Chapter 6

Design and Implementation of E-Health Insurance Web Application
DESIGN AND IMPLEMENTATION OF E-HEALTH INSURANCE WEB APPLICATION

"Every research work should provide benefits to human Society"

This chapter discusses various technologies used in developing and deploying web application hosted by third party vendors for health insurance purpose which is developed by using latest technologies. Section 6.1 discusses the evolution of Information Technology in the field of e-Health insurance usage of Insurers and Hospitals and Insurance Companies. Section 6.2 presents the design of proposed e-Health Insurance framework for Insurance sector. Section 6.3 presents the e-Health Insurance Architecture and 6.4 gives the implementation of proposed $S^2F$RSA (Selective Secure Field RSA) Public Key Cryptosystem for securing Insurer/patient data in MySQL database in the web and authentication of users to provide controlled access to the e-Health Insurance Services respectively. Section 6.5 presents the Multi attribute Secure Decision making algorithm. Section 6.6 presents screens showing the application of $S^2F$RSA Public Key Cryptosystem used for Restrict the Specific Fields of client data. Section 6.7 provides the implementation of e-Health Insurance Web Application along with software and hardware requirements.

6.1 Introduction

E-Health Insurance is a national marketplace which provides online sales quotes for various health insurance plans and allows consumers to apply for coverage online depending upon his requirements. Using their proprietary "e-Approval" technology, e-Health Insurance can approve some applications during a single online session. In the current situation clients (Insurance companies/Consumers) often visit the web portals to select insurance schemes based on their current financial status or other parameters relating to his family or occupation or living places. Based on the client analytics insurance companies offers various schemes at clients/patients disposal.

On the e-Health Insurance website, you can search for health insurance quotes by entering your personal information and then being matched to insurance company partners. The information regarding clients profile entered on the EHI web portal is

118
secure and protected through latest security technologies. It is helpful to not only insurers and it is helpful to directly insurance companies and indirectly hospitals. The Secure e-Health Insurance system maintains the insurer/patient/client personal data and payment information in between the hospital and insurance company is securely.

6.2 E-Health Insurance Framework [29]

The proposed Framework showed in fig 6.1 offers resources and services essential for insurers working at Government organizations or private organizations (others). It is proposed that this framework of e-Health Insurance which offers expertise service to insurers regarding applying insurance, selecting best hospital and selecting best insurance company etc. The third party portal maintains the information about the hospitals, Insurance Companies and Insurers. e-Health Insurance can benefit Hospitals and Insurance Companies by obtaining/inserting information regarding insurers information without any effort, benefits to be given to insurers for various suggestions like showing the best insurance companies in his/her town and best hospitals (insurance providing hospitals) and also protecting the personal information safely etc.

The proposed E-HEALTH-INSURANCE framework [29] shown in figure-6.1 is a layered architecture contains various service layers like

1. Patient/Insurer Registration Service
2. Expert Verification Service
3. Insurance Scheme Selection Service(ISSS)
4. Expert Security Service System for Data Module
5. Exchanging Client data between HIC and HC Service Module(ECHHM)
6. Payment Module.

All the above layers can be implemented in the e-Health-Insurance Application and data storage will be done at MYSQL securely.
6.2.1 Insurer/HIC Registration Service

This Service provides an interface to Insurers/Patients, Insurance Companies and Insurance provided hospitals to Registration and add or query data by using their applications service interfaces either through browsers. IDAL services layer mainly used for acquisition of Insurer data. Health Insurance Company Data and supply solutions to users. Vast data or historical data used for various purposes is stored in health portal DB.

6.2.2 Expert Verification Service

The third party portal (HSP Server) generates the Unique-Id for each type of user, which can be helpful to authenticate the users, the Unique-Id will be generated at the time of registration of the users in E-HEALTH-INSURANCE application which is used to authenticate Insurers, Health Insurance Companies etc., to use E-HEALTH-INSURANCE services. This service generates the Health Card for
authenticated Users. This service uses the proposed Selective Secure Field RSA($^{2}$FRSA) algorithm for authenticating clients. Authenticated users can only accessing the Services of E-HEALTH-INSURANCE.

6.2.3 Insurance Scheme Selection Service (ISSS)

ISSS is a Data processing layer contains set of libraries (classes) which will accept from ISSS into and performs computations on large data sets and reports to users of E-HEALTH-INSURANCE. ISSS provides service contains libraries for Data Processing, Insurance Decision Making service which allows Decision making and Data Reporting. ISSS contains services for analyzing Data in Database of Health-Services-Portal-DB (HSPDB) which formats and validates according to rules in Health portal, Data sharing service which allows sharing of data between HIC and HS in secure manner. Insurance Decision Making service layer, which makes the decisions for each request from Insurers, Health Insurance Companies and others.

6.2.4 Expert Security Service System for Data Module

This Service supports storage of Insurer information which will be used by higher layers for providing efficient service to clients. The Insurer data like age, occupation, diseases, health status, dependents. Medicine benefit, surgical benefit, tax benefit, insurance coverage to all, dependents etc. are stored in HSPDB created in MYSQL Server as encrypted form in e-Health-Insurance application. The data in HSPDB is in encrypted format for security purpose as it will be shared between the HIC and HS.

6.2.5 Exchanging Client data between HIC, HS and Third Party Server (HSP) Service Module(ECHHM)

This service supports to sharing the data between HSP server to HIC or HS securely. Where the HIC requests the Insurer data from HSP Server, but that data will be stored as an encrypted form at HPDB. The HSP Server authenticates the HIC request and gives the response only specified data of the Insurer that means Admin restrict the some fields like diseases, health status of insurer. This service uses the Selective Secure field RSA ($^{2}$FRSA) for encryption and restrict the fields. The HIC requests the Patient/Insurer Data from HS for paying the amount, this communication also done securely using SFRSA.
6.2.6 E-Payment Module (EPM)

This Service is used for exchange of client's data between Health Insurance Company (HIC) and Hospital Server (HS) for payments, HIC and HS uses SFRSA for secure exchange of data which exposes only fields required for Hospital and HIC thus hiding remaining fields which are not essential. HIC acknowledges and requests the Patients/Clients details for health diagnosis data for payment from HS, and then HS sends the patient data which is encrypted by using $S^2FRSA$. The HIC uses the secret key to decrypt the client's data.

