, 2015

[7] Jin Li, Qian Wang and Cong Wang, “Enabling Efficient Fuzzy Keyword Search over Encrypted Data in Cloud Computing”.

[8] Ashly.k.Achenkunju., “Privacy and Security in Data Storage Using Two Layer Encryption and MAC Verification” in e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. VIII (Sep – Oct. 2014), PP 17-23 www.iosrjournals.org.

[9] Jayanand A. Kambleand Vaishali P. Barde, “A Persual Study of methods used for Efficient Search over Encrypted Data” in ISSN: 2349-2163 Issue 5, Volume 2 (May 2015).

[10] JoonsangBaek, ReihanehSafiavi-Naini and Willy Susilo, “Public Key Encryption with Keyword Search Revisited”

[11] Yanbin Lu and Gene Tsudik, “Enhancing Data Privacy in the Cloud”

[12] Narender Tyagi and Anita Ganpati, “International Journal of Advanced Research in Computer Science and Software Engineering” in ISSN: 2277 128X,Volume 4, Issue 8, August 2014

[13] RanjeetMasram, VivekShahare, Jibi Abraham and RajniMoona, “Analysis and Comparison of Symmetric Key Cryptographic Algorithms based on various file features” in International Journal of Network Security & Its Applications (IJNSA), Vol.6, No.4, July 2014

[14] Almas Ansari and Prof.ChetanBawankar, “Privacy & Data Integrity for Secure Cloud Storage” in International Conference on Advances in Engineering & Technology – 2014

[15] P. Mell and T. Grance, “Draft nist working definition of cloud computing,” Referenced on June. 3rd, 2009 Online at http://csrc.nist.gov/groups/SNS/cloud-computing/index.html, 2009.

[16] Preethi Mathew and Dr. S. Sasidhar Babu, "Secure Fuzzy Multi-Keyword Ranked Search over Encrypted Cloud Data" in An ISO 3297: 2007 Certified Organization, Vol. 3, Issue 8, August 2015

[17] P. Kalidas and R. Chandrasekaran, "Efficient Interactive Fuzzy Keyword Search Over Encrypted Data in Cloud Computing" in ISSN: 2277 128X, Volume 3, Issue 4, April 2013.

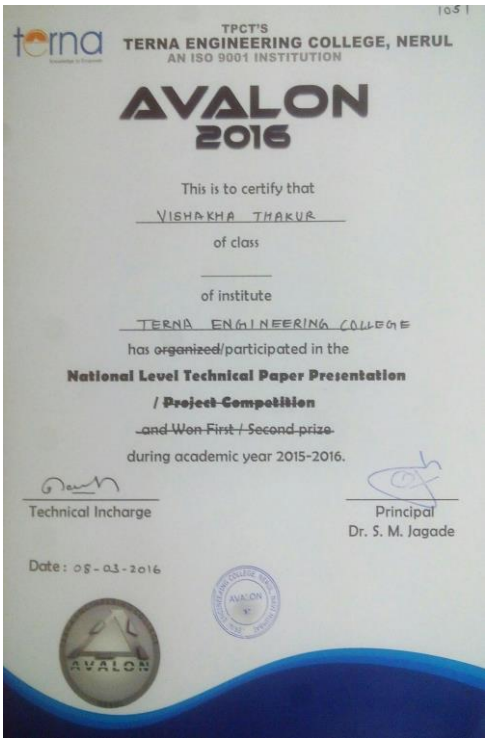
[18] Kailash Bhanushali, Neel Gala and NishaVanjari, "A Review Paper on Fuzzy Search over Encrypted Data in Cloud Computing" in ISSN: 2393-9842, Issue 3, Volume 2 (March 2015)

[19] Federal Information Processing Standards Publication 197, "Announcing the ADVANCED ENCRYPTION STANDARD (AES)" Referenced on November 26, 2001 Online at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

Publications

"COMPARATIVE STUDY ON KEYWORD SEARCHING TECHNIQUES OVER ENCRYPTED DATA IN CLOUD COMPUTING", Kshitija Jagtap, Shravani Kulkarni, Shuchita Kapoor, Vishakha Thakur, Rashmi Dhumal, International Journal of Technical Research and Application, Special Issue 41 (March 2016), Pg. No. 50-53.

