*INTRODUCTION*

**1. INTRODUCTION**

* 1. **Background of the Project**

Each organization has its datacenters in different geographical locations far from each other, in order to withstand any natural calamities. Each datacenter in turn has numerous severs associated with it. .These servers are run round the clock and databases are accessed several times a day. When a server goes down for a few hours, it can affect the service of thousands of customers. Scheduled shut downs are also carried out for the maintenance of servers. Other maintenance activities like Disk Usage, Application deployment; Host Reboot are also carried out.

A monitoring Architecture has to be in place for ensuring proper delivery of services. A monitoring architecture also reports the performance of the IT infrastructure. eMONA is a self-service tool that is developed to constantly monitor the servers, Hard disks, databases for slow or failing components and notifies the associated personnel (via email, SMS or other alarms) in real time in case of outages or other trouble. eMONA provides a dashboard that helps the personnel not only identify the issue but also specifies the severity of the issue.

*SOFTWARE PROJECT PLAN*

**2. SOFTWARE PROJECT PLAN**

**2.1 Existing System**

The eMONA Monitoring system used to detect and alert on events real-time that may create outages that may have customer impact. eMONA provides a browser based GUI to view monitoring screens, reports and to configure alerts. It is a self-service tool where the users are notified on real time.

**2.2 Proposed System**

The proposed system adds more values to the existing monitoring architecture.

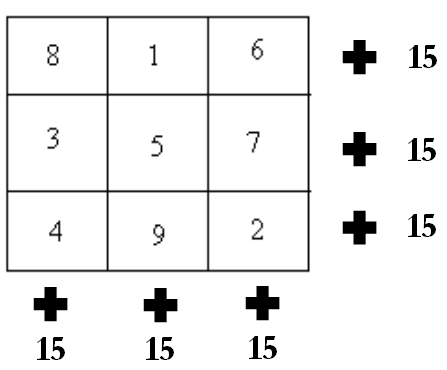
The results can be exported into multiple formats which include excel, txt, JSON. The dashboard also provides functionalitiesthe application usage can be visualised through various visual aids such as graph, pie chart, bar graph.

* 1. **Modules**

1. Dynamic charts construction.
2. Database design and Development of Web services.
3. Developing Test Cases using Junit.

***2.3.1 Magic Square Construction***

Magic Square is a matrix of size n where the numbers from 1 to in the matrix in a manner such a way that the sum when done in the row-side or column-side or diagonal-side gives a sum of. The initial magic square is constructed using the user given key. The empty cell of magic square are filled with characters in a specific order.

******

***2.3.2 Strengthened Key Extraction***

The strengthened key is extracted from the constructed magic square by traversing through the matrix in different orders such as row-wise or column-wise. The variations include traversals such as depth-first, breadth-first, zig-zag, maze etc.

***2.3.3 Encryption***

Once the strengthened key is extracted, it is fed as an input to any cryptographic algorithm which is to be enhanced in terms of security. The cipher text obtained from the encryption is highly secured in a way such that it makes the intruders almost impossible to crack the data transmitted.

***2.3.4 Variations in Magic Square Construction***

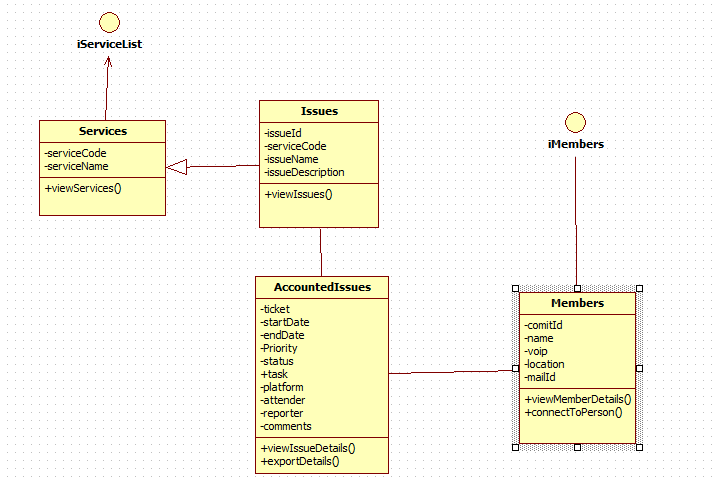
The basic method of constructing magic square includes filling the empty cells using alphabets from A to Z or in reverse order. The other variations include filling the empty cells with numeric characters or complex manipulations such as XOR of two magic squares.

