**Online real estate management system**

**Objective:**

This project would be very useful for people where regular evaluation of people is required. Further it can also be useful for anyone who requires feedback based on objective type responses.

**Abstract:** `

Aim of this project was to develop a real estate web application using people. The real estate system Give the functionality for buyers,sellor allowing them to search for houses by features or address. It provides functionality for the seller, authorize them to log into the system and add new advertisements or delete existing ones. For this each user is provided a login account with login ID and password.

**Introduction:**

Whenever searching is done for a new house, the main focus is on the location. As location being a spatial entity we are using the advantages given by spatial databases for our application. The application provides the user to select any particular location and get information appropriately .There are some important issues in developing the real estate web application. First, the search time should be minimum. This depends on 2 techniques. Second, the web application should give the services that both buyer and seller want. Third, the web application should have a friendly user interface.

**Existing System**

The present system is not dunce proof and has certain drawbacks. Being a manual system the possible limitations and loopholes in the present system is large. Some of them are:-

1.Human resource:

The current system has too much manual work from filling a form to filing a document, delivering manifesto. This increases burden on workers but does not yield the results it should.

2.error

In current system if any modification is to be made it increases manual work and is error prone.As the system is managed and maintained by workers errors are some of the possibilities.

**Disadvantages:**

• Reducing the manual labour (Decreases Overheads).

• Avoiding Mistakes Due To Human Error (Accurate).

• Will Increase Efficiency and Save Time.

.

**Proposed System:**

We used two different distance function one without unstructure text and second with during our experimentation we found advantage of combining both unstructured and structured data. In First distance function we computed distance by introducing a relative weight factor w. The normalization of data enables relative importance of some features in datum. For example, while clustering Type of property should be same that is, all "Apartments" are clustered together. As name suggest Fuzzy this distance function computes weights dynamically based on fuzziness1.

Advantages:

 They should be able to login with their id, name.

 They should be able to give the exam as per the details entered by respective admin.

user friendly.

**HARDWARE AND SOFTWARE SPECIFICATION**:

**Software Requirement:**

1. Language - Java(JDK 1.7)

2. OS - Windows 7 32bit

3. MySql Server

4. NetBeans IDE 7.1.2

**Hardware Requirement :**

1. 1 GB RAM

2. 80 GB Hard Disk

3. Above 2GHz Processor

4. Data Card

System diagram:

View location

Sellor add details

buyer

login

Store database

**MODULES:**

1) Login

Student need to login to view his account information and buy the location. If student forgets his password he can get back old password from Forget password link. New user can register for this site by click on register link.

2) Register

New user need to register to give . Type all the details of the user like email id, name, address, contact details and submit. System validate for email id, it should be unique. In login process student need to type email id and password. user should type valid email id because if he forget password old password will be sent to this email address.

3) buyer

Buyer can view the available location by selecting any place from list, system display details of test like, place name, description, total available place, , total place.

4) sellor

It display the test page, system randomly select place from sellor and display one by one to the buyer. Student navigate the questions and select the right answer from available options, after complete the process system display the results instantly and save the place details in system for future reference.

#### Use case diagram:



Administrator

View Reports

sellor

buYer

Insert info

Give details

Registration Process

Figure (3.3.1.1.1): the basic

#### Activity diagram:

Administrator

Registration Process

Receiving details

buYer details

sellor details

location details

Receive info

specility

location

inetersted

Request Report

View Report

(a)

sellor

Insert information

Insert details

location

Store data

Request Report

View Report

(b

admin

Give exam

Register

Receive details

User info

User master

Allow user

View Report

Control master

(c)

**Java (programming language)**

### History

The JAVA language was created by James Gosling in June 1991 for use in a set top box project. The language was initially called Oak, after an oak tree that stood outside Gosling's office - and also went by the name Green - and ended up later being renamed to Java, from a list of random words. Gosling's goals were to implement a virtual machine and a language that had a familiar C/C++ style of notation. The first public implementation was Java 1.0 in 1995. It promised "Write Once, Run Anywhere” (WORA), providing no-cost runtimes on popular platforms. It was fairly secure and its security was configurable, allowing network and file access to be restricted. Major web browsers soon incorporated the ability to run secure Java applets within web pages. Java quickly became popular. With the advent of Java 2, new versions had multiple configurations built for different types of platforms. For example, J2EE was for enterprise applications and the greatly stripped down version J2ME was for mobile applications. J2SE was the designation for the Standard Edition. In 2006, for marketing purposes, new J2 versions were renamed Java EE, Java ME, and Java SE, respectively.