6.3 Secure E-Health Insurance Architecture

The Server side of E-HEALTH INSURANCE PORTAL is developed and designed by using JSP, JavaBeans, JDBC and MySQL.

**Java Server Pages:** E-HEALTH INSURANCE PORTAL application is 3-tier Client Server Architecture with major part of server side programming is done in Java Server Pages (JSP) with Apache Tomcat Server. JSP is a Java technology developed by Sun Micro Systems in 1999 which allows a developer to rapidly create dynamic Web applications. The latest version of Java is 7 (specified in JSR 342 and released in July 2011). It includes JSP 2.2, Servlets 3.1 (JSR 340), EL 3.0 (JSR 341), and JSF 2.2 (JSR 344). JSP provides a powerful, portable, and easily extensible framework that supports the development of dynamic Web applications. JSP page contains various components like directives, declarations, expressions, scriptlets, comments; action tags, bean tag libraries can be combined in many different ways to form a dynamic web page[26][54].

**Apache Tomcat Server:** Apache Tomcat (or simply Tomcat, formerly also Jakarta Tomcat) is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the Java Server Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code that executes Java servlets and renders Web pages that include Java Server Page coding. Deployment of applications can be done in following ways in Tomcat[26].

122
• **War deployment:** You can deploy the WAR file in the CATALINA_BASE directory of Tomcat and restart Tomcat to view the application. This approach is widely used in the production environment.

• **Unpacked deployment:** In this deployment method, the WAR file is extracted on the CATALINA_BASE directory for the instance. This method is commonly used in the development server.

• **Tomcat Manager:** It's a very good tool which is widely used in the production environment, mainly in remote infrastructure deployment. You can log in to the Tomcat browser from your system and deploy.

**MySQL:** MySQL is an Open Source database system which is freely available. MySQL, a SQL client/server relational database management system includes an SQL server, client programs for accessing the server, administrative tools, and a programming interface for writing your own programs. MySQL is portable and runs on commercial operating systems (such as Mac OS X, HP-UX, and Windows) and on hardware all the way up to enterprise servers. Furthermore, its performance rivals any database system you care to put up against it, and it can handle large databases with billions of rows. MySQL has many attractive features like Speed, Ease of use, Capability, Connectivity and security, Portability, Small size, Availability and cost, Open distribution and source code. ANSI standardized SQL commands are used for data creation, manipulation and transaction operations [47].

**Java Beans:** The JavaBeans APIs is to define a *software component model* for Java, so that third party Software Vendors can create and ship Java components that can be composed together into applications by end users. JavaBeans components are reusable which will be used as building blocks in composing together into compound documents. JavaBeans have following features

• A bean provides all the benefits of Java's *write once, run anywhere* paradigm. The properties, events, and methods of a bean that are exposed to another application can be controlled.

• A bean may register to receive events from other objects and can generate events that are sent to those other objects.
The configuration setting of bean can be saved in a persistent storage and restored at a later time.[92]

**Java Data Base Connectivity**: The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database. JDBC includes JDBC™ API which provides programmatic access to relational data from the Java™ programming language. Using the JDBC API, applications can execute SQL statements, retrieve results, and propagate changes back to an underlying data source. The JDBC API can also interact with multiple data sources in a distributed, heterogeneous environment. The JDBC 4.0 API is divided into two packages: java.sql and javax.sql. JDBC Driver Manager class defines objects which can connect Java applications to a JDBC driver. Driver Manager has traditionally been the backbone of the JDBC architecture [93].

E-HEALTH INSURANCE PORTAL application is developed in three-tier model, commands are sent to a "middle tier" of services like java bean components, which then sends the commands to the data source. The data source processes the commands and sends the results back to the middle tier, which then sends them to the user. MIS directors find the three-tier model very attractive because the middle tier makes it possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is that it simplifies the deployment of applications. Finally, in many cases, the three-tier architecture can provide performance advantages.

![Fig 6.2 JDBC 3-tier architecture used in E-HEALTH INSURANCE PORTAL](image-url)
6.3.1 Deploying E-HEALTH INSURANCE Web Application in Apache Tomcat Server

1. Install Tomcat server 6.0 with JDK 1.6/1.7 either in cloud space or in server system, we can check the running of web server by checking at port local host: 8080 which is default http port tomcat web server.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Contains</th>
</tr>
</thead>
<tbody>
<tr>
<td>/E-Health Insurance</td>
<td>This is the root directory of the web application. All JSP and XHTML files are stored here.</td>
</tr>
<tr>
<td>/E-Health Insurance/WEB-INF</td>
<td>This directory contains all resources related to the application that are not in the document root of the application. This is where your web application deployment descriptor is located. Note that the WEB-INF directory is not part of the public document. No files contained in this directory can be served directly to a client.</td>
</tr>
<tr>
<td>/E-Health-Insurance/WEB-INF/classes</td>
<td>This directory is where servlet and utility classes are located.</td>
</tr>
<tr>
<td>/E-Health-Insurance/WEB-INF/lib</td>
<td>This directory contains Java Archive files that the web application depends upon. For example, this is where you would place a JAR file that contained a JDBC driver.</td>
</tr>
</tbody>
</table>

Table 6.1: E-HEALTH INSURANCE Web Application Directory Structure

2. Install MySQL community server either in cloud space or in server system which is used as database server for data storage of our e-Health Insurance web application and it will be running at server port 3033.

3. E-HEALTH INSURANCE web application is deployed in Apache Tomcat server 6.0. The name of our web application will be e-Health Insurance (in tomcat). The following steps are used to deploy the application

a) Create the web application directory structure

The directory structure is created with the following structure

The name of our web application, e-Health-Insurance, is the root of our directory structure. The last step in creating the web application directory structure is
adding a deployment descriptor. web.xml file that contains only the DTD, describing
the web.xml file, and an empty <webapp/> element. Listing 1 contains the source for
a default web.xml file.

web.xml

<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc./DTD Web Application 2.3//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_3.dtd">

Now copy this file to the TOMCAT_HOME/e-Health Insurance/WEB-INF/ directory.

b) Create a web application Servlet Context,

After one has created the web application directory structure, he must add a
new Servlet Context to Tomcat. The Servlet Context defines a set of methods that are
used by components of a web application to communicate with the servlet container.
The Servlet Context acts as a container for the web application. To add a new Servlet
Context to Tomcat you need to add the following entry to the
TOMCAT_HOME/conf/server.xml file, setting the values for the path and docBase to
the name of your web application.

