**PROJECT REPORT ON**

**Collateral Pledging**

Submitted in partial fulfilment of the requirements for

the award of the degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**OF**

**SASTRA UNIVERSITY**

**Submitted by**

**GOMATHI.M**



**Under the Guidance of**

**Prof. <Guide name>**

**SASTRA UNIVERSITY, THANJAVUR**

**SCHOOL OF COMPUTING**

**SHANMUGHA**

**ARTS, SCIENCE, TECHNOLOGY & RESEARCH ACADEMY**

**(SASTRA UNIVERSITY)**

**(A University Established under section 3 of the UGC Act, 1956)**

**TIRUMALAISAMUDRAM**

**THANJAVUR – 613 401**

**April 2017**

**SCHOOL OF COMPUTING**

**SCHOOL OF COMPUTING**

**SHANMUGHA**

**ARTS, SCIENCE, TECHNOLOGY & RESEARCH ACADEMY**

**(SASTRA UNIVERSITY)**

**(A University Established under section 3 of the UGC Act, 1956)**

**TIRUMALAISAMUDRAM, THANJAVUR – 613401**



**BONAFIDE CERTIFICATE**

Certified that this project work entitled “**<Title of the Project>**” submitted to the Shanmugha Arts, Science, Technology & Research Academy (SASTRA University), Tirumalaisamudram-613401 by **<name of the student(s) with Reg. No>** in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** is the original and independent work carried out under my guidance, during the period December 2016 - April 2017**.**

|  |  |
| --- | --- |
| **INTERNAL GUIDE** | **ASSOCIATE DEAN** |
| **<Name of the guide in Capital Letter>** | **Dr. A. UMAMAKESWARI** |
| **SCHOOL OF COMPUTING** | **SCHOOL OF COMPUTING** |

Submitted for University Examination held on\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**EXAMINER - I** **EXAMINER - II**

**SCHOOL OF COMPUTING**

**SHANMUGHA**

**ARTS, SCIENCE, TECHNOLOGY & RESEARCH ACADEMY**

**(SASTRA UNIVERSITY)**

**(A University Established under section 3 of the UGC Act, 1956)**

**TIRUMALAISAMUDRAM, THANJAVUR – 613401**



**DECLARATION**

I/We submit this project work entitled “**<Title of the project>**” to the Shanmugha Arts, Science, Technology & Research Academy (SASTRA) University, Tirumalaisamudram–613 401, in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** and declare that it is our original and independent work carried out under the guidance of **Prof. <Name of the guide>**, School of Computing, SASTRA.

|  |  |  |  |
| --- | --- | --- | --- |
| **Date :** | **Name** | **:** | **Signature:** |
| **Place :** | **Reg. No:** | |  |
|  | **Name** | **:** | **Signature:** |
|  | **Reg. No:** | |  |

(You may add name & signature based on your team members)

**ACKNOWLEDGEMENT**

First and Foremost, we take pride in thanking the almighty to gave us strength for the successful completion of this project.

We have immense pleasure in expressing my heartfelt thanks to our Vice-Chancellor **prof.R.Sethuraman** for the benevolent advice and guidance during my tenure in the college.

We would like to express our gratitude to **Dr.Balachandran, Registrar,**SASTRA UNIVERSITY, for his benevolent guidance through-out my college life.

We wish to express our thanks to **Dr.S.Vaidhyasubramanian, Dean of Planning and Development** for his encouragement and providing us with all the needed amenities.

We would like to express our gratitude to **Dr.P.Swaminathan,** Dean **,** School of Computing,SASTRA UNIVERSITY, for his benevolent guidance through-out my college life.

We would like to express our gratitude to **Dr.A.Umamakeshwari,** Associate Dean **,** Computer Science and Engineering,School of Computing, SASTRA UNIVERSITY, for her benevolent guidance through-out my college life.

We wish to express our thanks to **Prof. Kamakshi . S,** Associate ProfessorComputer Science and Engineering,School of Computing, SASTRA UNIVERSITYfor her encouragement and providing us with all the needed amenities.

we deeply thank our family and friends for supporting me in all the tasks that I have carried for the successful completion of this project.