In 1997, Sun Microsystems approached the ISO/IEC JTC1 standards bodyand later the Ecma International to formalize Java, but it soon withdrew from the process. Java remains a standard that is controlled through the Java Community Process. At one time, Sun made most of its Java implementations available without charge although they were proprietary software. Sun's revenue from Java was generated by the selling of licenses for specialized products such as the Java Enterprise System. Sun distinguishes between its Software Development Kit (SDK) and Runtime Environment (JRE)which is a subset of the SDK, the primary distinction being that in the JRE, the compiler, utility programs, and many necessary header files are not present.

On 13 Novmber2006, Sun released much of Java as free softwareunder the terms of the GNU General Public License(GPL). On 8 May2007Sun finished the process, making all of Java's core code open source, aside from a small portion of code to which Sun did not hold the copyright.

### Primary goals

There were five primary goals in the creation of the Java language:

* It should use the object-oriented programming methodology.
* It should allow the same program to be executed on multiple operating systems.
* It should contain built-in support for using computer networks.
* It should be designed to execute code from remote sources securely.
* It should be easy to use by selecting what were considered the good parts of other object-oriented languages

### The Java Programming Language:

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

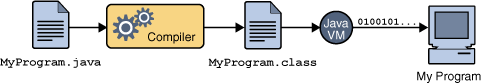
* Simple
* Architecture neutral
* Object oriented
* Portable
* Distributed
* High performance

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Each of the preceding buzzwords is explained in [The Java Language Environment](http://java.sun.com/docs/white/langenv/" \t "_blank) , a white paper written by James Gosling and Henry McGilton.

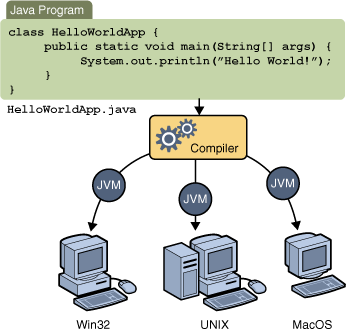
In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler.

A .class file does not contain code that is native to your processor; it instead contains byte codes — the machine language of the Java Virtual Machine[1](http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html" \l "FOOT#FOOT) (Java VM). The java launcher tool then runs your application with an instance of the Java Virtual Machine.



An overview of the software development process.

Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris TM Operating System (Solaris OS), Linux, or Mac OS. Some virtual machines, such as the Java Hot Spot virtual machineperform additional steps at runtime to give your application a performance boost. This include various tasks such as finding performance bottlenecks and recompiling (to native code) frequently used sections of code.



Through the Java VM, the same application is capable of running on multiple platforms.

### The Java Platform

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Microsoft Windows, Linux, Solaris OS, and Mac OS. Most platforms can be described as a combination of the operating system and underlying hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

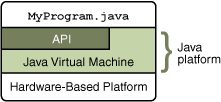
**The Java platform has two components:**

The Java Virtual Machine

The Java Application Programming Interface (API)

You've already been introduced to the Java Virtual Machine; it's the base for the Java platform and is ported onto various hardware-based platforms.

The API is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What CanJavaTechnologyDo?Highlights some of the functionality provided by the API.



The API and Java Virtual Machine insulate the program from the underlying hardware.

As a platform-independent environment, the Java platform can be a bit slower than native code. However, advances in compiler and virtual machine technologies are bringing performance close to that of native code without threatening portability.

### Java Runtime Environment

The Java Runtime Environment, or JRE, is the software required to run any application deployed on the Java Platform. End-users commonly use a JRE in software packages and Web browser plug-in. Sun also distributes a superset of the JRE called the Java 2 SDK(more commonly known as the JDK), which includes development tools such as the Javacompiler,Javadoc, Jarand debugger.

One of the unique advantages of the concept of a runtime engine is that errors (exceptions) should not 'crash' the system. Moreover, in runtime engine environments such as Java there exist tools that attach to the runtime engine and every time that an exception of interest occurs they record debugging information that existed in memory at the time the exception was thrown (stack and heap values). These Automated Exception Handling tools provide 'root-cause' information for exceptions in Java programs that run in production, testing or development environments.