*SOFTWARE REQUIREMENT SPECIFICATION*

**3. SOFTWARE REQUIREMENT SPECIFICATION**

**3.1 Functional Requirements**

***3.1.1 Class Diagram***



*Fig. no: 3.1.2 Class Diagram for user and system.*

***3.1.2 Front End Application***

In this project, the interface has been developed using high end technologies such as Angular js and mark js in order to make the application more user friendly. Datatables, a plug-in for jquery has been used to provide additional interaction to the existing HTML tables. Datatables have the following advantages:

* instant search and multi-column ordering
* Pagination
* [Scrolling options](https://datatables.net/examples/basic_init/scroll_y.html) for table viewport
* Smart handling of column widths
* State saving
* Hidden columns
* Dynamic creation of tables
* Ajax auto loading of data
* Custom DOM positioning
* Single column filtering

**3.2 NON FUNCTIONAL REQUIREMENTS**

***3.2.1 Performance Requirements***

The objective is to provide an efficient way of encryption to improve security in digital transmission of data. The user given key is strengthened and used as input for any cryptographic algorithm. The output of encryption, which is the cipher text, is stored in form of files. This file is again used for decryption of the cipher text to obtain the original message or the plain text.

***3.2.2 Interface Requirements***

The command prompt is used as an interface between the user and system where the user enters the initial key and the input file or text to be encrypted is given.

***3.2.3 Resource Requirements***

* *Software Requirements:*

Operating System - Windows

Front End - Java Applet

Language - Java

* *Hardware Requirements:*

Processor - Pentium IV or more

RAM - 512 Mb or more

Hard Disk - 50GB or Higher

Monitor - Display Panel(640 x 480)

***3.2.4 Security Requirements***

The application deals with

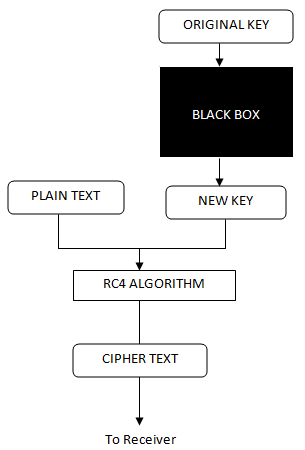
***3.2.5 Quality and Reliability Requirements***