**TABLE OF FIGURES USED IN THIS PROJECT:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO: | FIG.NO | NAME OF THE FIGURE: | PAGE NUMBER: |
| 1 | FIG 4.1 | Block Diagram for LandMine Detection Robot controlled by Android |  |
| 2 | FIG 4.2 | GPIO Pin Diagram for Raspberry pi controlled by Android |  |
| 3 | FIG 4.3 | Block Diagram of L293D |  |
| 4 | FIG 5.1 | Usecase Diagram for Robot controlled by Android application |  |
| 5 | FIG 5.2(1) | Activity Diagram for LandMine Detection Robot |  |
| 6 | FIG 5.2(2) | Activity Diagram for server driven model |  |
| 7 | FIG 5.3 | Sequence Diagram for the sensor detection |  |
| 8 | FIG 5.4 | Class Diagram for the developed autonomous robot |  |
| 9 | FIG 5.5 | Collaboration Diagram for detecting landmine |  |
| 10 | FIG 7.1 | Output Screen of Android Applicatin controlling Robot |  |
| 11 | FIG 7.2 | Output Screen of Google Maps android application which retrieves path from source to destination |  |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Title** | **Page No.** |
|  |
|  | **List of Tables**  **List of Figures**  **Acknowledgement**  **Abstract** | Pages must be numbered in continuation from certificate page using Roman numerals for this section also. |
| **1** | **Introduction** |  |
|  |  |
| **2** | **Problem Statement** |  |
|  |  |  |
| **3** | **Literature Review / Literature** |  |
|  |  |  |
| **4** | **Software, Hardware Requirement Specification** |  |
|  |  |  |
| **5** | **Conceptual Model / Proposed Architecture** |  |
|  |  |  |
| **6** | **Interaction Scenario** |  |
|  |  |  |
| **7** | **Methodology and Approach** |  |
|  |  |  |
| **8** | **Output/Results** |  |
|  |  |  |
| **9** | **Conclusion** |  |
|  |  |  |
| **10** | **References** |  |
|  |  |  |
| **11** | **Appendix(If necessary)** |  |
|  |  |  |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Title** | **Page No.** |
|  |
| **i** | **Abstract** |  |
|  |  |
| **ii** | **Introduction** |  |
|  |  |  |
| **iii** | **Problem Statement** |  |
|  |  |  |
| **iv** | **Literature Survey** |  |
|  |  |  |
| **v** | **Software, Hardware Requirement Specification** |  |
|  |  |  |
| **vi** | **Conceptual Model / Proposed Architecture** |  |
|  |  |  |
| **vii** | **Interaction Scenario** |  |
|  |  |  |
| **viii** | **Methodology and Approach** |  |
|  |  |  |
| **ix** | **Output/Results** |  |
|  |  |  |
| **x** | **Conclusion** |  |
| **xi** | **References** |  |

**ABSTRACT**

In this era, money exchange has become an integral part of our daily lives. The Internet, the largest network of computer networks, is the medium usually favoured for electronic commerce because it allows an organization to cut service costs while increasing the speed of service delivery. The rapid adoption of digital payment systems and it’s increasing usage needs lakhs of transactions to be made in one second.

There are MNC’s that adapt high speed searching and caching algorithms for their clients. But small organizations that do financial transactions like chit fund transactions don’t get to have those privileges. So, all those features are incorporated in an API and made available for the clients.

The main aim of this project is to develop a RESTful API for a dynamic Maven web project, a Chit Fund System, using Spring Framework. Angular JS is used to make it highly responsive, dynamic and powerful. Data security is established using internal security systems. All data will be stored in a warehouse. Classifiers, predictive models are built and adaptive neuro-fuzzy inference system is used to predict the rates and accurate stocks for each clients. Bootstrap is used to provide user with a rich and user friendly experience.

**2)INTRODUCTION**

**EXISTING SYSTEM:**

There are many chit fund systems that exist. They provide an interface to transfer money. But the stock rates and accurate predictions are not available. The system is globally available but they do not meet the requirements of our clients.

