



P-1637



ANNA UNIVERSITY: CHENNAI 600 025

**BONAFIDE CERTIFICATE**

**WEB BASED BUG TRACKING APPLICATION**

**A PROJECT REPORT**

*Submitted by*

<b>Muthukumar. T</b>	<b>71202205025</b>
<b>Prithvirajkumar. D</b>	<b>71202205030</b>
<b>Siby Mathew Kuntharayil</b>	<b>71202205046</b>

*in partial fulfillment for the award of the degree*

*Of*

**BACHELOR OF TECHNOLOGY  
IN  
INFORMATION TECHNOLOGY**

**KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE**

**ANNA UNIVERSITY: CHENNAI 600 025**

**APRIL 2006**

Certified that this project report "WEB BASED BUG TRACKING APPLICATION" is the bonafide work of "MUTHUKUMAR.T, PRITHVIRAJKUMAR. D, SIBY MATHEW KUNTHARAYIL" who carried out the project work under my supervision.

  
SIGNATURE

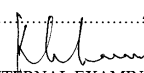
Prof. Dr. G.Gopalsamy  
**HEAD OF THE DEPARTMENT**  
Information Technology  
Kumaraguru College of Technology  
Coimbatore – 641 006

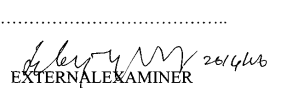
  
SIGNATURE

Mr. E.A.Vimal ,M.E.,  
**SUPERVISOR**  
Information Technology  
Kumaraguru College of Technology  
Coimbatore – 641 006

The candidates with University Register Numbers 71202205025, 71202205030, 71202205046 were examined by us in the project viva-voce examination held on

..... 26.04.2006 .....

 INTERNAL EXAMINER

 EXTERNAL EXAMINER

ii

**ACKNOWLEDGEMENT**

We express our sincere thanks to our chairman **Arutchelvar Dr.N. Mahalingam , B.Sc, B.Sc., F.I.E.** and correspondent **Prof. K. Arumugam, B.E., M.S., M.I.E.,** for all their support and ray of strengthening hope extended.

We are immensely grateful to our principal **Dr.K.K. Padmanabhan, B.Sc., M.Tech., Ph.D.,** for his invaluable support to the come outs of this project.

We are extremely thankful to **Dr. G.Gopalsamy, Ph.D.,** Head of the Department, Department of Information Technology for his valuable advice and suggestions throughout this project.

We are indent to express our heartiest thanks to **Prof. K.R.Baskaran, BE., M.S.,** project coordinator who has helped us to perform the project work extremely well.

We are indent to express our heartiest thanks to **Mr. E.A.Vimal, M.E.,** project guide who rendered his valuable guidance and support to perform our project work extremely well.

We are also thankful to all the faculty members of the department of Information Technology, Kumaraguru College of Technology, CBE for their valuable guidance, support and encouragement during the course of our project work.

We express our humble gratitude and thanks to our parents and family members who have supported and helped us to complete the project and our friends, for lending us valuable tips, support and co-operation throughout our project work.

iii

**DECLARATION**

We

Muthukumar.T	71202205025
Prithvirajkumar.D	71202205030
Siby Mathew Kuntharayil	71202205046


hereby declare that the project entitled "WEB BASED BUG TRACKING APPLICATION" is done by us and to the best of our knowledge a similar work has not been submitted to the Anna University or any other institution,for the fulfillment of the requirement of the course study.

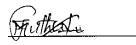
This report is submitted on the partial fulfillment of the requirements for all awards of the Degree of bachelor of Information Technology of Kumaraguru College Of Technology,Coimbatore.

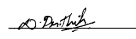
Place: COIMBATORE

Date: 22.04.2006.

Project Guided by

  
[Mr.E.A.Vimal ,M.E.,]

  
Muthukumar.T

  
Prithvirajkumar.D

  
Siby Mathew Kuntharayil

iv

## **ABSTRACT**

Testing process in the software firms has been playing a major role since a long time. Efficient means of testing are adopted widely at a very improved pace. Our project handles one of the tasks in the testing process which is keeping record of the commonly met errors. It also serves as a tool to communicate with the testers and the developers workgroups.