*SYSTEM ANALYSIS*

**4. SYSTEM ANALYSIS**

**4.1 Dataflow Diagram**

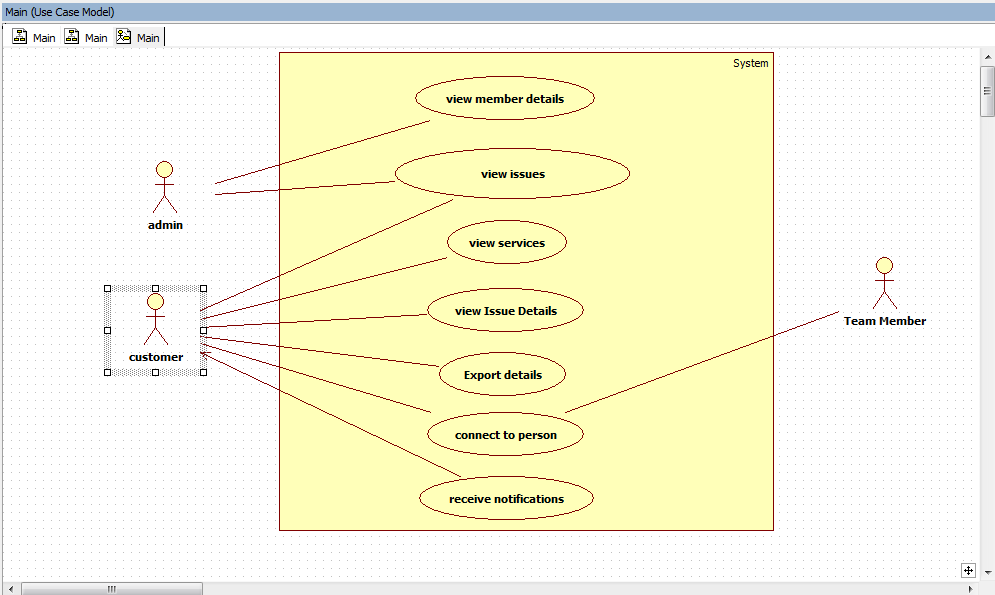
A data-flow diagram (DFD) is a graphical representation of the “flow” of data through an information system. DFDs can also be used for the visualization of data processing.



*Fig. no: 4.1.1 Dataflow diagram.*

**4.2 Use Case Diagram**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals, and any dependencies between those use cases.



*Fig. no: 4.2.1 Use Case diagram*

*SYSTEM DESIGN*

**5. SYSTEM DESIGN**

**5.1 Front End Design**

***5.1.1. Applet***

***5.1.2. Java (Programming Language)***

***5.1.3. Versions***

***5.1.4. Java platform***

***5.1.5. Implementations***

***5.1.6. Performance***

***5.1.7. Automatic memory management***

***5.1.8. JAVA (Software Platform)***

***5.1.9. Java Virtual Machine***

*CODING*

**6. CODING**

**6.1 Sample Coding**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>ira.training.xeccq49</groupId>

<artifactId>SpringEmonaWebService</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>SpringStudentRestWebService Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<jdk.version>1.7</jdk.version>

<spring.version>4.1.1.RELEASE</spring.version>

<jstl.version>1.2</jstl.version>

<junit.version>4.11</junit.version>

<logback.version>1.0.13</logback.version>

<jcl-over-slf4j.version>1.7.5</jcl-over-slf4j.version>

</properties>

<dependencies>

<!-- Unit Test -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>${junit.version}</version>

</dependency>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

<exclusions>

<exclusion>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jcl-over-slf4j</artifactId>

<version>${jcl-over-slf4j.version}</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>${logback.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- jstl -->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>${jstl.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- Jackson JSON Processor -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.4.1</version>

</dependency>

</dependencies>

<build>

<finalName>SpringStudentRestWebService</finalName>

</build>

</project>

Spring servlet.xml

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd*

*http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd"*>

<context:component-scan base-package=*"main.services"* />

<mvc:annotation-driven />

<mvc:default-servlet-handler/>

<bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>

<property name=*"driverClassName"* value=*"org.apache.derby.jdbc.ClientDriver"* />

<property name=*"url"* value=*"jdbc:derby://172.24.18.16:1527/book"* />

<property name=*"username"* value=*"user"* />

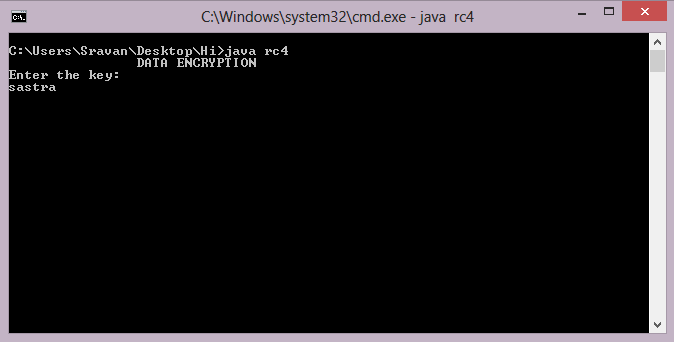
<property name=*"password"* value=*"pwd"* />