**Uses OF JAVA**

Blue is a smart card enabled with the secure, cross-platform, object-oriented Java Card API and technology. Blue contains an actual on-card processing chip, allowing for enhance able and multiple functionality within a single card. Applets that comply with the Java Card API specification can run on any third-party vendor card that provides the necessary Java Card Application Environment (JCAE). Not only can multiple applet programs run on a single card, but new applets and functionality can be added after the card is issued to the customer

* Java Can be used in Chemistry.
* In NASA also Java is used.
* In 2D and 3D applications java is used.
* In Graphics Programming also Java is used.
* In Animations Java is used.
* In Online and Web Applications Java is used.

**JSP :**

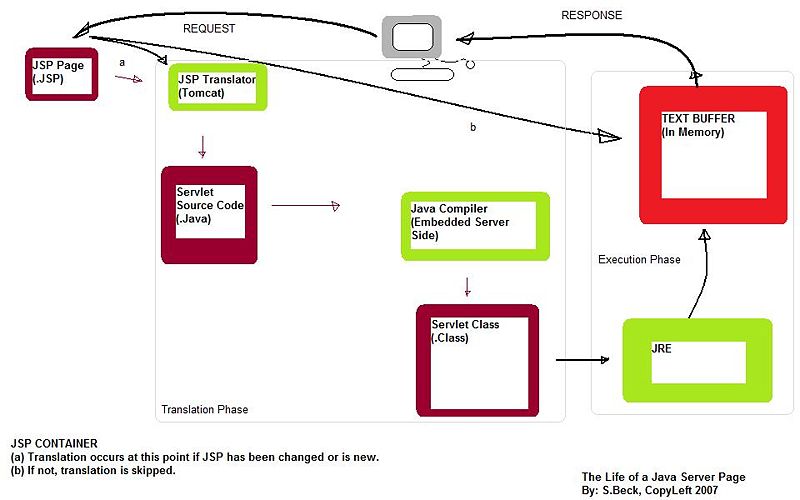
**JavaServer Pages** (**JSP**) is a Java technology that allows software developers to dynamically generate HTML, XML or other types of documents in response to a Web client request. The technology allows Java code and certain pre-defined actions to be embedded into static content.

The JSP syntax adds additional XML-like tags, called JSP actions, to be used to invoke built-in functionality. Additionally, the technology allows for the creation of JSP tag libraries that act as extensions to the standard HTML or XML tags. Tag libraries provide a platform independent way of extending the capabilities of a Web server.

JSPs are compiled into Java Servlet by a JSP compiler. A JSP compiler may generate a servlet in Java code that is then compiled by the Java compiler, or it may generate byte code for the servlet directly. JSPs can also be interpreted on-the-fly reducing the time taken to reload changes

JavaServer Pages (JSP) technology provides a simplified, fast way to create dynamic web content. JSP technology enables rapid development of web-based applications that are server and platform-independent.

**Architecture OF JSP**

**[](http://upload.wikimedia.org/wikipedia/en/4/46/JSPLife.jpg)**

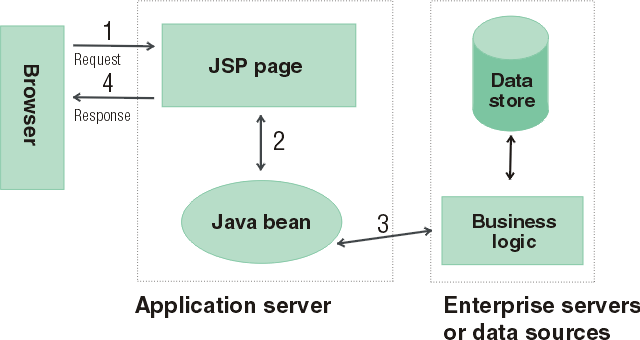
## The Advantages of JSP

Active Server Pages (ASP). ASP is a similar technology from Microsoft. The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS-specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers. Pure Servlet. JSP doesn't give you anything that you couldn't in principle do with a Servlet. But it is more convenient to write (and to modify!) regular HTML than to have a zillion println statements that generate the HTML. Plus, by separating the look from the content you can put different people on different tasks: your Web page design experts can build the HTML, leaving places for your Servlet programmers to insert the dynamic content.

Server-Side Includes (SSI). SSI is a widely-supported technology for including externally-defined pieces into a static Web page. JSP is better because it lets you use Servlet instead of a separate program to generate that dynamic part. Besides, SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like. JavaScript. JavaScript can generate HTML dynamically on the client. This is a useful capability, but only handles situations where the dynamic information is based on the client's environment.

