**FEASIBILITY REPORT**

**TECHNICAL FEASIBILITY:**

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis.

1. **Understand the different technologies involved in the proposed system:**

Before commencing the project, we have to be very clear about what are the technologies that are to be required for the development of the new system.

1. **Find out whether the organization currently possesses the required technologies:**
   * Is the required technology available with the organization?
   * If so is the capacity sufficient?

For instance –

“Will the current printer be able to handle the new reports and forms required for the new system?”

**OPERATIONAL FEASIBILITY:**

Proposed projects are beneficial only if they can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

* Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.
* Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.
* Have the user been involved in the planning and development of the project?
* Early involvement reduces the chances of resistance to the system and in
* General and increases the likelihood of successful project.

Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

**ECONOMIC FEASIBILITY:**

Economic feasibility attempts 2 weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system.

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

**Operating Environment :**

**SOFTWARE REQUIREMENTS**

Operating System : Windows XP/2007 or Linux

User Interface : HTML, CSS

Client-side Scripting : JavaScript

Programming Language : Java

Framework : struts 1.x, Hibernate 3.0

IDE/Workbench : My Eclipse 8.6

Database : Oracle 10g

Server Deployment : Tomcat 6.0/7.0

**HARDWARE REQUIREMENTS**

Processor : CORE 2 DUO

Hard Disk : 160GB

RAM : 1GB or more

**Design and Implementation Constraints :**

The system is designed using the Spiral model, with constant interaction with the clients. The implementations design is done thru Frame works Struts1.xand Hibernate 3.0 with HTML, CSS, and JAVASCRIPT.

The software will be required to be maintained by the TOMCAT 6.0. The website administrator is sole responsible for the maintenance of the site all Users authentication restricted. Sufficient security through firewalls and proxy to be implemented. It also requires the Oracle 10g database to store and retrieve data.

**Project Planning:**

**SDLC METHODOLOGIES**

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

* The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
* A preliminary design is created for the new system.
* A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
* A second prototype is evolved by a fourfold procedure:

1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
2. Defining the requirements of the second prototype.
3. Planning an designing the second prototype.
4. Constructing and testing the second prototype.

* At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
* The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
* The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
* The final system is constructed, based on the refined prototype.
* The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

**The following diagram shows how a spiral model acts like:**



**Fig 1.0-Spiral Model**

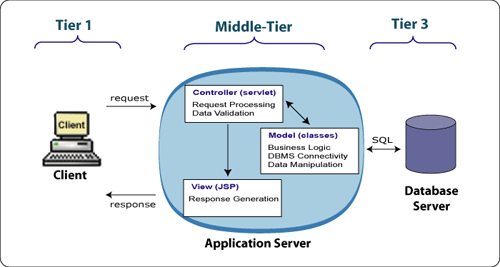
**ADVANTAGES**

* Estimates(i.e. budget, schedule etc .) become more relistic as work progresses, because important issues discoved earlier.
* It is more able to cope with the changes that are software development generally entails.
* Software engineers can get their hands in and start woring on the core of a project earlier.

**SYSTEM REQUIREMENT SPECIFICATION**

**Software Engineering Paradigm applied**

**ARCHITECTURE DIAGRAM:**



1. **THE PRESENTATION LAYER**

Also called as the client layer comprises of components that are dedicated to presenting the data to the user. For example: Windows/Web Forms and buttons, edit boxes, Text boxes, labels, grids, etc.

1. **THE BUSINESS RULES LAYER**

This layer encapsulates the Business rules or the business logic of the encapsulations. To have a separate layer for business logic is of a great advantage. This is because any changes in Business Rules can be easily handled in this layer. As long as the interface between the layers remains the same, any changes to the functionality/processing logic in this layer can be made without impacting the others. A lot of client-server apps failed to implement successfully as changing the business logic was a painful process

1. **THE DATA ACCESS LAYER**

This layer comprises of components that help in accessing the Database. If used in the right way, this layer provides a level of abstraction for the database structures. Simply put changes made to the database, tables, etc do not affect the rest of the application because of the Data Access layer. The different application layers send the data requests to this layer and receive the response from this layer.

1. **THE DATABASE LAYER**

This layer comprises of the Database Components such as DB Files, Tables, Views, etc. The Actual database could be created using SQL Server, Oracle, Flat files, etc.   
In an n-tier application, the entire application can be implemented in such a way that it is independent of the actual Database. For instance, you could change the Database Location with minimal changes to Data Access Layer. The rest of the Application should remain unaffected.