<Context path="/e-Health Insurance" docBase="/e-Health Insurance"debug="0"
reloadable="true" />

c) Adding JSPs, Servlets for our application

After creating the web application directories and added Servlet Context, you
can add server-side Java components i.e., JSP's and Servlets of an application like
login.jsp, cropadvice.jsp etc. Servlets if used are copied after compiling into directory
e-Health Insurance/WEB-INF/classes and add servlet entry into web.xml as

<servlet>

<servlet-name>login</servlet-name>
d) Create and deploy a WAR file.

After completing all the above steps E-HEALTH INSURANCE web application is ready for deployment, you need to package it for distribution by using war file. Change to the root directory of your web application. In this case the root directory would be TOMCAT_HOME/webapps/e-Health Insurance

1. Archive the web application:
   `jar cvf e-Health-Insurance.war`

2. Copy the resulting WAR file, e-Health-Insurance.war, to the TOMCAT_HOME/webapps directory.

3. Add a new Context entry to the /TOMCAT_HOME/conf/server.xml file, referencing the e-Health-Insurance web application.

4. Restart Tomcat.

E-Health Insurance web application is accessed by using domain name/home.jsp

---

Fig 6.3 Home Page of Health insurance services Portal of Third party for Insurer/Patient/Hospital/Insurance Company

127
The above figure shows the home page for E-Health Insurance Services Web portal, which provides the services to Insurers, Insurance Companies, Hospitals which can register and provides integrated services to Clients requiring assistance regarding selection of Insurance schemes, Hospitals available in location chosen by client.

6.4 Implementing S²FRSA Public Key Cryptosystem for Secure Data Accessing and Storage in MySQL database to store for Health Insurance Web Portal Data

The following Java code will stores Insurer data like age, occupation diseases, and health status data in encrypted format into health portal database by using Application admin keys.

```java
package health;

import java.math.BigInteger;

import java.util.*;

import java.sql.*;

import java.io.*;

public class S2FRSA {
    static BigInteger cipher[];
    static int i, k, tlen;
    static BigInteger g, e = new BigInteger("1"), o = new BigInteger("1"), n, d = new BigInteger("1"), Eu;
    static BigInteger p, q;
    static BigInteger Me, M1, C, tempc;
    static long startTime;
    static long stopTime, ptime;
    static Random rng;
    static BigInteger[] key;
```
static String Msg, PlainTxt, S, Ci, ans = "";

static int M[], Ptxt[];

static Connection con;

static PreparedStatement ps;

static ResultSet rs;

public static BigInteger enCrypt(BigInteger e, BigInteger n, BigInteger M) {
    return M.modPow(e, n);
}

public static int deCrypt(BigInteger C, BigInteger n, BigInteger d) {
    return C.modPow(d, n).intValue();
}

public static String beforeEncrypt(String args)
{
    try
    {
        con = health.DataBase.getConnection();
        ps = con.prepareStatement("select * from ekeys");
        rs = ps.executeQuery();
        if (rs.next())
        {
            n = new BigInteger("" + rs.getString(3));
            e = new BigInteger("" + rs.getString(4));
        }
    }
}
Msg=\text{\text{args}};

//System.out.println("FromS2FRSA:"+\text{\text{args}}
length:"+\text{\text{args}}.length());

\text{M} = \text{new int[args.length()];
Ptxt = \text{new int[args.length()];

\text{cipher=\text{new BigInteger[args.length()];

ans="";
for (k = 0; k < \text{args}.length(); k += 1)
{

M[k] = (int) args.charAt(k);
S = "" + M[k];
Me = \text{new BigInteger(S); //Ascii value of each Character
C= enCrypt(e, n, Me); //Encrypted text for each Character
based on Ascii value;

ans=\text{\text{ans}}+C.toString(16)+"h";
\text{\text{cipher}}[k]=C;

// System.out.println(M[k]+"=\"+C.toString(16));

130
public static String beforeDecrypt(String args)
{
    BigInteger eval;
    try {
        Ptxt=new int[100];
        con=health.DataBase.getConnection();
        ps=con.prepareStatement("select * from ekeys");
        rs=ps.executeQuery();
        if(rs.next())
        {
            n=new BigInteger(""+rs.getString(3));
            d=new BigInteger(""+rs.getString(5));
                
        }
        StringTokenizer st=new StringTokenizer(args,"h");
        int k=0;
        while(st.hasMoreTokens())
        {

    }
eval = new BigInteger(st.nextToken(), 16);

Pttx[k] = deCrypt(eval, n, d);

k++;
}

PlainTxt = new String(Pttx, 0, k);

//System.out.println("n\n PlainText : " + PlainTxt);

} catch (Exception ex) {
    System.out.println(ex.toString());
}

return PlainTxt;
}

The Following Code is Apply-Insurance bean created for storing into Database when applied by Insurer/Client.

package health;

public class ApplyInsurance
{
    String taxbnft, occupation, cashbnft, medicinebnft, surgicalbnft, insurancecoveragetoalld, premiumperyear, city, address, diseases, healthstatus, username;
    int age, dependents, diagnosis, amount;
public ApplyInsurance()
{

taxbnft=null;occupation=null;cashbnft=null;medicinebnft=null;surgicalbnft=null;insurancecoverage=null;
premianpervyear=null;city=null;address=null;diseases=null;healthstatus=null;username=null;
age=0;dependents=0;diagnosis=0;amount=0;
}

public void setUsername(String username) {
this.username=username;
System.out.println("username:"+username);
}

public String getUsername()
{
return username;
}

public void setTaxbnft(String taxbnft)
{
this.taxbnft=taxbnft;
System.out.println("Taxbnft:"+taxbnft);
}

public String getTaxbnft()
{
return taxbnft;
}
```java
public void setAge(int age)
{
    this.age=age;
}

public int getAge()
{
    return age;
}

public void setOccupation(String occupation)
{
    this.occupation=occupation;
}

public String getOccupation()
{
    return occupation;
}

public void setCashbnft(String cashbnft)
{
    this.cashbnft=cashbnft;
}

public String getCashbnft()
{