Presently, testers of the organizations are at a different stage which is away from the solution makers. Hence there is a lacking of discussions with the error makers and the solvers. This leads to drastic problems in the future progression of the development process. An efficient means to handle this challenge is the main aim of our project. This web-based business application is a great tool for assigning and tracking issues and tasks during software development and any other projects that involve teams of two or more people. The system is designed specifically with the IT department in mind, where quick access to shared data and history is a requirement, both from an internal organizational perspective, as well as to fulfill the needs of the customers.

Bug track is a Web-based bug tracking, defect tracking, issue tracking, used in a distributed team environment to track software bugs,

v

hardware defects, test cases, or any other issues. It can also be used equally well as a helpdesk customer support, email management system to collect and manage customer feedbacks, incidents, requests, and issues. The application itself can be installed virtually on any web server, whether internal within the organization, or external, hosted by a web hosting company. It is easy to use, but still flexible and adaptive, and can be configured to fit to your organization's unique business process and workflow.

vi

## **TABLE OF CONTENTS**

<b><u>CHAPTER NO.</u></b>	<b><u>TITLE</u></b>	<b><u>PAGE</u></b>			
	<b>ABSTRACT</b>	v			
	<b>LIST OF TABLES</b>	ix			
	<b>LIST OF FIGURES</b>	ix			
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>	<b>4.</b>	<b>SOFTWARE DESIGN</b>	<b>23</b>
	1.1 Existing system and its limitations	2		4.1 Introduction	24
	1.2 Proposed system and its advantages	2		4.2 UML Diagrams	24
<b>2.</b>	<b>IMPLEMENTATION DETAILS</b>	<b>3</b>		4.3 Input/Output Design	25
	2.1 Hardware Requirements	4		4.4 Process Design	28
	2.2 Software Requirements	4		4.5 Database Design	30
<b>3.</b>	<b>LITERATURE SURVEY</b>	<b>5</b>	<b>5.</b>	<b>PRODUCT TESTING</b>	<b>32</b>
	3.1 Apache Tomcat Webserver	6	<b>6.</b>	<b>FUTURE ENHANCEMENTS</b>	<b>35</b>
	3.2 JDBC	10	<b>7.</b>	<b>CONCLUSION</b>	<b>37</b>
	3.3 Java Script	13	<b>8.</b>	<b>APPENDIX</b>	<b>39</b>
				8.1 Sample Code	40
				8.2 Sample Output	57
			<b>9.</b>	<b>REFERENCES</b>	<b>68</b>

vii

viii

**LIST OF TABLES**

<b><u>TABLE NO</u></b>	<b><u>TABLE NAME</u></b>	<b><u>PAGE NO</u></b>
1.	Project	30
2.	Status	30
3.	Bug Record	30
4.	Priority	31
5.	Employees	31

**LIST OF FIGURES**

<b><u>FIGURE NO</u></b>	<b><u>FIGURE NAME</u></b>	<b><u>PAGE NO</u></b>
1.	Tomcat Server	7
2.	Structure of Tomcat	9
3.	JDBC-ODBC Connectivity	11
4.	Usecase Diagram	24

ix

---

---

**1. INTRODUCTION**

---

---

1

**1.0 Introduction**

**1.1 Existing system and its limitations:**

The currently available testing process lacks communication between the testers and the debuggers which is a drastic drawback in the software development overflow. The existing applications comes with limited functionalities. There is no reference to the bugs that are once solved and lacks a shared resource so that the users, testers and the debuggers can have a common perspective of the bugs that are been sorted out. It doesn't have mailing facility for communication between the work groups.

**1.2 Proposed system and its advantages:**

Our bug tracking application is deployed in a distributed environment to record the commonly made errors in the process of software development. All the users in different work groups can have a common access to the bugs and provide suggestions for resolving them. Provision of history lookup in which the solved errors are stored and can be retrieved through queries. It uses a local domain mailing facility in which the messages along with the bug reports can be sent to the respective users. The system serves as a powerful communication tool between the testers and debuggers.

---

---

**2. IMPLEMENTATION DETAILS**

---

---

2

3

## 2.0 Implementation details

### 2.1 Hardware Requirements:

Motherboard	:	Intel 845GVSR.
Cache	:	32MB L1 Cache.
RAM	:	256MB.
Hard Disk	:	40GB.
Processor	:	P4 (32 bit processor)
Video System	:	VGA or SVGA.
Keyboard	:	101 Keys enhanced.