**External Interface Requirements**

3.1 User Interfaces

The application to be provided with keyboard shortcuts and a facility to use the mouse to trigger the required actions. They act as shortcuts and provide an easy navigation within the software. Error detection is handled by using Exception handling. Exception class is used to trap abnormal conditions and terminations.

3.2 Hardware Interfaces

The system requires an Internet connection with a decent band width. A printer in addition to take printouts of reports. A internet connection that is either thru dial-up, cable , modem, Wi-Fi is required. Appropriate networking and protocols should be.

3.3 Software Interfaces

The incoming data to the product would be raw text data and outgoing data would be text itself. Both input and output are handled thru dynamic HTML. A browser is required for access.

**Modules of the System**

1. Administrator Module
2. User Module
3. Encryption Decryption Module
4. Security and Authentication
5. Reports

**Functional Requirements of the Project:**

**Administrator:**

* View Users
* Accept Users
* Delete or reject user authentication

**User:**

**Banks Accounts Information**

* Add Banks Accounts Information
* View Banks Accounts Information
* Update Bank Account Information
* Delete Bank Acc Information

**Mails Information**

* Add Mails
* View Mails
* Update Mails
* Delete Mails

**Career Information**

* Add Study Details
* View Study Details
* Delete Study Details
* Update Study Details

**License Details**

* Add License Details
* View License Details
* Delete License Details
* Update License Details

**Passport Details**

* Add Passport Details
* View Passport Details
* Delete Passport Details
* Update Passport Details

**Pan card Details**

* Add pan card Details
* View pan card Details
* Delete Pan card Details
* Update Pan card Details

**Add Insurance Details**

* Add Insurance Details
* View Insurance Details
* Delete Insurance Details
* Update Insurance Details

**Imp Files Details**

* Add
* View
* Delete
* Update

**3. Encryption and Decryption Module:**

**ECC is abbreviated as Elliptic Curve Cryptography:**

The primary benefit promised by ECC is a smaller key size, reducing storage and transmission requirements—i.e., that an elliptic curve group could provide the same level of security afforded by an RSA-based system with a large modulus and correspondingly larger key—e.g., a 256bit ECC public key should provide comparable security to a 3072bit RSA public key (see [:Key sizes](http://en.wikipedia.org/wiki/Elliptic_curve_cryptography#Key_sizes)).

Public-key cryptography is based on the [intractability](http://en.wikipedia.org/wiki/Intractability_%28complexity%29#Intractability) of certain mathematical problems. Early public-key systems are secure assuming that it is difficult to [factor](http://en.wikipedia.org/wiki/Integer_factorization) a large integer composed of two or more large prime factors. For elliptic-curve-based protocols, it is assumed that finding the [discrete logarithm](http://en.wikipedia.org/wiki/Discrete_logarithm) of a random elliptic curve element with respect to a publicly known base point is infeasible. The size of the elliptic curve determines the difficulty of the problem

Example:

* Consider y2 = x3 + 2x + 3 (mod 5)

x = 0 ⇒ y2 2= 3 ⇒ no solution (mod 5)

x = 1 ⇒ y2 = 6 = 1 ⇒ y = 1,4 (mod 5)

x = 2 ⇒ y2 = 15 = 0 ⇒ y = 0 (mod 5)

x = 3 ⇒ y2 = 36 = 1 ⇒ y = 1,4 (mod 5)

x = 4 ⇒ y2 = 75 = 0 ⇒ y = 0 (mod 5)

* Then points on the elliptic curve are

(1,1) (1,4) (2,0) (3,1) (3,4) (4,0) and the point at infinity: ∞

By using above mathematical calculations are getting Encryption & decryption.

**RSA:**

The RSA algorithm was publicly described in 1977 by [Ron Rivest](http://en.wikipedia.org/wiki/Ron_Rivest), [Adi Shamir](http://en.wikipedia.org/wiki/Adi_Shamir), and [Leonard Adleman](http://en.wikipedia.org/wiki/Leonard_Adleman)

The RSA algorithm involves three steps:

[key](http://en.wikipedia.org/wiki/Key_%28cryptography%29) generation, encryption and decryption.

### Key generation

RSA involves a **public key** and a [**private key**](http://en.wikipedia.org/wiki/Private_key)**.**

The public key can be known to everyone and is used for encrypting messages.

Messages encrypted with the public key can only be decrypted using the private key.