With the exception of cookies, HTTP and form submission data is not available to JavaScript. And, since it runs on the client, JavaScript can't access server-side resources like databases, catalogs, pricing information, and the like. Static HTML. Regular HTML, of course, cannot contain dynamic information. JSP is so easy and convenient that it is quite feasible to augment HTML pages that only benefit marginally by the insertion of small amounts of dynamic data. Previously, the cost of using dynamic data would preclude its use in all but the most valuable instances.

**ARCHITECTURE OF JSP**



* The browser sends a request to a JSP page.
* The JSP page communicates with a Java bean.
* The Java bean is connected to a database.
* The JSP page responds to the browser.

**SERVLETS – FRONT END**

The Java Servlet API allows a software developer to add dynamic content to a Web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlet are the Java counterpart to non-Java dynamic Web content technologies such as PHP, CGI and ASP.NET. Servlet can maintain state across many server transactions by using HTTP cookies, session variables or URL rewriting.

The Servlet API, contained in the Java package hierarchy javax. Servlet, defines the expected interactions of a Web container and a Servlet. A Web container is essentially the component of a Web server that interacts with the Servlet. The Web container is responsible for managing the lifecycle of Servlet, mapping a URL to a particular Servlet and ensuring that the URL requester has the correct access rights.

A Servlet is an object that receives a request and generates a response based on that request. The basic Servlet package defines Java objects to represent Servlet requests and responses, as well as objects to reflect the Servlet configuration parameters and execution environment. The package javax .Servlet. Http defines HTTP-specific subclasses of the generic Servlet elements, including session management objects that track multiple requests and responses between the Web server and a client. Servlet may be packaged in a WAR file as a Web application.

Servlet can be generated automatically by Java Server Pages(JSP), or alternately by template engines such as Web Macro. Often Servlet are used in conjunction with JSPs in a pattern called "Model 2”, which is a flavour of the model-view-controller pattern.

Servlet are Java technology's answer to CGI programming. They are programs that run on a Web server and build Web pages. Building Web pages on the fly is useful (and commonly done) for a number of reasons:.

The Web page is based on data submitted by the user. For example the results pages from search engines are generated this way, and programs that process orders for e-commerce sites do this as well. The data changes frequently. For example, a weather-report or news headlines page might build the page dynamically, perhaps returning a previously built page if it is still up to date. The Web page uses information from corporate databases or other such sources. For example, you would use this for making a Web page at an on-line store that lists current prices and number of items in stock.

### The Servlet Run-time Environment

A Servlet is a Java class and therefore needs to be executed in a Java VM by a service we call a Servlet engine. The Servlet engine loads the servlet class the first time the Servlet is requested, or optionally already when the Servlet engine is started. The Servlet then stays loaded to handle multiple requests until it is explicitly unloaded or the Servlet engine is shut down.

Some Web servers, such as Sun's Java Web Server (JWS), W3C's Jigsaw and Gefion Software's Lite Web Server (LWS) are implemented in Java and have a built-in Servlet engine. Other Web servers, such as Netscape's Enterprise Server, Microsoft's Internet Information Server (IIS) and the Apache Group's Apache, require a Servlet engine add-on module. The add-on intercepts all requests for Servlet, executes them and returns the response through the Web server to the client. Examples of Servlet engine add-ons are Gefion Software's WAI Cool Runner, IBM's Web Sphere, Live Software's JRun and New Atlanta's Servlet Exec.

All Servlet API classes and a simple Servlet-enabled Web server are combined into the Java Servlet Development Kit (JSDK), available for download at Sun's official Servlet site .To get started with Servlet I recommend that you download the JSDK and play around with the sample Servlet.

**Life Cycle OF Servlet**

* The Servlet lifecycle consists of the following steps:
* The Servlet class is loaded by the container during start-up.

The container calls the init() method. This method initializes the Servlet and must be called before the Servlet can service any requests. In the entire life of a Servlet, the init() method is called only once. After initialization, the Servlet can service client-requests.

Each request is serviced in its own separate thread. The container calls the service() method of the Servlet for every request.

The service() method determines the kind of request being made and dispatches it to an appropriate method to handle the request. The developer of the Servlet must provide an implementation for these methods. If a request for a method that is not implemented by the Servlet is made, the method of the parent class is called, typically resulting in an error being returned to the requester. Finally, the container calls the destroy() method which takes the Servlet out of service. The destroy() method like init() is called only once in the lifecycle of a Servlet.