### 2.2 Software Requirements:

Operating system	:	Windows platform
IDE	:	NetBeans IDE 3.6
Server	:	Apache Tomcat 5.0
Client	:	Any web browser
Database	:	Ms Access

4

---

## 3. LITERATURE SURVEY

---

5

## 3.0 Literature survey

### 3.1 Apache Tomcat Web server:

Programs need an environment in which to run within a host computer. Sometimes the operating system is sufficient to provide the environment, but at other times a more sophisticated container is needed. Tomcat is a container that's used to provide an environment for java code running on a web server.

If you are going to be running java code on your web server (either in the form of Servlets or java Server pages), then you will need appropriate software for the purpose. An operating system isn't enough as it won't provide your Java Runtime Environment nor your web server, nor the tools to tie java to the web. You will need a container in which to run your servlets and JSPs and the most commonly used container is tomcat.

#### What is Tomcat?

Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and java Server Pages technologies. The java servlet and java server pages specifications are developed by sun under the java community process.

Tomcat is developed in an open and participatory environment and released under the Apache Software licence. Tomcat is intended to be a collaboration of the best-of-breed developers from around the world.

The Apache software foundation provides support for the Apache community of open source software projects. The Apache projects are characterized by a collaborative, consensus based development process, an open pragmatic software license, and a desire to create high quality software

6

that leads the way in its field.

Amongst the projects that come under the "The Apache banner are the container used for the majority of web sites world wide, Ant, (a build tool which allows the developer excellent control of the compiling and the bundling processes), and Jakarta.

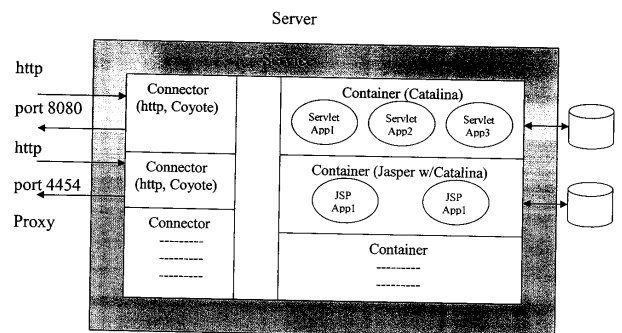


Fig 3.1 (a)

7

### Jakarta

The Jakarta Project creates and maintains open source solutions on the Java platform for distribution to the public at no charge. Jakarta Products are developed by and distributed through various sub projects. Jakarta is the name for the Apache project which deals with the provision of open source additions in Java. More than 20 such additions (known as sub projects) are listed on their website, including Struts and Tomcat.

### Tomcat

Tomcat is a servlet container for the Java Servlets and Java Server Pages. It provides a Java Virtual Machine and associated elements to give a complete Java Run time Environment, and it also provides web server software to make that environment accessible on the Web. Configuration and management tools are also provided, with configuration data largely held in XML.

Its worth noting that Tomcat is much more than just an implementation of Servlets and JSPS, it's the official reference implementation and the standard against which all other suppliers of containers for Servlets and JSPs must measure their products. It means that developers know that works under Tomcat that developers know that if they develop code that works under Tomcat, that code should work under other containers that conform to the standards set.

Tomcat itself has a number of elements to it such as Catalina, Coyote and Jasper.

### Catalina

Catalina is the servlet Container portion of Tomcat.

### Coyote

Coyote is the Web connector

### Jasper

Jasper is the JSP Engine that's used in Tomcat from version 4.1

### The structure of Tomcat

Tomcat runs as a Windows service or a Linux or Unix Daemon, awarding connections (by default) on port 8080. A single instance of Tomcat can provide several services, though this is unusual.

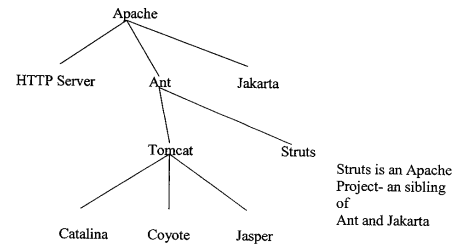


Fig 3.1 (b)

Each Tomcat service will have at least one (and possibly more) connectors, and at least one (and possibly more containers) in which an engine such as Catalina provides a service.