```
return cashbnft;
}

public void setDependents(int dependents)
{

this.dependents = dependents;
}

public int getDependents()
{

return dependents;
}

public void setAmount(int amount)
{

this.amount = amount;
}

public int getAmount()
{

return amount;
}

public void setDiognasis(int diognasis)
{

this.diognasis = diognasis;
}

public int getDiognasis()
{
return diagnosis;
}

public void setMedicinebnft(String medicinebnft)
{
    this.medicinebnft=medicinebnft;
}

public String getMedicinebnft()
{
    return medicinebnft;
}

public void setSurgicalbnft(String surgicalbnft)
{
    this.surgicalbnft=surgicalbnft;
}

public String getSurgicalbnft()
{
    return surgicalbnft;
}

public void setInsurancecoveragetoalld(String insurancecoveragetoalld)
{
    this.insurancecoveragetoalld=insurancecoveragetoalld;
}

public String getInsurancecoveragetoalld()
{ 
    return insurancencoveragetoallld;
}

public void setPremiumperyear(String premiumperyear) 
{
    this.premiumperyear=premiumperyear;
}

public String getPremiumperyear() 
{
    return premiumperyear;
}

public void setCity(String city) 
{
    this.city=city;
}

public String getCity() 
{
    return city;
}

public void setAddress(String address) 
{
    this.address=address;
}

public String getAddress()
{ 

    return address;

}

public void setDiseases(String diseases)
{

    this.diseases=diseases;

}

public String getDiseases()
{

    return diseases;

}

public void setHealthstatus(String healthstatus)
{

    this.healthstatus=healthstatus;

}

public String getHealthstatus()
{

    return healthstatus;

}
The following code is used to generate keys by using $S^2$FRSA cryptosystem for e-Health-Insurance Application admin which will be used for encryption and decryption of data in Health-Portal Database (HpDB). Data is posted through Apply for Insurance Bean class defined above.

```java
// Java Document

class ApplyForInsuranceAction
{
    Connection con;
    ResultSet rs;
    PreparedStatement ps;

    String etaxbnft,eage,eoccupation,ecashbnft,emedicinebnft,esurgicalbnft,ependents,ediagnosis,econtact,eaddress,ediseases,ehealthstatus,einsurance,escoverage,osid,username;

    SFRSA sfrsa=new SFRSA();
    Login log=new Login();

    public ApplyForInsuranceAction()
    {
        con=null;
        rs=null;
        ps=null;
    }
}
```
public boolean store(ApplyInsurance ai, String uname)
{
    try{
        con=health.DataBase.getConnection();
        if(con!=null)
        {
            etaxbnft=sfrsa.beforeEncrypt(ai.getTaxbnft());
            //System.out.println("etaxbnft:"+etaxbnft);

            eage=sfrsa.beforeEncrypt(""+ai.getAge());

            ecoccupation=sfrsa.beforeEncrypt(ai.getOccupation());
            ecashbnft=sfrsa.beforeEncrypt(ai.getCashbnft());
            emedicinebnft=sfrsa.beforeEncrypt(ai.getMedicinebnft());
            esurgicalbnft=sfrsa.beforeEncrypt(ai.getSurgicalbnft());
            ediagnosis=sfrsa.beforeEncrypt(""+ai.getDiagnosis());
            edependents=sfrsa.beforeEncrypt(""+ai.getDependents());
            ecity=sfrsa.beforeEncrypt(ai.getCity());
            eaddress=sfrsa.beforeEncrypt(ai.getAddress());
            ediseases=sfrsa.beforeEncrypt(ai.getDiseases());
            ehealthstatus=sfrsa.beforeEncrypt(ai.getHealthstatus());
            einscvgtoalld=sfrsa.beforeEncrypt(ai.getInsurancecoveragealld());
            epremiumamount=sfrsa.beforeEncrypt(""+ai.getAmount());
            epremiumperyear=sfrsa.beforeEncrypt(ai.getPremiumperyear());
        }
    }
//username=log.getUsername();

//if(username==null)

//System.out.println("username:"+username);

String Status="false";

ps=con.prepareStatement("insert into
applyinsurance(username,taxbenefit,age,occupation,dependents,fullcashbnt,medicine
bnt,surgicalbnt,diagnosistimes,insurancecvcrgtoalld,premiumamount,premiumperyear,
city,address,diseases,healthstatus)values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");

ps.setString(1,uname);
ps.setString(2,etaxbnt);
ps.setString(3,eage);
ps.setString(4,ecooperation);
ps.setString(5,edependents);
ps.setString(6,ecashbnt);
ps.setString(7,emedicinebnt);
ps.setString(8,esurgicalbnt);
ps.setString(9,ediagnosis);
ps.setString(10,esigncvcrgtoalld);
ps.setString(11,epremiumamount);
ps.setString(12,epremiumperyear);
ps.setString(13,ecity);
ps.setString(14,eaddress);
ps.setString(15,ediseases);
ps.setString(16,healthstatus);
int res=ps.executeUpdate();

if(res>0)return true;
else {System.out.println("some database Error");
return false;
}

} catch(Exception ex)
System.out.println("From ApplyForInsuranceActon:",ex.toString());
return false;}}

Fig 6.4 Encrypted data stored in table of health portal DB stored in MySQL database
The following screen is used by Insurer/Client for registration purpose in Web portal.

Fig 6.5 Screen showing Registration form for Client

Fig 6.6 Screen for Applying Insurance by Insurer/Patient
Fig 6.7 Showing Health Insurance Card with photo-id and Fields in Encrypted Format (Originally magnetic strip will hide encrypted details in above figure)

6.5 Implementing Threshold Multi Attribute Secure Decision Making Algorithm

The MASDMA (Multi Attribute Secure Decision Making Algorithm) algorithm generates the keys based on the client personal information like age, occupation, date of birth, diseases, health status etc and show the possible appropriate analyzed information to users based on their town, which information provides to users easily select the best hospital and best insurance company and insurance scheme. The following JSP code uses for showing the appropriate information to users based on his/her personal information.
InsuranceAction.jsp

<jsp:useBean id="insurance" class="health.ApplyInsurance">
</jsp:useBean>

<jsp:setProperty name="insurance" property="*"/>

</div class="container">
<! freshdesignweb top bar -->

<div class="freshdesignweb-top">

<a href="PatientHome.jsp">Home</a>

<span class="right">

<a href="index.html">

<strong>Logout</strong>

</a>

</span>

</div>  <!-- freshdesignweb top bar -->

<header>

<h1><span>HEALTH INSURANCE PORTAL</span></h1>

Apply For Insurance</h1>

</header>

<% 

PreparedStatement ps_1=null,ps4=null,ps3=null,ps5=null; 