</bean>

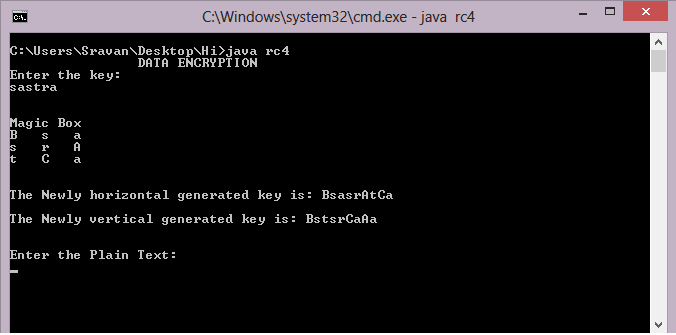
<bean id=*"issuesDAO"* class=*"main.services.IssuesDAO"* />

</beans>

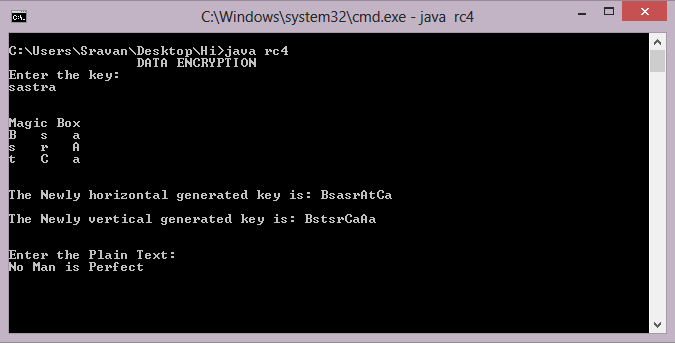
**6.2 Screenshots**

**

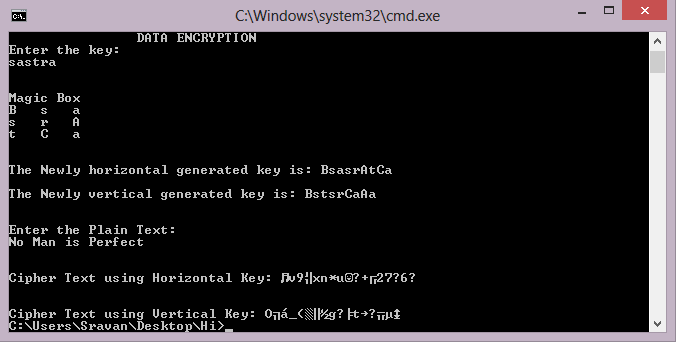
*Fig. no: 6.2.1 Getting the initial key from the user*

**

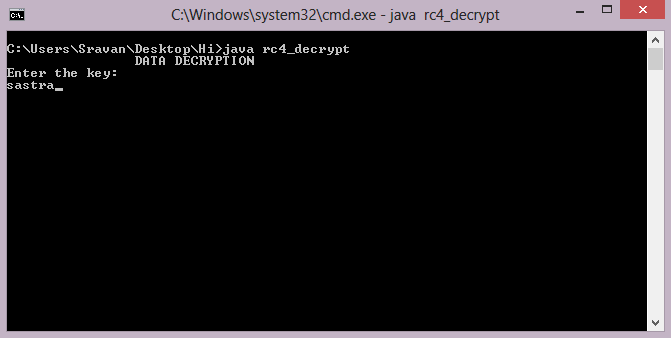
*Fig. no: 6.2.2 Magic Square Construction and Strengthened Generation*