Server, Service, Connector, Container and Engine are all very flexibly configurable, and the default application configuration can be overridden on a per-application basis .The Tomcat Manager is a useful application which runs in one of the standard Tomcat containers and is a used to control loading, reloading and unloading of individual applications or of the engine as a whole

### 3.2 JDBC (Java Data Base Connectivity):

JDBC is a SQL-level API- one that allows you to execute SQL statements and retrieve the results, if any. This API is a set of interfaces and classes designed to perform actions any database.

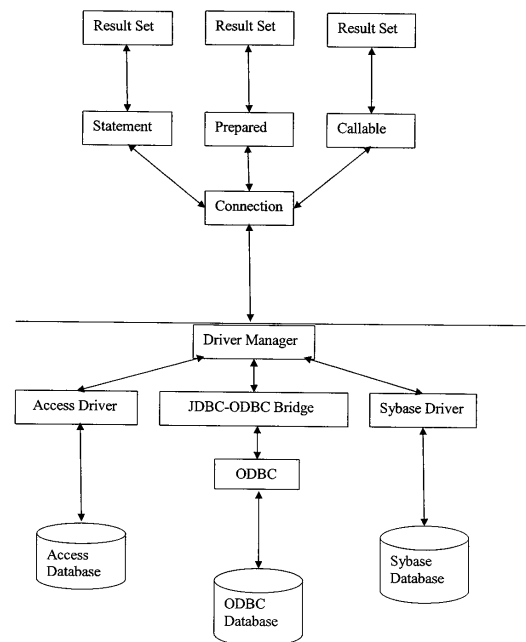


Fig 3.2 JDBC-ODBC connectivity

The JDBC API, found in the java.sql package, contains only a few concrete classes. Much of the API is distributed as database-neutral interface classes that specify behavior without providing any implementation. Third-party vendors provide the actual implementations.

An individual database system is accessed via a specific JDBC driver that implements the java.sql.Driver interface. Drivers exist for nearly all popular RDBMS systems, though few are available for free. Sun bundles a free JDBC-ODBC bridge driver with the JDK to allow access to standard ODBC data sources, such as a Microsoft Access Database. However, Sun advises against using the bridge driver for anything other than development and very limited deployment. Servlet developers in particular should heed this warning any problem in the JDBC-ODBC Bridge driver's native code section can crash the entire server not just your servlets.

JDBC drivers are available for most database platforms, from a number of vendors and in number of different flavors. There are four driver categories:

#### **Type 1: JDBC-ODBC Bridge Driver**

Type 1 use a bridge technology to connect a Java client to a ODBC database service. Sun's JDBC-ODBC bridge is the most common type 1 driver. These drivers are implemented using native code.

#### **Type 2: Native-API Partly-Java Drivers**

Type 2 drivers wrap a thin layer of Java around database-specific native code libraries. For Oracle databases, the active code libraries might be based on the OCI (Oracle Call Interface) libraries, which were originally designed for C/C++ programmers. Because Type 2 drivers are implemented using native code, in some cases they have better performances than their all-

12

Protocol(HTTP)". The computer running the web browser that makes the request is known as the client and the computer satisfying the client's requests is called the Server.

The main reason for using Java Script in this project is its wide spread use and availability. Java Script is very versatile and not just limited to use within a web browser.

#### **Advantages of Java Script**

Java Script is having more advantage over competing scripting languages like VB Script, which are,

##### **1. Capable of running in any Browser**

The java Script is capable of running in any browser like Netscape navigator, Internet Explorer and others, at the other scripting language VB Script can only be run on Internet Explorer and Perl.

##### **2. Managing Administration Tasks**

Java Script is used to automate computer administration tasks. It can also be used for Interacting with the users and getting information from them and validating their actions.

#### **3.4 Ms-Access:**

Access is used as back end. It allows creation of tables and queries in an easier way and it does not place much constraint before the user. Conversion of existing ODBC databases is also a good advantage of using Access. Forms can also be prepared in Access and when converted into MDE files give us a read only design of code and forms.

14

Java counter parts. They add an element of risk, however, because a defect in driver's native code section can crash the entire server.