**OBJECTIVES**

To develop a chit fund system which provides :

• Integrating with other platforms

• Organisation based customization.

• Low cost.

• Upgradability with upcoming database technologies

• Lightweight processing

• Concurrency

• High availability or disaster recovery

• Storage and compression

• Ease of integration with external libraries

• Full SQL support vs partial support

• Backup option

• Integration with reporting and archive tools

• Integration with other API’s

**3)PROBLEM STATEMENT**

A Chit Fund System can be considered as transfer of money or stocks with some predefined rules that dynamically changes. It is an agreement between the client and the host. The debtor pledges the shares as an asset against the amount of money taken from a lender and promises to return the amount within specific period. The debtor pledges the stocks as a security against the debt. According to the law, after the payment of the obligation the bank in which stocks are ledged must return the stocks to the debtor and the agreement stands void.

In Early, it was very difficult to manage this pledge process for the individual clients. Many components are needed to check the status of the individual user’s pledge process. To view the pledge details, the client has to navigate to different modules which are tedious process. Since, many components are needed to check the status of the individual user’s pledge process, the convenience of the user and the response time is affected.

In order to overcome all the burden of the user, a single page is developed to fetch all the details from various components and displayed in the required format. User can specify their own requirements and can change the columns to be displayed. Every shares and securities associated with each banks for the particular broker id can be viewed with great ease. The pledging process can be scalable for existing and new client.

**4)REQUIREMENT SPECIFICATION**

**4.1. Functional Requirement Specification :**

Functional requirement as the product capabilities are things that a product must do for its user. Functional requirements define how software behaves to meet user needs. A functional requirement is a requirement that, when satisfied, will allow the user to perform some kind of function. In Software engineering and systems engineering, a functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behaviour, and outputs.

Some of the functional requirements used in this system are :

* Data must be entered before a request can be approved.
* Clicking the Approve button moves the request to the Approval Workflow.

**4.2. Non Functional Requirement Specification :**

Non-functional requirements as the quality attributes, design and implementation constraints, and external interfaces which a product must have. Non-functional requirements may describe aspects of the system that don't relate to the execution, but rather to the evolution over time. Security, usability, testability, extensibility, reliability, scalability, portability and safety are the non-functional requirements that are met in this project.

**4.3. HARDWARE REQUIREMENT SPECIFICATION:**

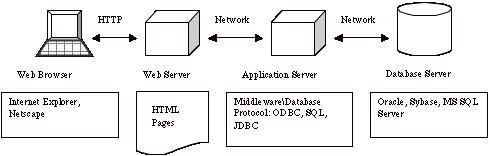
**4.3.1.Client Side:**

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirement list is often accompanied by a Hardware Compatibility List (HCL), especially in case of operating systems. An HCL lists tested, compatible and sometimes incompatible hardware devices for a particular operating system or application.

|  |  |
| --- | --- |
| **Processor** | Intel core 2 duo and advance |
| **Speed** | 2.0 GHz |
| **Hard Disk Drive** | 250 GB and above. |
| **Operating System** | Windows, linux |
| **Memory** | 2 GB RAM and above |
| **System Type** | 32,64 bit Operating System |

**4.3.2.Server Side:**

An n-tier web based database architecture model is used. Virtual dedicated hosting is followed for server side hosting. The server is up for 24/7. If the server is down, a backup alternate server is used seamlessly.



**4.4. SOFTWARE REQUIREMENTS SPECIFICATION:**

**1) Apache Tomcat Installation**

Apache Tomcat is an open-source Web server and servlet container. It requires a Java Standard Edition Runtime Environment (JRE) version 6 or later.

**STEPS TO INSTALL:**

1. Download and install JRE from http://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Download and install Apache Tomcat, a binary distribution of tomcat from <http://tomcat.apache.org/>
3. Unpack the binary distribution so that it resides in its own     directory (conventionally named "apache-tomcat-[version]").
4. Configure Environment Variables
   * 1. Set CATALINA\_HOME and CATALINA\_BASE(optional).
     2. The CATALINA\_HOME environment variable should be set to the location of the root directory of the "binary" distribution of Tomcat.
     3. The CATALINA\_BASE environment variable specifies location of the root directory of the "active configuration" of Tomcat. It is optional. It defaults to be equal to CATALINA\_HOME.
5. Set JRE\_HOME or JAVA\_HOME.