**

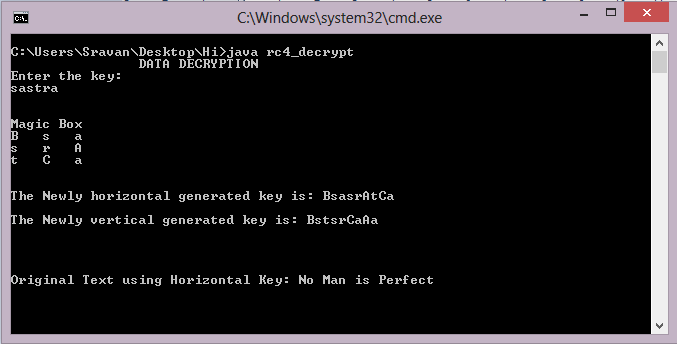
*Fig. no: 6.2.3 Getting the Plain Text from the User*

**

*Fig. no: 6.2.4 Generation of Cipher Text*

**

*Fig. no: 6.2.5 Getting the key from the user for Decryption*

**

*Fig. no: 6.2.6 Decryption of Original Message*

*IMPLEMENTATION*

**7. IMPLEMENTATION**

**7.1 Problems Faced**

The main objective is to provide purely random and dynamic keys for encryption. The process of traversing the magic square to extract different patterns of keys was difficult. The implementation of encryption algorithms using various keys simultaneously was tedious. Generating a highly secured and uncrackable cipher text was challenging.

**7.2 Lessons Learnt**

Through this project, we learnt in depth knowledge about cryptography and importance of secured communication given the advancement in hacking techniques. We learnt about the concepts of cryptanalysis and tried cracking out cipher text using brute force attack tools. We also learnt how important are the communications that happens between two parties nowadays.

*FUTURE SCOPE*

**8. FUTURE SCOPE**

In our proposed system, the order of Magic Square we use is ODD. This approach can also be implemented using Magic Squares of EVEN order. Here we use only certain patterns of key extraction. These patterns can be extended to further complex ways using several techniques computer graphics, where we can use pixel positions to extract the key from the magic square.

*REFERENCES*

**9. REFERENCES**

[1]G. Manikandan, R. Manikandan, G. Sundarganesh,2011, *A New approach for generating Strong Key in RivestCipher4 Algorithm*, Journal of Theoretical and Applied Information Technology,113-119.

[2]G.Manikandan, N.Sairam, M.Kamarasan, 2012,*A New Approach For Improving Data Security Using Iterative Blowfish Algorithm,*Research Journal of Applied Sciences, Engineering and Technology, 4(6) ,,603-607.

[3]G.Manikandan, N.Sairam, M.Kamarasan, A Hybrid Approach For Security Enhancement by Compressed Crypto-Stegno Scheme in Research Journal of Applied Sciences, Engineering and Technology,Vol 4(6) ,2012,

608-614.

[4] B.Karthikeyan, V.Vaithiyanathan, B. Thamotharan, M.Gomathymeenakshi and S.Sruthi, LSB Replacement Stegnography in an image Using Psudorandomised Key Generation. Research Journal of Applied Sciences, Engineering and Technology, 4(5), 2012, 491-494.

[5]S.Karthikeyan, N.Sairam, G.Manikandan, J.Sivaguru, A Parallel Approach For Improving Data Security in Journal of Theoretical and Applied Information Technology ,Vol 39(2) , May 2012,119-125.

[6]G.Manikandan, P.Rajendiran, K.Chakarapani, G.Krishnan,G. SundarGanesh, A Modified Crypto Scheme for Enhancing Data Security in Journal of Theoretical and applied information Technology ,Vol 35(2) , January 2012,149-154.

[7]S.Karthikeyan, N.Sairam, G.Manikandan, A New Approach For Enhancing Data Security Using Parallel Processing in Advances in Natural and Applied Sciences ,Vol 6(6) , July 2012,696-703.

[8]G.Manikandan,N.Sairam,M.Kamarasan, A New Approach for Secure Data Transfer based on Wavelet Transform in International Journal of Network Security, Vol 15(1) , 1-March 2013,88-94.