7.1 MVC ARCHITECTURE:

The idea behind MVC is that the visual aspects of a system should be isolated from the internal workings, which, in turn, should be separate from the mechanism that starts and controls the internals. The MVC architecture was first prominently adopted by small talk and its practitioners but is now a widely used design pattern [1] [5]. The following figure illustrates how MVC works.
**MODEL:**

The model refers to code that manages the abstract internal state and operations of the system. It handles database access and most business logic. The model has no visual component, providing instead an application programming interface that's accessible to other parts of the system. This makes it possible to write a driver program that can test and debug the model from a simple command-line interface.

**VIEW:**

The view is the presentation layer of the system. It does no database access and contains no business logic. What little non visual code the view is limited to presentation logic, such as looping over an array of objects to display. By design, a model can be associated with more than one view, perhaps a graphical user interface (GUI) and a printed report.

**CONTROLLER:**

The controller is what manipulates the model according to user input. Based on the current view, the state of the model and the actions taken by the user, the controller calls the model API to update the model state and select the next view. Roughly speaking, the controller handles input from the user, where as the view handles output going to user.
7.2 PROCESS DIAGRAM
package bean;

import java.sql.Date;

public class FreeKiderInforBean {
    private String mailId = "";
    private Date rideDate = null;

    public String getMailId() {
        return mailId;
    }

    public void setMailId(String mailId) {
        this.mailId = mailId;
    }

    public Date getRideDate() {
        return rideDate;
    }
}
public void setRideDate(Date rideDate) {
    this.rideDate = rideDate;
}

package bean;
import java.sql.Date;

public class FreeRidingInfoBean {
    private String url = "";
    private String mailId = "";
    private Date usageDate = null;
    private double usageTime = 0;
    public String getMailId() {
        return mailId;
    }
    public void setMailId(String mailId) {
        this.mailId = mailId;
    }
    public String getUrl() {
        return url;
    }
    public void setUrl(String url) {
        this.url = url;
    }
    public double getUsageTime() {
        return usageTime;
    }
    public void setUsageTime(double usageTime) {
        this.usageTime = usageTime;
    }
}
public Date getUsageDate() {
    return usageDate;
}

public void setUsageDate(Date usageDate) {
    this.usageDate = usageDate;
}

package org.trans;
import org.com.Conn;
import java.sql.ResultSet;
import java.sql.Statement;
import org.bean.ItemOrderedBean;

public class ItemOrderedTrans {
    ItemOrderedBean iob;
    Conn conn;
    String qry;
    ResultSet rs;
    Statement stmt;
    int c;
    public ItemOrderedTrans() {
        conn=new Conn();
    }

    public int insert(ItemOrderedBean iob) {
        this.iob=iob;
        try {
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
qry = "insert into item_order values(";
qry += iob.getSale_date()+",";
qry += iob.getCx_id()+",";
qry += iob.getPrd_id()+",";
qry += iob.getPrd_name()+","
qry += iob.getPrd_category()+",");
qry += iob.getPrd_price()+",");
qry += iob.getPrd_quantity()+",");
qry += iob.getTotal_amount()+"")";

System.out.println(qry);
c = conn.executeUpdate(qry);
}
catch(Exception e)
{
e.printStackTrace();
}
return c;
}

package org.trans;
import org.bean.ProductUploadBean;
import org.com.Conn;
import java.sql.*;

public class ProductUploadTrans {

ProductUploadBean pub, pub1;
Conn conn;
int c;
ResultSet rs=null;

public ProductUploadTrans()
{
    conn=new Conn();
}

public int executeUpdate(ProductUploadBean pub)
{
    this.pub=pub;
    String qry;
    int updaterecord=0;
    qry="insert into product_upload values(";
    qry+=pub.getProd_id()+"","",
    qry+=pub.getProd_category()+"","",
    qry+=pub.getProd_name()+"","",
    qry+=pub.getProd_price()+"","",
    qry+=pub.getProd_instock()+"","",
    qry+=pub.getProd_image()+""/");

    System.out.println(qry);
    updaterecord= conn.executeUpdate(qry);
    return updaterecord;
}

public int check()
{

    try{
        String qry;

        qry="select max(prd_id) from product_upload";

        System.out.println(qry);
        rs=conn.executeQuery(qry);

    }
public int update(ProductUploadBean pub1) {
    this.pub1=pub1;
    String qry;
    int updaterecord=0;
    qry="update product-upload set prd_jnstock=
    \"+pub1.getProd_instock()\" where prd_id=
    \"+pub1.getProd_id()\";"
    System.out.println(qry);
    updaterecord= conn.executeUpdate(qry);
    return updaterecord;
}