**5.1.1** The JRE\_HOME variable is used to specify location of a JRE. The JAVA\_HOME variable is used to specify location of a JDK.

**5.1.2** Using JAVA\_HOME provides access to certain additional startup options that are not allowed when JRE\_HOME  is used. If both JRE\_HOME and JAVA\_HOME are specified, JRE\_HOME is used. The best place to include these variables is a "setenv" script.

1. Other Variables like CATALINA\_OPTS are optional to set with. It allows specification of additional options for the java command to start tomcat.
2. Start Tomcat
   * 1. On Windows

                             %CATALINA\_HOME%\bin\startup.bat

                     or

                      %CATALINA\_HOME%\bin\catalina.bat start

                 or

                    $CATALINA\_HOME/bin/catalina.sh start

* + 1. After startup, the default web applications included with

**2) DERBY Installation**

cd C:\BXP Dev Env\eclipse-kepler-win32-1.0\eclipse-kepler-win32\jdk7-u51\db\bin

set  DERBY\_HOME=C:\Data\jdk7-u51\db;

set  JAVA\_HOME=C:\Data\jdk7-u51\bin;

set  PATH=C:\Data\jdk7-u51\db\bin;C:\Data\jdk7-u51\bin;%PATH%;

set  DERBY\_HOME=C:\Data\jdk7-u51\db

C:\BXP Dev Env\eclipse-kepler-win32-1.0\eclipse-kepler-win32\jdk7-u51\db\bin>startNetworkServer -h  172.00.00.00

**Bringing up your Client :**

cd C:\BXP Dev Env\eclipse-kepler-win32-1.0\eclipse-kepler-win32\jdk7-u51\db\bin

set  DERBY\_HOME=C:\Data\jdk7-u51\db;

set  JAVA\_HOME=C:\Data\jdk7-u51\bin;