The keys for the RSA algorithm are generated the following way:

1. Choose two distinct [prime numbers](http://en.wikipedia.org/wiki/Prime_number) *p* and *q*.
   * For security purposes, the integers *p* and *q* should be chosen at random, and should be of similar bit-length.
2. Compute *n* = *pq*.
   * *n* is used as the [modulus](http://en.wikipedia.org/wiki/Modular_arithmetic) for both the public and private keys. Its length, usually expressed in bits, is the [key length](http://en.wikipedia.org/wiki/Key_length).
3. Compute φ(*n*) = (*p* – 1)(*q* – 1), where φ is [Euler's totient function](http://en.wikipedia.org/wiki/Euler%27s_totient_function).
4. Choose an integer *e* such that 1 < *e* < φ(*n*) and [greatest common divisor](http://en.wikipedia.org/wiki/Greatest_common_divisor) gcd(*e*, φ(*n*)) = 1; i.e., *e* and φ(*n*) are [coprime](http://en.wikipedia.org/wiki/Coprime).
   * *e* is released as the public key exponent.
   * *e* having a short [bit-length](http://en.wikipedia.org/wiki/Bit-length) and small [Hamming weight](http://en.wikipedia.org/wiki/Hamming_weight) results in more efficient encryption – most commonly 216 + 1 = 65,537. However, much smaller values of *e* (such as 3) have been shown to be less secure in some settings.[[4]](http://en.wikipedia.org/wiki/RSA_%28algorithm%29#cite_note-Boneh-4)
5. Determine *d* as *d* ≡ *e*−1 (mod φ(*n*)), i.e., *d* is the [multiplicative inverse](http://en.wikipedia.org/wiki/Modular_multiplicative_inverse) of *e* (modulo φ(*n*)).

* This is more clearly stated as solve for *d* given *de* ≡ 1 (mod φ(*n*))
* This is often computed using the [extended Euclidean algorithm](http://en.wikipedia.org/wiki/Extended_Euclidean_algorithm).
* *d* is kept as the private key exponent.

By construction, *d*⋅*e* ≡ 1 (mod φ(*n*)). The **public key** consists of the modulus *n* and the public (or encryption) exponent *e*. The **private key** consists of the modulus *n* and the private (or decryption) exponent *d*, which must be kept secret. *p*, *q*, and φ(*n*) must also be kept secret because they can be used to calculate *d*.

* An alternative, used by [PKCS#1](http://en.wikipedia.org/wiki/PKCS1), is to choose *d* matching *de* ≡ 1 (mod λ) with λ = lcm(*p* − 1, *q* − 1), where lcm is the [least common multiple](http://en.wikipedia.org/wiki/Least_common_multiple). Using λ instead of φ(*n*) allows more choices for *d*. λ can also be defined using the [Carmichael function](http://en.wikipedia.org/wiki/Carmichael_function), λ(*n*).
* The [ANSI X9.31](http://en.wikipedia.org/w/index.php?title=ANSI_X9.31&action=edit&redlink=1) standard prescribes, [IEEE 1363](http://en.wikipedia.org/wiki/P1363) describes, and [PKCS#1](http://en.wikipedia.org/wiki/PKCS1) allows, that *p* and *q* match additional requirements: being [strong primes](http://en.wikipedia.org/wiki/Strong_prime), and being different enough that [Fermat factorization](http://en.wikipedia.org/wiki/Fermat_factorization) fails.

### Encryption

[Alice](http://en.wikipedia.org/wiki/Alice_and_Bob) transmits her public key (*n*, *e*) to [Bob](http://en.wikipedia.org/wiki/Alice_and_Bob) and keeps the private key secret. Bob then wishes to send message *M* to Alice.

He first turns *M* into an integer *m*, such that 0 ≤ *m* < *n* by using an agreed-upon reversible protocol known as a [padding scheme](http://en.wikipedia.org/wiki/RSA_%28algorithm%29#Padding_schemes). He then computes the ciphertext *c* corresponding to

 c \equiv m^e \pmod{n} .

This can be done quickly using the method of [exponentiation by squaring](http://en.wikipedia.org/wiki/Exponentiation_by_squaring). Bob then transmits *c* to Alice.

### Decryption

Alice can recover *m* from *c* by using her private key exponent *d* via computing

 m \equiv c^d \pmod{n} .

Given *m*, she can recover the original message *M* by reversing the padding scheme.

(In practice, there are more efficient methods of calculating *cd* using the precomputed values below.)

### A working example

Here is an example of RSA encryption and decryption. The parameters used here are artificially small, but one can also [use OpenSSL to generate and examine a real keypair](http://en.wikibooks.org/wiki/Transwiki:Generate_a_keypair_using_OpenSSL).