### Request and Response Objects

The do Get method has two interesting parameters: HttpServletRequest and HttpServletResponse. These two objects give you full access to all information about the request and let you control the output sent to the client as the response to the request. With CGI you read environment variables and stdin to get information about the request, but the names of the environment variables may vary between implementations and some are not provided by all Web servers.

The HttpServletRequest object provides the same information as the CGI environment variables, plus more, in a standardized way. It also provides methods for extracting HTTP parameters from the query string or the request body depending on the type of request (GET or POST). As a Servlet developer you access parameters the same way for both types of requests. Other methods give you access to all request headers and help you parse date and cookie headers.

Instead of writing the response to stdout as you do with CGI, you get an OutputStream or a PrintWriter from the HttpServletResponse. The OuputStream is intended for binary data, such as a GIF or JPEG image, and the PrintWriter for text output. You can also set all response headers and the status code, without having to rely on special Web server CGI configurations such as Non Parsed Headers (NPH). This makes your Servlet easier to install.

**ServletConfig and Servlet Context:**

There is only one Servlet Context in every application. This object can be used by all the Servlet to obtain application level information or container details. Every Servlet, on the other hand, gets its own ServletConfig object. This object provides initialization parameters for a servlet. A developer can obtain the reference to Servlet Context using either the ServletConfig object or Servlet Request object.

All servlets belong to one servlet context. In implementations of the 1.0 and 2.0 versions of the Servlet API all servlets on one host belongs to the same context, but with the 2.1 version of the API the context becomes more powerful and can be seen as the humble beginnings of an Application concept. Future versions of the API will make this even more pronounced.

Many servlet engines implementing the Servlet 2.1 API let you group a set of servlets into one context and support more than one context on the same host. The Servlet Context in the 2.1 API is responsible for the state of its servlets and knows about resources and attributes available to the servlets in the context. Here we will only look at how Servlet Context attributes can be used to share information among a group of servlets.

There are three Servlet Context methods dealing with context attributes: get Attribute, set Attribute and remove Attribute. In addition the servlet engine may provide ways to configure a servlet context with initial attribute values. This serves as a welcome addition to the servlet initialization arguments for configuration information used by a group of servlets, for instance the database identifier we talked about above, a style sheet URL for an application, the name of a mail server, etc.

**JDBC**

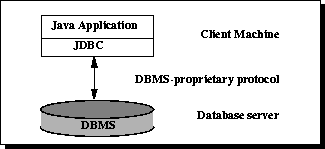
Java Database Connectivity (JDBC) is a programming framework for Java developers writing programs that access information stored in databases, spreadsheets, and flat files. JDBC is commonly used to connect a user program to a "behind the scenes" database, regardless of what database management software is used to control the database. In this way, JDBC is cross-platform. This article will provide an introduction and sample code that demonstrates database access from Java programs that use the classes of the JDBC API, which is available for free download from Sun's site.

A database that another program links to is called a data source. Many data sources, including products produced by Microsoft and Oracle, already use a standard called Open Database Connectivity (ODBC). Many legacy C and Perl programs use ODBC to connect to data sources. ODBC consolidated much of the commonality between database management systems. JDBC builds on this feature, and increases the level of abstraction. JDBC-ODBC bridges have been created to allow Java programs to connect to ODBC-enabled database software.

### JDBC Architecture

### **Two-tier and Three-tier Processing Models**

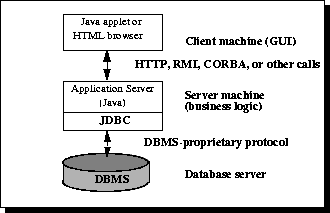
The JDBC API supports both two-tier and three-tier processing models for database access.



In the two-tier model, a Java applet or application talks directly to the data source. This requires a JDBC driver that can communicate with the particular data source being accessed. A user's commands are delivered to the database or other data source, and the results of those statements are sent back to the user. The data source may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the data source as the server. The network can be an intranet, which, for example, connects employees within a corporation, or it can be the Internet.

In the three-tier model, commands are sent to a "middle tier" of services, 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.