#### **Type3 : Net-Protocol All-Java Driver**

Type 3 drivers communicate via a generic network protocol to a piece of custom middleware. The middleware component might use any type of driver to provide the actual database access. Web Login's Tengah product line as an example. These drivers are all Java, which makes them useful for applet deployment and safe for servlet deployment.

#### **Type4: Native-Protocol All-Java Driver**

Type 4 drivers are the most direct of the lot. Written entirely in Java. Type 4 drivers understand database specific networking protocols and can access the database directly without any additional software.

A list of currently available JDBC drivers can be found at <http://javasun.com/products/jdbc/jdbc.drivers.html>.

#### **3.3 Java Script:**

Java Script is an Interpreted language, rather than a compiled language. The Java Script code runs inside a web page loaded into a browser. All we need to create is a text editor like windows notepad, and a web browser such as Netscape Navigator or internet Explorer, with which we can view our pages. These browsers come equipped with Java Script interpreters. Basically the job of a web browser is to hold lots of web pages on its hard drive. Then when a browser usually on a different computer requests a web page that is contained on that web server, the web server load it from its own hard drive and then pass the page back to the requesting computer via a special communication protocol called "Hyper Text Transfer

13

Access provides extensive new features to easily use the Internet and develop a World Wide Web (WWW) application. We need a web browser, such as Microsoft Internet Explorer, and a modem, Intranet Connection, or other network connection to access the Internet and take advantage of these features.

Access provides complete documentation for the tables created and also provides the option of viewing the relationships used in creating tables. Foreign Key can be created using the relationship window. Access also comes with wizards under all its options i.e., under tables, Queries, reports, etc... All these wizards help the user to help the user to work in a flexible and interactive environment.

We can use Access to create a World Wide Web Application. For example, we can create a corporate home page, an online magazine or newsletter, a registration system for a trade show, or online product catalog. to create this web application, we output objects to HTML format or use the Publish to the Web Wizard.

We can export reports to static HTML format and you can export datasheets and forms to static and dynamic HTML format. Access created web page for each report page, datasheet, and form you export. Exporting objects to HTML format is useful for creating a simple web application, verifying the format and appearance of an object's output, or adding files to existing web applications.

15

### 3.5 Java Server Pages (JSP):

The java 2 Enterprise edition j2ee has taken the once chaotic task of building an internet presence and transformed it to the point where developers can use java to efficiently create multi-user, server-side applications. Today, the java Enterprise API's have expanded to encompass a number of areas: RMI AND CORBA for remote object handling ,JDBC for database interaction ,JNDI for accessing naming and directory services, Enterprise JavaBeans for creating reusable business components, JMS TM(java Messaging service)for message-oriented middleware, jaxptm for xml processing ,and jtatm (java transaction api) for performing atomic transactions. In addition, j2ee also supporters Servlets ,an extremely popular java substitute for cgi scripts. The combination of these technologies allows programmers to create distributed business solutions for a variety of tasks.

In late 1999,sun Microsystems added a new element to the collection of enterprise java tools: java server pages (JSPs).java server pages are built on top of java Servlets and are designed to increase the efficiency in which programmers, and even nonprogrammers, can create web content .this book is primarily about java Server pages, covering the latest version of this technology,jsp1.2,as well as the related jsp standard tag library (jstl) version 1.0 It also covers other j2ee technologies, such as Servlets and jdbc ,with focus on how to combine them with jsp in the most efficient way.

#### What is Java Server Pages?

Java Server Pages is a technology for developing web pages that include dynamic content .unlike a plain HTML page, which contains static content that always remains the same, a jsp page can change its content

16

#### Why use JSP?

In the early days of the web, the common gateway interface (cgi) was the only tool for developing dynamic web content. However, cgi is not an efficient solution .for every request that comes in, the web server has to create a new operating system process, load an interpreter and a script, execute the script ,and then tear it all down again .this is very taxing for the server and doesn't scale well when the amount of traffic increases. Numerous cgi alternatives and enhancements, such as fastcgi, mod\_perl from apache, SAPI from Netscape, isapi from Microsoft, java Servlets from sun Microsystems, have been created over the years .While these solutions offer better performance and scalability, all these technologies suffer from a common problem: they generate web pages by embedding HTML directly in programming language code .this pushes the creation of dynamic web pages exclusively into the realm of programmers. java Server pages ,however changes all that.