1. Choose two distinct prime numbers, such as

p = 61and q = 53.

1. Compute *n* = *pq* giving

n = 61 \times 53 = 3233.

1. Compute the [totient](http://en.wikipedia.org/wiki/Totient) of the product as φ(*n*) = (*p* − 1)(*q* − 1) giving

\varphi(3233) = (61 - 1)(53 - 1) = 3120.

1. Choose any number 1 < *e* < 3120 that is [coprime](http://en.wikipedia.org/wiki/Coprime) to 3120. Choosing a prime number for *e* leaves us only to check that *e* is not a divisor of 3120.

Let e = 17.

1. Compute *d*, the [modular multiplicative inverse](http://en.wikipedia.org/wiki/Modular_multiplicative_inverse) of *e* (mod φ(*n*)) yielding

d = 2753.

The **public key** is (*n* = 3233, *e* = 17). For a padded [plaintext](http://en.wikipedia.org/wiki/Plaintext) message *m*, the encryption function is *m*17 (mod 3233).

The **private key** is (*n* = 3233, *d* = 2753). For an encrypted [ciphertext](http://en.wikipedia.org/wiki/Ciphertext) *c*, the decryption function is *c*2753 (mod 3233).

For instance, in order to encrypt *m* = 65, we calculate

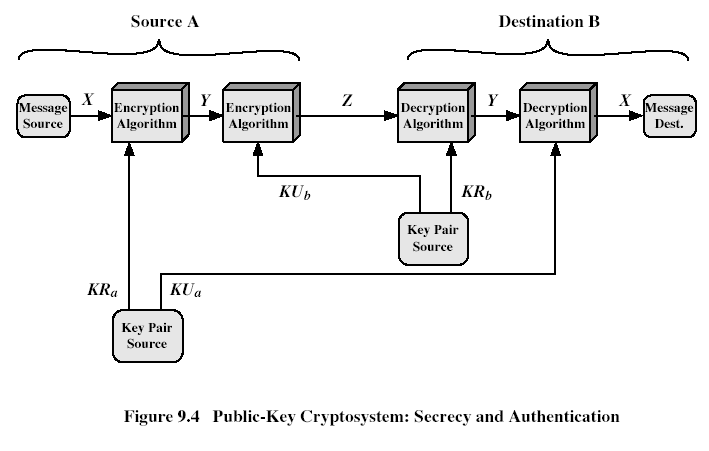
c \equiv 65^{17} \equiv 2790 \pmod{3233} .

To decrypt *c* = 2790, we calculate

m \equiv 2790^{2753} \equiv 65 \pmod{3233}.

Both of these calculations can be computed efficiently using the [square-and-multiply algorithm](http://en.wikipedia.org/wiki/Square-and-multiply_algorithm) for [modular exponentiation](http://en.wikipedia.org/wiki/Modular_exponentiation). In real life situations the primes selected would be much larger; in our example it would be relatively trivial to factor *n*, 3233, obtained from the freely available public key back to the primes *p* and *q*. Given *e*, also from the public key, we could then compute *d* and so acquire the private key.

* **Public –key cryptosystems:**



**ECC:**

**Elliptic curve cryptography (ECC)** is an approach to [public-key cryptography](http://en.wikipedia.org/wiki/Public-key_cryptography) based on the algebraic structure of [elliptic curves](http://en.wikipedia.org/wiki/Elliptic_curve) over [finite fields](http://en.wikipedia.org/wiki/Finite_field).

Ell[iptic Curve Cryptography](http://www.certicom.com/index.php/ecc) (ECC) was discovered in 1985 by Victor Miller (IBM) and Neil Koblitz (University of Washington) as an alternative mechanism for implementing [public-key cryptography](https://www.rsa.com/rsalabs/node.asp?id=2165).

I  assume that those who are going through this article will have a basic understanding of [cryptography](https://en.wikipedia.org/wiki/Cryptography) ( terms like encryption and decryption ) .

The equation of an elliptic curve is given as,

[http://bithin.files.wordpress.com/2012/02/eccequation.png?w=500](http://bithin.files.wordpress.com/2012/02/eccequation.png)

Few terms that will be used,

**E -> Elliptic Curve**

**P -> Point on the curve**

**n -> Maximum limit ( This should be a prime number )**

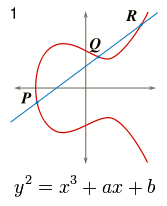
[](http://bithin.files.wordpress.com/2012/02/ell-curve.png)

Fig 3

The fig 3 show are simple elliptic curve.