Until recently, the middle tier has often been written in languages such as C or C++, which offer fast performance. However, with the introduction of optimizing compilers that translate Java byte code into efficient machine-specific code and technologies such as Enterprise JavaBeans™, the Java platform is fast becoming the standard platform for middle-tier development. This is a big plus, making it possible to take advantage of Java's robustness, multithreading, and security features.

With enterprises increasingly using the Java programming language for writing server code, the JDBC API is being used more and more in the middle tier of a three-tier architecture. Some of the features that make JDBC a server technology are its support for connection pooling, distributed transactions, and disconnected rowsets. The JDBC API is also what allows access to a data source from a Java middle tier.

System Implementation:

**Testing:**

The various levels of testing are:

1. White Box Testing

2. Black Box Testing

3. Unit Testing

4. Functional Testing

5. Performance Testing

6. Integration Testing

7. Objective

8. Integration Testing

9. Validation Testing

10. System Testing

11. Structure Testing

12. Output Testing

13. User Acceptance Testing

White Box Testing

 Execution of every path in the program.

Black Box Testing

 Exhaustive input testing is required to find all errors.

Unit Testing

 Unit testing, also known as Module Testing, focuses verification efforts on the module. The module is tested separately and this is carried out at the programming stage itself.

 Unit Test comprises of the set of tests performed by an individual programmer before integration of the unit into the system.

 Unit test focuses on the smallest unit of software design- the software component or module.

 Using component level design, important control paths are tested to uncover errors within the boundary of the module.

 Unit test is white box oriented and the step can be conducted in parallel for multiple components.

Functional Testing:

 Functional test cases involve exercising the code with normal input values for which the expected results are known, as well as the boundary values

Objective:

 The objective is to take unit-tested modules and build a program structure that has been dictated by design.

Performance Testing:

 Performance testing determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization of the program unit. It occurs throughout all steps in the testing process.

Integration Testing:

 It is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with in the interface.

 It takes the unit tested modules and builds a program structure.

 All the modules are combined and tested as a whole.

 Integration of all the components to form the entire system and a overall testing is executed.

Validation Testing:

 Validation test succeeds when the software functions in a manner that can be reasonably expected by the client.

 Software validation is achieved through a series of black box testing which confirms to the requirements.

 Black box testing is conducted at the software interface.

 The test is designed to uncover interface errors, is also used to demonstrate that software functions are operational, input is properly accepted, output are produced and that the integrity of external information is maintained.

System Testing:

 Tests to find the discrepancies between the system and its original objective, current specifications and system documentation.

Structure Testing:

 It is concerned with exercising the internal logic of a program and traversing particular execution paths.

Output Testing:

 Output of test cases compared with the expected results created during design of test cases.

 Asking the user about the format required by them tests the output generated or displayed by the system under consideration.

 Here, the output format is considered into two was, one is on screen and another one is printed format.

 The output on the screen is found to be correct as the format was designed in the system design phase according to user needs.

 The output comes out as the specified requirements as the user’s hard copy.

User acceptance Testing:

 Final Stage, before handling over to the customer which is usually carried out by the customer where the test cases are executed with actual data.

 The system under consideration is tested for user acceptance and constantly keeping touch with the prospective system user at the time of developing and making changes whenever required.

 It involves planning and execution of various types of test in order to demonstrate that the implemented software system satisfies the requirements stated in the requirement document.

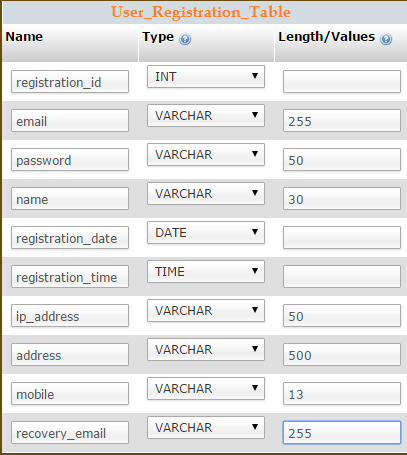
Two set of acceptance test to be run:

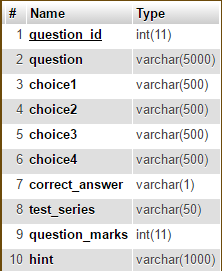
1. Those developed by quality assurance group.