Appendix

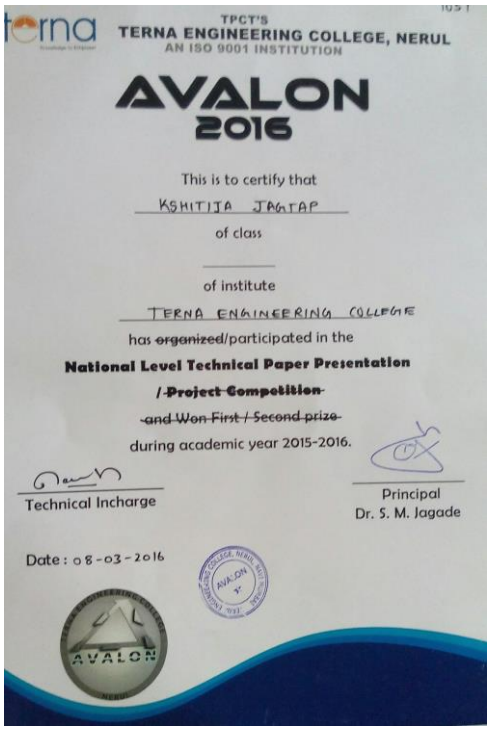

TPCT'S
TERNA ENGINEERING COLLEGE, NERUL
AN ISO 9001 INSTITUTION

AVALON 2016

This is to certify that
VISHAKHA THAKUR
of class
of institute
TERNA ENGINEERING COLLEGE
has ~~organized~~/participated in the
National Level Technical Paper Presentation
/ ~~Project Competition~~
~~and Won First / Second prize~~
during academic year 2015-2016.

Technical Incharge
Principal
Dr. S. M. Jagade

Date : 08-03-2016


TPCT'S
TERNA ENGINEERING COLLEGE, NERUL
AN ISO 9001 INSTITUTION

AVALON 2016

This is to certify that
KSHITIJA JAGTAP
of class
of institute
TERNA ENGINEERING COLLEGE
has ~~organized~~/participated in the
National Level Technical Paper Presentation
/ ~~Project Competition~~
~~and Won First / Second prize~~
during academic year 2015-2016.

Technical Incharge
Principal
Dr. S. M. Jagade

Date : 08-03-2016

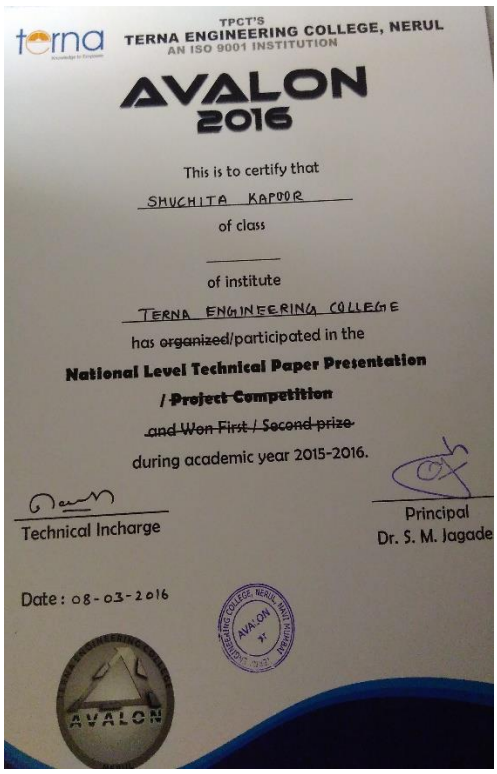

TPCT'S
TERNA ENGINEERING COLLEGE, NERUL
AN ISO 9001 INSTITUTION

AVALON 2016

This is to certify that
SHRAVANI KULKARNI
of class
of institute
TERNA ENGINEERING COLLEGE
has ~~organized~~/participated in the
National Level Technical Paper Presentation
/ ~~Project Competition~~
~~and Won First / Second prize~~
during academic year 2015-2016.

Technical Incharge
Principal
Dr. S. M. Jagade

Date : 08-03-2016


TPCT'S
TERNA ENGINEERING COLLEGE, NERUL
AN ISO 9001 INSTITUTION

AVALON 2016

This is to certify that
SHUCHITA KAPOOR
of class
of institute
TERNA ENGINEERING COLLEGE
has ~~organized~~/participated in the
National Level Technical Paper Presentation
/ ~~Project Competition~~
~~and Won First / Second prize~~
during academic year 2015-2016.

Technical Incharge
Principal
Dr. S. M. Jagade

Date : 08-03-2016