7.4 SAMPLE INPUT AND OUTPUT
Your order is placed
Your order is placed

Welcome To Admin Login

User Name Admin
Password admin

Login
### Free Rater Report Based Billing Time

<table>
<thead>
<tr>
<th>Email ID</th>
<th>Rater Set</th>
<th>Billing Date</th>
<th>Usage Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:oes@xilfint.com">oes@xilfint.com</a>, MySp2</td>
<td>2008-03-18</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:raf@xilfint.com">raf@xilfint.com</a></td>
<td>2008-03-18</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:raf@xilfint.com">raf@xilfint.com</a></td>
<td>2008-03-18</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:raf@xilfint.com">raf@xilfint.com</a></td>
<td>2008-03-18</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:dars@xilfint.com">dars@xilfint.com</a></td>
<td>2008-03-18</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:dars@xilfint.com">dars@xilfint.com</a></td>
<td>2008-03-18</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:dars@xilfint.com">dars@xilfint.com</a></td>
<td>2008-03-18</td>
<td>03:00</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:dars@xilfint.com">dars@xilfint.com</a></td>
<td>2008-03-18</td>
<td>03:00</td>
<td></td>
</tr>
</tbody>
</table>
7.5 TESTING AND IMPLEMENTATION

7.5.1 TESTING:

> Testing is a process of executing a program with intent of finding an error.

> Testing presents an interesting anomaly for the software engineering.

> The goal of the software testing is to convince system developer and customers that the software is good enough for operational use. Testing is a process intended to build confidence in the software.

> Testing is a set of activities that can be planned in advance and conducted systematically.

> Software testing is often referred to as verification & validation.

TYPES OF TESTING:

The various types of testing are

- White Box Testing
- Black Box Testing
- Alpha Testing
- Beta Testing

WHITE BOX TESTING:

- It is also called as glass-box testing. It is a test case design method that uses the control structure of the procedural design to derive test cases.

- Using white box testing methods, the software engineer can derive test cases that
  1. Guarantee that all independent parts within a module have been exercised at least once,
  2. Exercise all logical decisions on their true and false sides.
BLACK BOX TESTING:

- It is also called as behavioral testing. It focuses on the functional requirements of the software.
- It is complementary approach that is likely to uncover a different class of errors than white box errors.
- A black box testing enables a software engineering to derive a set of input conditions that will fully exercise all functional requirements for a program.

ALPHA TESTING:

Alpha testing is the software prototype stage when the software is first able to run. It will not have all the intended functionality, but it will have core functions and will be able to accept inputs and generate outputs. An alpha test usually takes place in the developer's offices on a separate system.

BETA TESTING:

The Beta test is a "live" application of the software in an environment that cannot be controlled by the developer. The beta test is conducted at one or more customer sites by the end user of the software.

TESTING TOOLS:

WIN RUNNER:

We use Win Runner as a load testing tool operating at the GUI layer as it allows us to record and play back user actions from a vast variety of user applications as if a real user had manually executed those actions.

LOAD RUNNER:

With Load Runner, you can obtain an accurate picture of end-to-end system performance. Verify that new or upgraded applications meet specified performance requirements.
7.5.2 TESTING USED IN THIS THESIS:

SYSTEM TESTING:

Testing of the debugging programs is one of the most critical aspects of the computer programming triggers, without programs that works, the system would never produce the output for which it was designed. Testing is best performed when user development are asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands [2] [3].

UNIT TESTING:

In this testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In this testing step each module is found to working satisfactorily as regard to the expected output from the module. There are some validation checks for fields also. It is very easy to find error debut in the system [4].

FUTURE ENHANCEMENTS

- Can be enhanced to attract the prospective customers.
- More graphs showing projections and graphs for the depth of vulranableness could be generated.
7.6 OBSERVATION

In this chapter, the developed system “Graph based Web log classification” overcomes from all the problems that existing system one has. The system effectively maintains the register records in the database. As the system developed based on a frame work, it is easy to enhance the system in future. The framework separates the business logic from the presentation logic. So, that one can easily modify or update the presentation without known of business tier and vice-versa. The GUI of the system gives the user friendliness to operate the system. No need of training to operate the system. Reports and searching for records is error prone, tedious as well as time consuming environment. The system makes report generation process from tedious to easy one. Unauthorized persons could not able to access the system. This makes the system security one. It becomes quite unreliable on keeping accumulated large number of records in the manual system. But the developed system overcomes from this problem. The data in the database is maintained with integrity and security.
7.7 REFERENCES

1. The Complete Reference – The J2EE 1.4 Tutorial
   Eric Armstrong,
   Jennifer Ball,
   Debbie Carson,
   Tata McGraw-Hill Edition,
   New Delhi.

2. The Complete Reference-JSP 2.0
   Phil Hanna
   Tata McGraw-Hill Edition,
   New Delhi.

3. Head First Servlets & JSP
   Ken Arnold,
   McNealy,
   O’REILLY Edition.

4. Professional JSP
   Karl Avedal,
   Danny Ayers,
   Timothy Briggs,
   Carl Burnham,
   Wrox Press Ltd,
   Calcutta.
   Tata McGraw-Hill Edition,
   New Delhi.

5. Thinking in Java
   Bruce Eckel,
   O’REILLY Edition.