Connection con=null; 

ResultSet rs=null,rs3=null,rs4=null,rs5=null,rs6=null; 

String compname[]=new String[10]; 

int successrate[]; 

int i,len,count,temp=0,j,hrcnt; 

String decision,insname,tcompname[],hnames[],temphnames[],tinsnames[]; 

String taxbntf,occupation,fullcashbntf,medicinebntf;
String surgicalbnft,insurancecvgtoalld,premiumperyear;

int age,dependents,diagnosistimes,premiumamount;

int iage,idependents,idiagnosistimes,ipremiumamount;

String itaxbnft,ioccupation,ifullcashbnft,imedicinebnft;

String isurgicalbnft,iiinsurancecvgtoalld,ipremiumperyear,icity;

String str=request.getParameter("submit");

String uname=(String)session.getAttribute("uname");

session.setAttribute("uname",uname);

health.ApplyForInsuranceAction insaction=new health.ApplyForInsuranceAction();

health.InsuranceFinal insfinal=new health.InsuranceFinal();

con=health.DataBase.getConnection();

tcompname=new String[10];

temphnames=new String[10];

tinsnames=new String[20];

<%>

<center><h1><b>The Below Information is generated from Your Data</b></h1>

<table border="1"><tr><th>Success Rate</th><th>Company Name</th><th>Decision</th>

<%

itaxbnft=request.getParameter("taxbnft");

ioccupation=request.getParameter("occupation");

ifullcashbnft=request.getParameter("cashbnft");

imedicinebnft=request.getParameter("medicinebnft");

147
isurgicalbnft=request.getParameter("surgicalbnft");
insurancecvrgtoalld=request.getParameter("inscvrgtoalld");
premiumperyear=request.getParameter("premiumperyear");
city=request.getParameter("city");
age=Integer.parseInt(request.getParameter("age"));
dependents=Integer.parseInt(request.getParameter("dependents"));
diagnosistimes=Integer.parseInt(request.getParameter("diagnosis"));
premiumamount=Integer.parseInt(request.getParameter("amount"));

if(str.equals("submit"))
{
    compname=insfinal.decisionMaking(insurance);

    successrate=insfinal.selectCompany(compname);

    len=compname.length;

    j=0;
    for(i=0;i<len;i++)
    {
        if(successrate[j]<=40&&successrate[j]>20)
        {
            tcompname[j]=compname[i];
            j++;
            %>
            <tr><td><%=successrate[i]%></td><td><%=compname[i]%></td></tr>
        </td><h2>Prefear
\texttt{able}\texttt{/h2}\texttt{/td}\texttt{/tr>}

\texttt{<%}

\}

\texttt{else}

{

\texttt{%>}

\texttt{<tr><td><%=successrate[i]%></td><td><%=compname[i]%></td><td><h2 color="red"> NOT Prefearable</h2></td></tr>}

\texttt{<%}

\}

\texttt{}%>

\texttt{if(insidection.store(insurance,uname))}

{

\texttt{out.println("Record Inserted SuccessFully");}

\}

\texttt{else}

{

\texttt{%>}

\texttt{<jsp:forward page="error.html"/>}

\texttt{<%}

\}

\texttt{%>}

\texttt{</table>}

\texttt{<br><br>}

\texttt{149}
The following are the Insurance names are suitable to Your data with Percentages:

<table>
<thead>
<tr>
<th>Insurance Name</th>
<th>Percentage</th>
</tr>
</thead>
</table>

```%
ps_l=con.prepareStatement("select * from insurances");

//ps_l.setString(1,disid);

rs=ps_l.executeQuery();

j=0;

while(rs.next())
{
    insname=rs.getString(1);
    taxbnft=rs.getString(2);
    age=rs.getInt(3);
    occupation=rs.getString(4);
    dependents=rs.getInt(5);
    fullcashbnft=rs.getString(6);
    medicinebnft=rs.getString(7);
    surgicalbnft=rs.getString(8);
    diagnosistimes=rs.getInt(9);
    insurancecvgtoall=rs.getString(10);
    premiumamount=rs.getInt(11);
    premiumperyear=rs.getString(12);//System.out.println("premiumamount:"+pre
    miumamount);

    if(age>=iage)
{  
    count=0;
    if(taxbnft.equals(itaxbnft))
    {  
        count++;  
    }
    if(dependents>=idependents)
    {  
        count++;  
    }
    if(fullcashbnft.equals(ifullcashbnft))
    {  
        count++;  
    }
    if(medicinebnft.equals(imedicinebnft))
    {  
        count++;  
    }
    if(surgicalbnft.equals(isurgicalbnft))
    {  
        count++;  
    }
    if(diagnosistimes>=idiagnosistimes)
    {  

count++; 
}
if(insurancecvgtoallld.equals(iinsurancecvgtoallld))
{
    count++; 
}
if(premiumamount>=ipremiumamount)
{
    count++; 
}
if(premiumperyear.equals(ipremiumperyear))
{
    count++; 
}
>
<%
if(count==8)
{
    tinsnames[j]=insname;
    j++; 
}>
<tr><td><%=insname%></td><td>100%</td></tr>
<%  }
if(count==7)

152
{  
    tinsnames[j]=insname;
    j++;
    %>
    <tr><td><%=insname%></td><td>90%</td></tr>
    <%
}

if(count==6)
{
    tinsnames[j]=insname;
    j++;
    %>
    <tr><td><%=insname%></td><td>80%</td></tr>
    <%
}

if(count==5)
{tinsnames[j]=insname;
    j++;
    %>
    <tr><td><%=insname%></td><td>70%</td></tr>
    <%}

153
if(count==4)
{

tinsnames[j]=insname;
j++;
%

<tr><td><%=insname%></td><td>60%</td></tr>
<%}

})})%>

</table>
<br>
<table border="1"><tr><th><b>Insurance Company Name</b></th><th><b>Insurance Name</b></th></tr>

<h1>Preferable Insurance Company wise Insurances.....</h1>

<%}

ps4=con.prepareStatement("select insurancename from companyinsurances where inscompanyname=?");

for(i=0;i<compname.length;i++)
{

if(compname[i]!=null)
{
%

<tr><td colspan="2"><b><%=compname[i]%></b></td><td><b><%=compname[i]%></b></td></tr>

154
<%  
ps4.setString(1,tcompname[i]);  
rs4=ps4.executeQuery();  
while(rs4.next())  
{  
  %>
<tr><td><td><%rs4.getString(1)%></td></tr>
<%  
  }  
%>
}
ps4.close();
rs4.close();
%

<h2><b>Target Attribute is.... 'Hospitalname'</b></h2><br />
<h2><b>Success rate is 'high or medium' decision=yes</b></h2>
</table>
<table border="1"><tr><th><b>Hospital Name</b></th><th><b>Suggestion</b></th></tr>
</table>

String srae;
ps3=con.prepareStatement("select hospitalname from hospitalcities where city=?");
ps3.setString(1,icity);
rs3=ps3.executeQuery();
rs3.last();
hrcnt=rs3.getRow();

rs3.close();

hnames=new String[hrcnt];

rs5=ps3.executeQuery();

<h2><b>Possible Hospitals based on your city and your insurance Schemes</b></h2>

<h2><b>Target Attribute is... Hospitalname</b></h2>

<h2><b>Success rate is 'high or medium' decision=yes</b></h2>

<% i=0; while(rs5.next()) {

hnames[i]=rs5.getString(1);

i++;

}%

ps5=con.prepareStatement("select successrate from hospitals where hospital name=?");

j=0;

for(i=0;i<hnames.length;i++) {

ps5.setString(1,hnames[i]);

rs6=ps5.executeQuery();

156
while(rs6.next())
{
    srate=rs6.getString(1);
    if(srate.equals("high")||srate.equals("medium"))
    {
        tempnames[j]=hnames[i];
        j++;
    }
    
    <tr><td><%=hnames[i]%></td><td>Preferable</td></tr>
    
    }else
    {
    
    <tr><td><%=hnames[i]%></td><td>NOT Preferable</td></tr>
    
    }}
}}

</table></center>

<br><br>

<center><h1><b>From The Above Information Select Hospital, Insurance Company and Insurance Type</b></h1></center>

<form name="f1" method="post" action="InsuranceSelection.jsp">
<label>Insurance Company Name</label>

<select class="select-style gender" name="inscname">
  <% for(i=0;i<tcompname.length;i++) {
    if(tcompname[i]!=null) {
      %>
  <option value="<%=tcompname[i]%>"><%=tcompname[i]%></option>
  <% } %>
  %>
</select>
<br><br>
<label>Insurance Name</label>

<select class="select-style gender" name="insname" 158
<% 
for(i=0;i<tnsnames.length;i++) 
{ 
	if(tnsnames[i]!=null) 
	{ 
		<option value="<%=tnsnames[i]%>><%=tnsnames[i]%></option>
	<% } 
	} 
%>
</select><br><br>

<input class="button" name="submit" id="submit" tabindex="1" value="submit" type="submit">

</form>

<h1><%=temphnames[0]%></h1>

</div>

</body></html>
Fig 6.8 Screens showing the possible insurers choices in e-Health Insurance
Application for selecting insurance plan after applying with his options

Fig 6.9 Screens showing the possible insurers choices in e-Health Insurance
Application for selecting insurance plan
Selection of Insurance/Hospital by Client

It uses the following JSP code (InsuranceSelection.jsp) for selecting appropriate insurance and hospital based on the data given by user/client, where the user makes the decision using the above data (screens). It shows the clear information like which hospital is preferable and which Insurance company is preferable is viewed in above screens. (based on their Entropy, root field is success rate)

InsuranceSelection.jsp

<%@ page contentType="text/html; charset=iso-8859-1" language="java" import="java.sql.*" errorPage="/" %>

<%@ page import="health.DataBase" %>

<!DOCTYPE html PUBLIC "+//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Health Insurance Portal</title>

</head>
<h1><span>HEALTH INSURANCE PORTAL</span> Apply For Insurance</h1>