2. Those developed by customer.

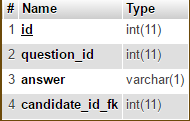
3. Methodology

**Table Description:**

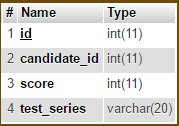
1. **USER REGISTERATION TABLE:**
2. **TEST PAPER TABLE:**

****

1. **ANSWER TABLE:**

****

1. **SCORE TABLE:**

****

**Coding:**

package com.connect.security;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/\*\*

\*

\* @author Devanand

\*/

public class ProcessQuestion extends HttpServlet {

/\*\*

\* Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

String dbUrl = "jdbc:mysql://localhost:3306/onlineexam";

String dbClass = "com.mysql.jdbc.Driver";

String query = "";

private String readTextarea(String textAreaName, HttpServletRequest request)

{

String textAreaContent = "";

textAreaContent = request.getParameter(textAreaName);

return textAreaContent;

}

private String convertTextareaContent(String content)

{

StringBuilder text = new StringBuilder(content);

int loc = (new String(text)).indexOf('\n');

while(loc > 0){

text.replace(loc, loc+1, "<BR>");

loc = (new String(text)).indexOf('\n');

}

return text.toString();

}

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

String txtQuestion = "";

String txtOptionA = "";

String txtOptionB = "";

String txtOptionC = "";

String txtOptionD = "";

String Answer = "";

int QuestionNo = 0;

HttpSession UserSession = request.getSession(false);

UserSession.setAttribute("QuestionEntryException", null);

try {

txtQuestion = readTextarea("addNewQuestion",request);

txtOptionA = readTextarea("addOptionA",request);

txtOptionB = readTextarea("addOptionB",request);

txtOptionC = readTextarea("addOptionC",request);

txtOptionD = readTextarea("addOptionD",request);

Answer = request.getParameter("CorrectAnswerRadio");

//out.print(txtOptionD);

UserSession.setAttribute("QuestionEnteredValue", txtQuestion);

UserSession.setAttribute("OptionAEnteredValue", txtOptionA);

UserSession.setAttribute("OptionBEnteredValue", txtOptionB);

UserSession.setAttribute("OptionCEnteredValue", txtOptionC);

UserSession.setAttribute("OptionDEnteredValue", txtOptionD);

if(txtQuestion==null|| "".equals(txtQuestion))

{

UserSession.setAttribute("QuestionEntryException", "Please enter a question");

UserSession.setAttribute("QuestionEnteredValue", null);

response.sendRedirect("AddQuestions.jsp");

}

else if(txtOptionA==null || "".equals(txtOptionA))

{

UserSession.setAttribute("QuestionEntryException", "Please enter Option A");

response.sendRedirect("AddQuestions.jsp");

}

else if(txtOptionB==null || "".equals(txtOptionB))

{

UserSession.setAttribute("QuestionEntryException", "Please enter Option B");

response.sendRedirect("AddQuestions.jsp");

}

else if(txtOptionC==null || "".equals(txtOptionC))

{

UserSession.setAttribute("QuestionEntryException", "Please enter Option C");

response.sendRedirect("AddQuestions.jsp");

}

else if(txtOptionD==null || "".equals(txtOptionD))

{

UserSession.setAttribute("QuestionEntryException", "Please enter Option D");

response.sendRedirect("AddQuestions.jsp");

}

else if(Answer==null)

{

UserSession.setAttribute("QuestionEntryException", "Please select the correct answser");

response.sendRedirect("AddQuestions.jsp");

}

else

{

UserSession.setAttribute("QuestionEnteredValue", null);

UserSession.setAttribute("OptionAEnteredValue", null);

UserSession.setAttribute("OptionBEnteredValue", null);

UserSession.setAttribute("OptionCEnteredValue", null);

UserSession.setAttribute("OptionDEnteredValue", null);

UserSession.setAttribute("QuestionEntryException", null);

txtQuestion = convertTextareaContent(txtQuestion);

txtOptionA = convertTextareaContent(txtOptionA);

txtOptionB = convertTextareaContent(txtOptionB);

txtOptionC = convertTextareaContent(txtOptionC);

txtOptionD = convertTextareaContent(txtOptionD);

QuestionNo = Integer.parseInt(UserSession.getAttribute("RegisterQNo").toString());

out.print(QuestionNo);

Connection con = DriverManager.getConnection (dbUrl,"root","password");

PreparedStatement insertNewQuestion;

String ExamInfoInsertQuery=""

+ "insert into Exam\_Question\_Bank "

+ "(ExamId, QuestionNo, Question, OptionA, OptionB, OptionC, OptionD, Answer) "