### ****Key Generation****

Key generation is an important part where we have to generate both public key and private key. The sender will be encrypting the message with receiver’s public key and the receiver will decrypt its private key.

Now, we have to select a number **‘d’** within the range of **‘n’**.

Using the following equation we can generate the public key

### ****Q = d \* P****

**d** = The random number that we have selected within the range of ( **1 to n-1** ).

**P** is the point on the curve.

**‘Q’ is the public key** and **‘d’ is the private key.**

### Encryption

Let ‘m’ be the message that we are sending. We have to represent this message on the curve. This have in-depth implementation details. All the advance research on ECC is done by a company called [certicom](http://www.certicom.com/index.php/ecc).

Conside ‘m’ has the point ‘M’ on the curve ‘E’. Randomly select ‘k’ from [1 - (n-1)].

Two cipher texts will be generated let it be **C1** and **C2**.

### ****C1 = k\*P****

### C2 = M + k\*Q

C1 and C2 will be send.

### ****Decryption****

We have to get back the message ‘m’ that was send to us,

### M = C2 – d \* C1

M is the original message that we have send.

### ****Proof****

How does we get back the message,

M = C2 – d \* C1

‘M’ can be represented as ‘C2 – d \* C1′

C2 – d \* C1 = (M + k \* Q) – d \* ( k \* P )          ( C2 = M + k \* Q and C1 = k \* P )

C2 – d \* C1 = M +( k \* d\*P) – (d \* k \* P)

=  M +( k  \* d \* P) – (d \* k \*P)          ( canceling out k \* d \* P )

= M  ( Original Message )

**4. Security and Authentication:**

* Login
* Logout
* Registration
* Change Passwords
* Forget Password
* View Profile
* Update Profile

**5. Reports**

**\*** Generating Different Format report To be download (.xls, pdf, html, txt)

**Front end (Store from front) Requirements**

**Bank Account Information**

* Add Banks Accounts Information
* Update Bank Account Information

**Mails Information**

* Add Mails
* Update Mails

**Career Information**

* Add Study Details
* Update Study Details

**License Details**

* Add License Details
* Update License Details

**Passport Details**

* Add Passport Details
* Update Passport Details

**Pan card Details**

* Add pan card Details
* Update Pan card Details

**Add Insurance Details**

* Add Insurance Details
* Update Insurance Details

**Imp Files Details**

* Add
* Update

**Security and Authentication:**

* Login
* Logout
* Registration
* Change Passwords
* Update Profile

**Back end (Retrive from backend) Requirements :**

**Administrator:**

* View Users
* Accept or reject user authentication
* Delete user (or ) user Authentication

**Banks Accounts Information**

* View Banks Accounts Information
* Delete Bank Acc Information

**Mails Information**

* View Mails
* Delete Mails

**Career Information**

* View Study Details
* Delete Study Details

**License Details**

* View License Details
* Delete License Details

**Passport Details**

* View Passport Details
* Delete Passport Details

**Pan card Details**

* View pan card Details
* Delete Pan card Details

**Add Insurance Details**

* View Insurance Details
* Delete Insurance Details

**Imp Files Details**

* View
* Delete
* Login
* View Profile
* Generating Different Format report To be download (.xls, pdf, html, txt)

5). Non Funtional Requirements

5.1 Usability Requirements

(As it is a Internet Application, must have some usabilty Features. End users of this System are Unlimited and from Various Skilled groups, so that we can’t restrict them. By providing some fecilities we have to make them comfortable.)

* Colors what we use in this Web Portal design are must be attractive.
* Fonts that uses for User Interface (Customer Store front) Design are must be in Uniform.
* Easy Navigations are freferable to do any task.
* Multiple flows (ways) are freferable to do any task.
* Home page Should be Centralized System (Screen/Window) to go to any feature and to get any result.
* The fecility to return to Home page from any page Should available.
* Labels of all Objects in the entire system Must be in Understadable form(Meaningful form).

5.1 Performance Requirements

(Application’s performance not only depends on application design also on Customers System’s Configuration (both Hardware and Software), Internet Access Speed, networks and Others)

Even though the performance is not only depends on application design, our application design and implimentation also responcible for the Performance.

* It has to load, with in the Industry Standard time.
* It has to support up to 2000 Concurrent users.
* It has to update the database in short time in order to reduce the stock verfication problems.

5.3 Compatibility Requirements

(As it is a Internet Application, it has to support various Hardware configurarions, Softwares and Network Communications)

It should support all types of Hardware versions, Operating Systems and Browsers