```java
String hname=request.getParameter("hospital");
String inscname=request.getParameter("inscname");
String insname=request.getParameter("insname");
out.println(hname);
out.println(inscname);
out.println(insname);
String uname=(String)session.getAttribute("uname");
health.SFRSA sfrsa=new health.SFRSA();
String ehname=sfrsa.beforeEncrypt(hname);
String einsname=sfrsa.beforeEncrypt(insname);
String einscname=sfrsa.beforeEncrypt(inscname);
try{Connection con=null;
ResultSet rs=null;
PreparedStatement ps=null;
```
con = health.DataBase.getConnection();

ps = con.prepareStatement("insert into
customerinsdetails(username,hname,inscname,insname,status) values(?, ?, ?, ?, ?)");

ps.setString(1, uname);
ps.setString(2, curname);
ps.setString(3, einscname);
ps.setString(4, insname);
ps.setString(5, "false");

int res = ps.executeUpdate();

if (res > 0)
{

<%>
<center><br><br><hr><br><h1><b> Your Details Inserted Success Fully</b></h1><br></center>
<%>
}
else
{

<%>
<h1><b> Some Problem Occured... </b></h1>
<%>
}
catch(Exception ex) {out.println(ex.toString());}

<%>
6.6 Implementation of Restricting the Specific Fields of Client in Portal

The Admin/client can impose restriction on specified fields of patient/insurer, because some data must be confidential like health status, diseases etc. When the insurer enter the personal information for applying the insurance, the required information will be generated dynamically based on their age, occupation and city, from that information the user can make the decision without ambiguity.

When the Insurance Company requesting the insurer information, they don't get the full information of the insurer, because some information will be restricted by the admin as follows.
Figure 6.12 Admin select the insurer and restrict the fields based on the insurance type

Where the admin select the any one of the user and restrict the fields by using S²FRSA algorithm. Admin will get the information regarding which fields of patient/client, admin should select to encrypt for ensuring security. The field numbers may not be same for all clients, so for each client admin will hide specific fields.
HEALTH INSURANCE PORTAL
Restrict the Field Details
Customer Name: keerthi
Selected Insurance: accident plan platinum
Restrict: 4 6 11 Fields Only

Fig 6.13 Screen showing restriction of Fields to admin for each client

The following JSP code can be used to restrict the fields by admin based on their insurance scheme as follows.

RestrictFields.jsp

<%@ page contentType="text/html; charset=iso-8859-1" language="java" import="java.sql.*" errorPage="" %>

<%@ page import="health.DataBase" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<div class="container">

  <!-- freshdesignweb top bar -->

  <div class="freshdesignweb-top">

    <a href="AdminHome.jsp">Home</a>

    <span class="right">

      <a href="index.html">

        <strong>Logout</strong>

      </a>

    </span>

  </div>

  <div class="clr"></div>

</div>

<header>

  <h2 style="color:blue"><span>HEALTH INSURANCE PORTAL</span></h2>

  Admin Restrict the Fields by Selecting the Insurer

</header>

167
String uname=(String)session.getAttribute("uname");

health.SFRSA sfrsa=new health.SFRSA();

<h2 style="color:green">Welcome to...<%=uname%></h2>

<form name="f3" method="post" action="RestrictFieldsAction.jsp">

<center><table border="1"><tr><th>User Name</th><th>Insurance Name</th><th>Select</th></tr>

<%try{

String status="false";

String duname,dinsname,dlname;

Connection con=null;

ResultSet rs=null;

PreparedStatement ps=null;

con=health.DataBase.getConnection();

ps=con.prepareStatement("select username,insname,hname from customerinsdetails where status=?");

ps.setString(1,status);

rs=ps.executeQuery();

while(rs.next())
{

}

}catch(SQLException e){}

</center></table></form>
<tr><td><%=duname%></td><td><%=dinsname%></td><td><input type="checkbox" value="<%=duname%>" name="cname"></td></tr>
</form>
</div>

RestrictFieldsAction.jsp

<%@ page contentType="text/html" charset=iso-8859-1" language="java" import="java.sql.*" errorPage="" %>
<%@ page import="health.DataBase" %>
<%@ page import="health.SFRSA"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Demo Beautiful Registration Form with HTML5 and CSS3</title>

<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7;
IE=EmulateIE9">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1.0, user-scalable=no"/>

<link rel="stylesheet" type="text/css" href="style1.css" media="all" />

<link rel="stylesheet" type="text/css" href="demo.css" media="all" />

</head>

<body>

<% 

Connection con=null;

ResultSet rs=null,rs1=null;

PreparedStatement ps=null,ps1=null;

ResultSetMetaData rsmt;

String insname="";

String uname=(String)session.getAttribute("uname");

String cname=request.getParameter("cname");

session.setAttribute("cname",cname);

170
health.SFRSA sfrsa=new health.SFRSA();
con=health.DataBase.getConnection();
ps=con.prepareStatement("Select * from applyinsurance");
rs=ps.executeQuery();
rsmt=rs.getMetaData();
int ccnt=rsmt.getColumnCount();
String row[];
row=new String[ccnt];
int i,j,k=0;

j=0;
for(i=1;i<=ccnt;i++)
{
    row[j]=rsmt.getColumnName(i);
    j++;
}
ps1=con.prepareStatement("select insname from customerinsdetails where username=?");
ps1.setString(1,cname);
rs1=ps1.executeQuery();
while(rs1.next())
{
    insname=sfrsa.beforeDecrypt(rs1.getString(1));
}
%>

<div class="container">

<!-- freshdesignweb top bar -->

<div class="freshdesignweb-top">

<a href="AdminHome.jsp">Home</a>

<span class="right">

<a href="index.html">

<span>Logout</span>

</a>

</span>

</div>

</div><!--/ freshdesignweb top bar -->

<header>

<h1><span>HEALTH INSURANCE PORTAL</span></h1>

<h1>Restrict the Field Details</h1>

</header>

<div class="form">

<form id="contactform" name="registration" method="post" action="RestrictFieldsActionResults.jsp">

<center><table border="1"></center>

<tr><th>Column Names</th><th>Value</th></tr>

172
<% for(i=0;i<cnt;i++) {
<tr><td height="40" width="100"><%=row[i]%></td><td height="40"><center><input type="checkbox" name="column" value="%=row[i]%"></center></td></tr>
</%>

<tr><td colspan="2"><input name="submit" id="submit" value="submit" type="submit"></td></tr>
</table>

</form>
</div>
</div>
</body>
</html>

Restrict Fields Action Results.jsp

<%@ page content Type="text/html; charset=iso-8859-1" language="java" import="java.sql.*" error Page="" %>

<%@ page import="health.Database" %>

173
<div class="container">
  <!-- freshdesignweb top bar -->
  <div class="freshdesignweb-top">
    <a href="AdminHome.jsp">Home</a>
    <span class="right">
      <a href="index.html">Logout</a>
    </span>
  </div>
</div>
String columns[] = request.getParameterValues("column");

String cname = (String) session.getAttribute("cname");

int i;

String columnnames = "";
for (i = 0; i < columns.length; i++)
    columnnames = columnnames + columns[i] + ",";

String value = "true";

String uname = (String) session.getAttribute("uname");

health.SFRSA sfrsa = new health.SFRSA();

try{
    Connection con = null;
    ResultSet rs = null;
    PreparedStatement ps = null, ps_u = null;
    con = health.DataBase.getConnection();
    ps = con.prepareStatement("insert into customerrestrictfields(username,columnnames) values(?,?)");
    ps.setString(1, cname);
    ps.setString(2, columnnames);
int res = ps.executeUpdate();

if (res > 0)
{

\%

<center><br><br><hr><br><h1><b>Your Details Inserted Success Fully</b></h1><br><hr>

<%}

} else
{

\%

<h1><b>Some Problem Occured...</b></h1><br><hr>

<%}

ps_u = con.prepareStatement("update customerinsdetails set status=? where username=?");

ps_u.setString(1, value);

ps_u.setString(2, cname);

int res1 = ps_u.executeUpdate();

if (res1 > 0)
{

\%

<center><br><br><hr><br><h1><b>Your Status updated Success Fully</b></h1><br><hr>

<%
Where data is stored as it is in customer restrict fields table in health portal DB, the data fields are username, column names, ukey. The key will be generated based on the selected field numbers. For example the ukey for the above selected fields is 238, that is column numbers in the table (apply insurance).
Fig 6.14 Fields to be restricted are stored for each customer table in health portal DB

In the fig 6.14 shows the restricted field of insurer the admin has chosen to be viewed by insurance company. As only the specific fields are given, whenever the Insurance company request the insurer data the restrict fields are not visible to Insurance company and only the following fields are visible after giving the key regarding the client is shown. The following screens in fig 6.15 and 6.16 shows when the key is given for a specific insurer only the details entitled to company is viewed but not the other details. The code in JSP is given after the screens.
Figure 6.15 Key entered by Insurance company to view the Insurer Data/Decision

Figure 6.16 Represents the Insurer Details viewed to hospital except restricted Fields
The following JSP code can be used to implement the above screen is as follows.

ExceptRestrictFields.jsp