+ "values(?,?,?,?,?,?,?,?);";

insertNewQuestion = con.prepareStatement(ExamInfoInsertQuery);

insertNewQuestion.setInt(1, Integer.parseInt(UserSession.getAttribute("NewExamId").toString()));

insertNewQuestion.setInt(2, QuestionNo);

insertNewQuestion.setString(3, txtQuestion);

insertNewQuestion.setString(4, txtOptionA);

insertNewQuestion.setString(5, txtOptionB);

insertNewQuestion.setString(6, txtOptionC);

insertNewQuestion.setString(7, txtOptionD);

insertNewQuestion.setString(8, Answer);

int output = insertNewQuestion.executeUpdate();

QuestionNo++;

if(QuestionNo<=25)

{

UserSession.setAttribute("RegisterQNo", QuestionNo);

response.sendRedirect("AddQuestions.jsp");

}

else

{

response.sendRedirect("AdminConsoleHome.jsp");

}

}

}

catch(Exception e)

{

}

//response.sendRedirect("AddQuestions.jsp");

finally {

out.close();

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.connect.security;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/\*\*

\*

\* @author Devanand

\*/

public class RegisterNewExamination extends HttpServlet {

/\*\*

\* Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

String dbUrl = "jdbc:mysql://localhost:3306/onlineexam";

//String dbClass = "com.mysql.jdbc.Driver";

String dbClass = "com.mysql.jdbc.Driver";

String query = "";

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

HttpSession UserSession = request.getSession(false);

UserSession.setAttribute("ExamCreationError", null);

String ExamId = "";

String ExamName = "";

String ExamDesc="";

String ExamIdStatus = "";

try {

ExamId = request.getParameter("examid");

ExamName = request.getParameter("examname");

ExamDesc = request.getParameter("examdescription");

ExamIdStatus = request.getParameter("ExamIdStatus");

StringBuilder text = new StringBuilder(ExamDesc);

int loc = (new String(text)).indexOf('\n');

while(loc > 0){

text.replace(loc, loc+1, "<BR>");

loc = (new String(text)).indexOf('\n');

}

//out.println("Textarea value : "+text);

if(ExamId == null || "".equals(ExamId))

{

UserSession.setAttribute("ExamCreationError", "Exam ID may not be left blank");

response.sendRedirect("AddNewExam.jsp");

}

else if(!ExamIdStatus.contains("Available"))

{

UserSession.setAttribute("ExamCreationError", "Exam ID is not valid or already taken");

response.sendRedirect("AddNewExam.jsp");

}

else if(ExamName == null || "".equals(ExamName))

{

UserSession.setAttribute("ExamCreationError", "Exam Name may not be left blank");

response.sendRedirect("AddNewExam.jsp");

}

else if(ExamDesc == null || "".equals(ExamDesc))

{

UserSession.setAttribute("ExamCreationError", "Exam Description may not be left blank");

response.sendRedirect("AddNewExam.jsp");

}

else

{

Connection con = DriverManager.getConnection (dbUrl,"root","password");

PreparedStatement insertUserInfo;

String ExamInfoInsertQuery="insert into Exam\_List (ExamId, ExamName, ExamDescription) "

+ "values(?,?,?);";

insertUserInfo = con.prepareStatement(ExamInfoInsertQuery);

insertUserInfo.setInt(1, Integer.parseInt(ExamId));

insertUserInfo.setString(2, ExamName);

insertUserInfo.setString(3, ExamDesc);

int output = insertUserInfo.executeUpdate();

UserSession.setAttribute("NewExamId", ExamId);

UserSession.setAttribute("RegisterQNo", 1);

response.sendRedirect("AddQuestions.jsp");

}

}

catch(IOException e)

{

e.printStackTrace(out);

} catch (SQLException e) {

e.printStackTrace(out);

} catch (NumberFormatException e) {

e.printStackTrace(out);

}

finally {

out.close();

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

Screen Shots:

.

**Bibliography:**

Java Server programming Ivan Bayross.

**Conclusion:**

As far as design is concerned no design is complete ever and there are chances of improvement each moment. However performing all the necessary testing, we will conclude that our design will implement properly that it absolutely was made.

**Future Scope :**

Real Estate Management System (REMS) is an online real estate software application that **manages the overall** operational activities and processes, starting from the management of the property, to the management of real estate agencies, agents, clients and financial transactions.