#### Embedding dynamic elements in HTML pages

JSP tackles the problem from the other direction .instead of embedding HTML in programming code; jsp lets you embed special active elements into HTML pages. These elements look similar to HTML elements, but behind the scenes they are actually

Componentized Java programs that the server executes when a user requests the page. Here's a simple JSP page that illustrates this:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<html>
<body bgcolor="white">
```

18

based on any number of variable items, including the identity of the user, the user's browser type, information provided by the user, and selections made by the user. This functionality is key to web applications such as online shopping and employee directories, as well as for personalized and internationalized content .

A JSP page contains standard markup language elements, such as HTML tags, just like a regular web page. However, a jsp page also contains special jsp elements that allow the server to insert dynamic content in the page .jsp elements can be used for a variety of purposes, such as retrieving information from a database or registering user preferences. when a user asks for a jsp page, the server executes the jsp elements, merge the results with the static parts of the page, and sends the dynamically composed page back to the browser.

JSP defines a number of standard elements that are useful for any web application, such as accessing JavaBeans components, passing control between pages and sharing information between requests, pages and users. Programmers can also extend the jsp syntax by implementing application specific elements that perform tasks such as accessing databases and enterprise JavaBeans, sending email, and generating HTML to present application specific data. One such set of commonly needed custom elements is defined by a specification related to the jsp specification: the jsp standard tag library (JSTL) specification. The combination of standard elements and customs elements allows for the creation of powerful web applications

17

```
<jsp:useBean id="clock" class="java.util.Date" />
<c:choose>
<c:when test="{clock.hours < 12}">
<h1>Good morning!</h1>
</c:when>
<c:when test="{clock.hours < 18}">
<h1>Good day!</h1>
</c:when>
<c:otherwise>
<h1>Good evening!</h1>
</c:otherwise>
</c:choose>
Welcome to our site, open 24 hours a day.
</body>
</html>
```

This page inserts a different message to the user based on the time of day: "Good morning!" if the local time is before 12 P.M., "Good day!" if between 12 P.M. and 6 P.M., and "Good evening!" otherwise. When a user asks for this page, the JSP-enabled web server executes the logic represented by the highlighted JSP elements and creates an HTML page that is sent back to the user's browser. For example, if the current time is 8:53 P.M., the resulting page sent from the server to the browser looks like this:

```
<html>
<body bgcolor="white">
<h1>Good evening!</h1>
Welcome to our site, open 24 hours a day.
```

19

</body>  
</html>

In addition to the HTML-like JSP elements, a JSP page can also contain Java code embedded in so-called *scripting elements*. This feature has been part of the JSP specification from the very first version, and it used to be convenient for simple conditional logic. With the introduction of the new JSP Standard Tag Library (JSTL), however, Java code in a page is rarely needed. In addition, embedding too much code in a web page is no better than using HTML elements in a server-side program, and often leads to a web application that is hard to maintain and debug. The examples in this book rarely use scripting elements.

#### Compilation

Another benefit that is important to mention is that a JSP page is always compiled before it's processed by the server. Remember that older technologies such as CGI/Perl require the server to load an interpreter and the target script each time the page is requested. JSP gets around this problem by compiling each JSP page into executable code the first time it's requested (or on demand), and invoking the resulting code directly on all subsequent requests. When coupled with a persistent Java virtual machine on a JSP-enabled web server, this allows the server to handle JSP pages much faster.

#### Integration with enterprise Java APIs

Finally, because Java Server Pages are built on top of the Java Servlets API, JSP has access to all the powerful Enterprise Java APIs, including:

- JDBC
- Remote Method Invocation (RMI) and OMG CORBA support
- JNDI (Java Naming and Directory Interface)
- Enterprise JavaBeans (EJB)
- JMS (Java Message Service)
- JTA (Java Transaction API)
- JAXP (Java API for XML Processing)
- Java Mail

This means that you can easily integrate Java Server Pages with your existing Java Enterprise solutions.

#### Advantages of JSP

JSP 1.2 combines the most important features found in the alternatives:

- JSP supports both scripting- and element-based dynamic content and allows programmers to develop custom tag libraries to satisfy application-specific needs.
- JSP pages are compiled for efficient server processing.
- JSP pages can be used in combination with Servlets that handle the business logic, the model supported by Java servlet template engines. In addition, JSP has a couple of unique advantages that make it stand out from the crowd:
  - JSP is a specification, not a product. This means vendors can compete with different implementations, leading to better performance and quality.

It also leads to a less obvious advantage, namely that when so many companies have invested time and money in the technology, chances are it

20

21

will be around for a long time, with reasonable assurances that new versions will be backward-compatible; with a proprietary technology, this is not always a given. JSP is an integral part of J2EE, a complete platform for enterprise class applications.

This means that JSP can play a part in the simplest applications to the most complex and demanding.

---

---

## 4. SOFTWARE DESIGN

---

---

22

23



## 4.0 Software Design

### 4.1 Introduction:

A Software Design is a model of a real world system that has many participating entities and relationships. This design is used in a number of different ways. It acts as a basis for detailed implementation

### 4.2 UML Diagrams:

UML is a notation that is used for developing software blue prints. It is an object oriented modeling technique.

#### USECASE DIAGRAM

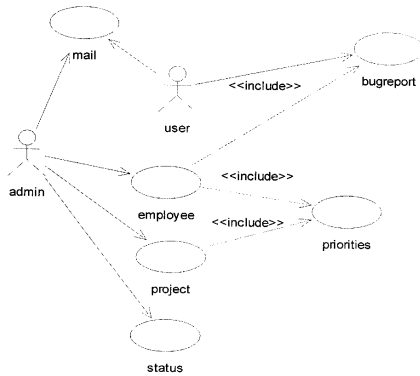


Fig 4.2

## 4.3 Input/Output Design:

### QUERY PAGE

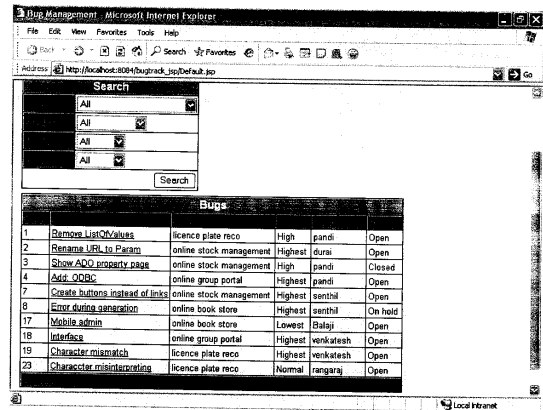


Fig 4.3 (a)

### BUG REPORT

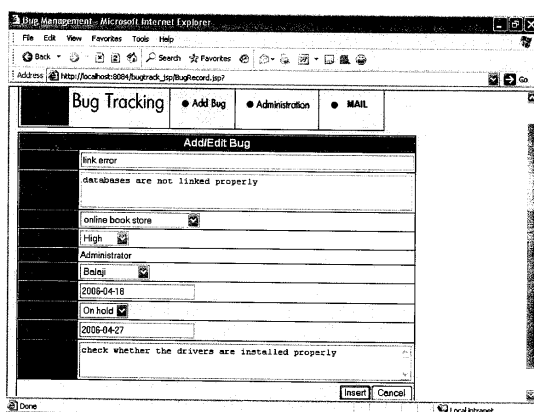


Fig 4.3 (b)

### MAIL

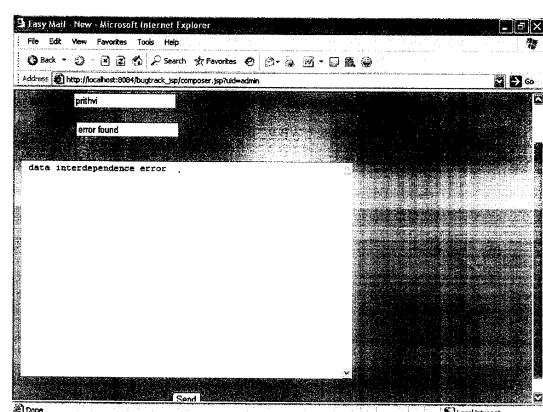