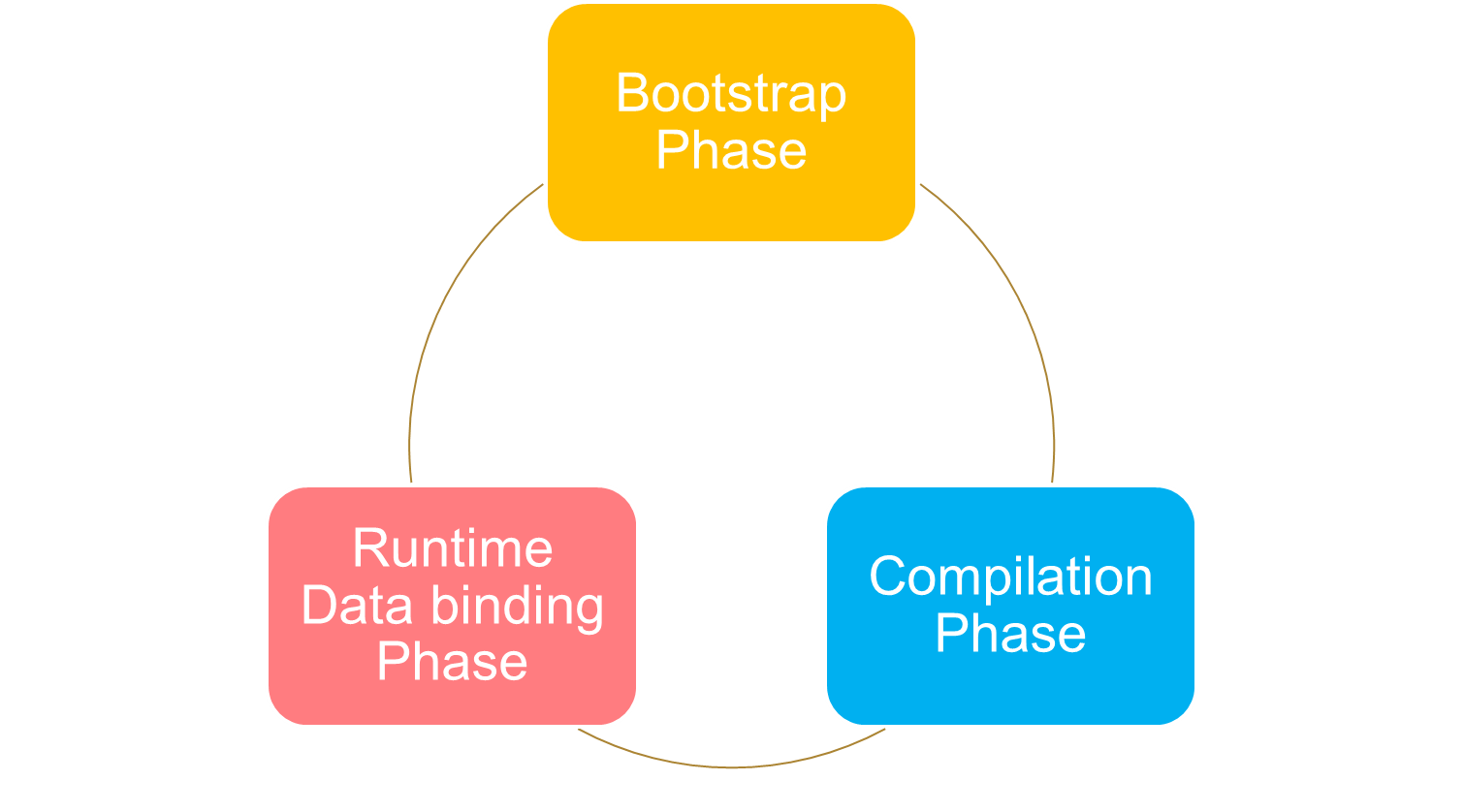
set  PATH=C:\Data\jdk7-u51\db\bin;C:\Data\jdk7-u51\bin;%PATH%;

set  DERBY\_HOME=C:\Data\jdk7-u51\db

ij;

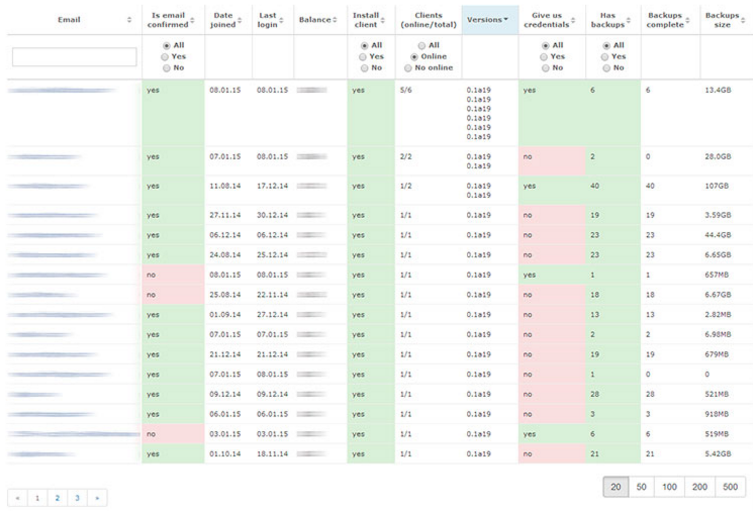
**3) AngularJS**

Angular is a client side JavaScript framework for dynamically adding interactivity for HTML. Data-binding is an automatic way of updating the view whenever the model changes, as well as updating the model whenever the view changes. It saves developer’s productivity by reducing considerable amount of code for manipulating, traversing and listening to DOM. AngularJS has a built-in dependency injection subsystem that helps the developer by making the application easier to develop, understand, and test.

 To use Angular JS include the following script in the head tag.

**<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>**

Using Angular JS with RESTful web services a responsive table like the below can be generated.