```jsp
<%@ page contentType="text/html; charset=iso-8859-1" language="java"
    import="java.sql.*" errorPage="" %>

<%@ page import="health.DataBase" %>

<%@ page import="java.util.*" %>

<%@ page import="health.SFRSA" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Demo Beautiful Registration Form with HTML5 and CSS3</title>
```

180
ResultSetMetaData rsmt;

String insname="", username, remcolumns="", columnnames="";

String remcol[], restrictcolumns[] = new String[10];

String key = request.getParameter("key");

String dkey="";

String uname=(String)session.getAttribute("uname");

String inscname=(String)session.getAttribute("inscname");

username=(String)session.getAttribute("username");

health.SFRSA sfsa=new health.SFRSA();

String einscname=sfsa.beforeEncrypt(inscname);

int c=0,i,l;

con=health.DataBase.getConnection();

//ps3=con.prepareStatement("select key from customerrestrictfields where username=?");

ps=con.prepareStatement("select columnnames, ukey from customerrestrictfields where username=?");

ps.setString(1, username);

rs=ps.executeQuery();

while(rs.next())
{
    columnnames=(rs.getString(1));
    dkey=rs.getString(2);
}

182
if(key.equals(dkey))
{
  ps1=con.prepareStatement("Select * from applyinsurance");
  rs1=ps1.executeQuery();
  rsmt=rs1.getMetaData();
  int ccnt=rsmt.getColumnCount();
  System.out.println("column count:"+ccnt);
  String row[];
  row=new String[ccnt];
  int j,k=0;
  j=0;
  for(i=1;i<=ccnt;i++)
  {
    row[j]=rsmt.getColumnLabel(i);
    j++;
  }
  StringTokenizer st=new StringTokenizer(columnnames,".");
  k=0;
  while(st.hasMoreTokens())
  {
    restrictcolumns[k]=(st.nextToken());
    k++;
  }
}

c=0;}

String dval="";
int len=remcolumns.length();
String fremcolumns=remcolumns.substring(0,len-1);

183
String str="select "+frcolumns+" from applyinsurance where username="+"+username+'"';

System.out.println(str);

ps2=con.prepareStatement(str);

//ps2.setString(1,username);

rs2=ps2.executeQuery();

if(rs2.next()){
    for(i=0;i<remcol.length;i++){
        if(i!=0)
            dval=sfrsa.beforeDecrypt(rs2.getString(i+1));
        else
            dval=rs2.getString(i+1);
        
        <tr><td><b>"+remcol[i]+"</b></td><td><b>"+dval+"</b></td></tr>
    
    
}  

</table></center>

</body>

</html>
6.7 Implementation of e-Health Insurance Web Application

E-HEALTH INSURANCE Web application is intended for health insurance field which will perform following activities depending upon the user like

- Patients/Insurers in getting decisions regarding Hospitals, Insurance Companies, Insurance Schemes, facilities to be used.

- Insurance Companies to access the Insurer details (Specific field's data only). Insurance Companies get the benefit for acquiring the customers through this Health Insurance portal without any additional effort.

- Hospitals access the personal data of the patient and payment details securely. Hospitals and Insurance Companies share the data like payment and patient details in an encryption mode.

- Application admin can restrict the patient/insurer details for confidentiality. Some specific fields only view by the insurance Company.

6.7.1 Software Requirements used in developing E-HEALTH INSURANCE Web application

Operating System : Windows XP/Windows 7

Language : Java 1.6/1.7

Client Side : HTML, CSS

Server Side : Java Server Pages and Bean technology

Database connectivity : JDBC

Database : MySQL and MySQL workbench

RAD Tools : Dreamweaver, Netbeans 6.0, Eclipse

Webserver : Apache Tomcat 6.0
6.7.2 **Hardware Requirements used in developing E-HEALTH INSURANCE Web application**

- Intel i3 CPU with 2.4 Ghz
- RAM with capacity 4 GB
- HDD with capacity 500 GB
- Internet facility of 2Mbps line.

6.7.3 **Minimum Requirements for accessing E-HEALTH INSURANCE Application by Clients**

- Any Intel Pentium CPU device either
- Hard disk of size 100 GB, RAM of 2GB
- Any Operating System like Linux/Windows
- Any Browser like IE/FireFox/Chrome
- Internet connection of speed 512 kbps

The following pages contain screen shots of E-HEALTH INSURANCE Web Application third party portal.

Below screen shows login page for Insurer or Hospital or Insurance company into Web portal where they can modify or add new schemes or facilities or any other information.
Figure 6.18 Login Page of e-Health-Insurance Web Application
Figure above is used by hospital authorities to register with third party with all details which will assist clients/patients to make decision in selecting the hospital. The hospital information is secured and stored in database and for each Hospital card is generated which is used for verification with third party if required which is shown in fig 6.20.

Figure 6.20 Screen showing Hospital Card generated for Hospital for Verification
Fig : 6.21 Screen showing registration of Insurance Company

The above figure is used by insurance companies which provides health insurance for clients registered in web portal. Insurance companies also give details about hospital that have agreement with company to provide financial assistance to insurers. The details are exchanged securely between stake holders in web portal by using the protocols which we have already discussed. The Insurance company information is secured and stored in database and for each insurance company Identification card is generated which is used for verification with third party if required which is shown in fig 6.22.
Figure 6.22 Screen showing Insurance Company Card generated for Insurance Company for Verification

The above figure shows the details partly remaining details are hidden in card and encrypted using $S^2$FRSA algorithm.

6.8 Conclusion

In this chapter the e-Health’s Insurance Framework is discussed with all the services it provides. The stated framework is implemented by using Java technologies like JSP, JDBC with back end of MYSQL. The proposed algorithms MASDMA, $S^2$FRSA and Protocols are implemented as part of web application and the screens are given and results are successfully shown.