**Sample Code for ng-table**

<div class=*"row"*>

<div class=*"col-lg-3"*></div>

<div id=*"searchPP"* class=*"col-lg-6"*>

<div class=*"form-group has-warning has-feedback"*>

<input type=*"text"* class=*"form-control"* id=*"srPP"* ng-model=*"searchMsg"*>

<span class=*"glyphicon glyphicon-search form-control-feedback"*></span>

</div>

</div>

<div class=*"col-lg-3"*></div>

</div>

<br>

<table class=*"table table-bordered"* ng-controller=*"MessageController"*>

<thead>

<tr>

<th>Message ID</th>

<th>From</th>

<th>Message</th>

<th>Sent Date</th>

<th>Reply or Ignore</th>

</tr>

</thead>

<tbody ng-repeat=*"data in datas | filter:searchMsg"*>

<tr>

<td> {{ data.mid}}</td>

<td> {{ data.pid}}</td>

<td> {{ data.message}}</td>

<td> {{ data.sentDate}}</td>

<td> {{ data.replied==0 ? 'Replied' : 'Not Replied'}}</td>

</tr>

</tbody>

<tfoot>

<tr>

<td colspan=*"5"* align=*"center"*>

<!-- <label class="radio-inline">

<input type="radio" name="yes" value="yes" ng-model="myOpt"">Reply

</label> -->

<label class=*"checkbox-inline"*>

<input type=*"checkbox"* ng-model=*"myOpt"*>Reply

</label>

</td>

</tr>

<tr>

<td colspan=*"5"* ng-show=*"myOpt"* align=*"center"*>

<form method=*"Post"* action=*"UReply"*>

<div class=*"col-lg-4"*>

</div>

<div class=*"col-lg-4"*>

<div class=*"form-group"*>

<label for=*"per"*>User ID</label>

<div class=*"input-group"*>

<input type=*"number"* class=*"form-control"* id=*"per"* name=*"per"* placeholder=*"XX123"* required>

</div>

</div>

<div class=*"form-group"*>

<label for=*"rep"*>Reply Message</label>

<div class=*"input-group"*>

<textarea class=*"form-control"* id=*"rep"* name=*"rep"* required ng-model=*"sub"*></textarea>

</div>

</div>

<input type=*"submit"* name=*"repmsg"* id=*"repmsg"* value=*"Reply"* class=*"btn btn-default float-xs-right"* ng-show=*"sub"*>

</div>

<div class=*"col-lg-4"*>

</div>

</form>

</td>

</tr>

</tfoot>

</table>

<div class=*"col-lg-8"* ng-show=*"myOpt"*></div>

A controller is created for the above table. It helps to get the data from the https RESTful web services using GET.

app.controller("MessageController",**function**($scope,$http){

$http.get("message/getMsg").then(**function**(response){

$scope.datas=response.data;

});

});

**4) Spring MVC**

The Spring Web MVC framework provides Model-View-Controller (MVC) architecture and ready components that can be used to develop flexible and loosely coupled web applications. The MVC pattern results in separating the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements.

* The **Model** encapsulates the application data and in general they will consist of POJO.
* The **View** is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.
* The **Controller** is responsible for processing user requests and building an appropriate model and passes it to the view for rendering.

The following Configurations should be made,

<!DOCTYPE web-app PUBLIC

"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

"http://java.sun.com/dtd/web-app\_2\_3.dtd" >

<web-app>

<display-name>OCF</display-name>

<servlet>

<servlet-name>spring</servlet-name>

<servlet-class>

org.springframework.web.servlet.DispatcherServlet

</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>spring</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

A Controller is defined in the following way,

@RestController

@RequestMapping(value="/message")

**public** **class** MessageController {

@RequestMapping(value = "/getMsg" , method = RequestMethod.***GET*** )

**public** ArrayList<MessageBean> getAllMessages() {

DBMSDao dd=**new** DBMSDao();

ArrayList<MessageBean> mAl=dd.getMessages();

**return** mAl;

}

}

It returns a JSON object. Since the JSON format is text only, it can easily be sent to and from a server. It is used for data transfer through all the pages.