Fig 4.3 (c)

4.4 Process Design:

SEQUENCE DIAGRAM

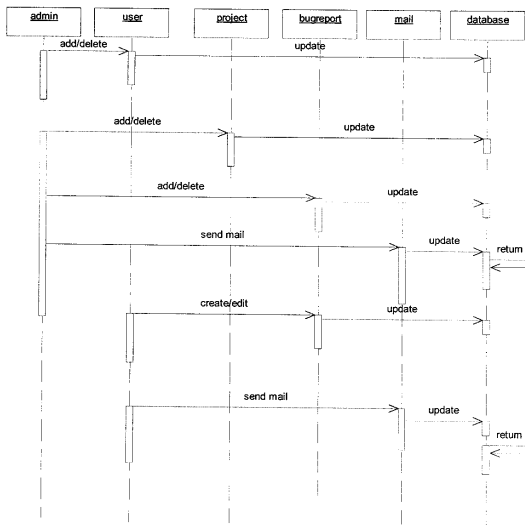


Fig 4.4 (a)

COLLABORATION DIAGRAM

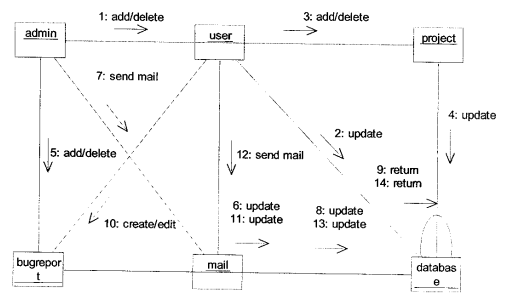


Fig 4.4 (b)

4.5 Database Design:

Field	Description
project_id	The key field present to identify a project bug which becomes a foreign key field
project_name	Name of the project module in which the error has happened
manager_id	Name given to the person for receiving mail

Status	
Field	Description
status_id	The unique identifier for a status to set for the Errors
status	Description of the status levels

Errors	
Field	Description
date_resolved	The due date for the Error to be solved
project_id	Uniquely identifies a project with a n error
bug_id	Distinguishes a bug
priority_id	Lists the priority for the bug
status	Sets status
bug_name	Name of the Error found
bug_desc	Details of the Error
resolution	The steps to solve the Error
assigned_by	One who finds the Error
assigned_to	The person to whom it is assigned to solve
date_assigned	The date on which the Error was found

Priority	
Field	Description
priority_id	The identifier used to be set for the Errors found
priority_desc	Description for the priority level

Employee	
Field	Description
employee_id	Differentiates an employee which is a primary key field
employee_name	Name of the person who is employed
login	The unique id assigned for the person
pass	The password that restricts the entry for the employees
email	Email id to be identified among the employees
security_level	The permission level given to the entry of the person

## 5.0 Product testing

Testing is done to detect the errors in the software .This implies not only to the coding phase but to uncover errors introduced in all the previous phases .The following are the types of tests that were performed.

### Unit testing:

Each and every module is tested separately to check if its intended functionality is met .Some unit testing performed are,

- Checking for proper connectivity between the client and web server module
- Verification of the mailing ensuring correctness of the transmission along with message and report

### Integration testing:

It is the testing performed to detect errors on interconnection between modules. Here, all the modules pertaining to the client system and server system are combined to form the client and server applications respectively and tested to ensure that they work in synchronization and without interference from each other.

---

---

## 5. PRODUCT TESTING

---

---

32

33

P-1637

### System testing:

The system is tested against the system requirements to see if all the requirements are met and if the system performs as per the specified requirements .The system is tested as a whole to check for its functionality.

### Validation testing:

This test is done to check for the validity of the entered input. The input is provided for admin, employee's user id and password. Invalid characters and symbols are recognized and properly handled.

34

35

---

---

## 6. FUTURE ENHANCEMENTS

---

---

## 6.0 Future Enhancements

Our project "WEB BASED BUG TRACKING APPLICATION" is designed in such a way that future enhancements can be made in an easy manner. The project is quite flexible and can be customized according to the user's requirements.

One possible enhancement to the project is adding of testing tools. Testing tools like Rational robot, Rational test manager can be easily embedded in our software. This aids in testing being performed in the same software independent of others.

Another useful enhancement is the automation of mailing facility. In this automated mailing, mail will automatically be sent to the respective user with the reports and the message immediately after the bugs are tracked.

36

---

---

## 7. CONCLUSION

---

---

37

## 7.0 Conclusion:

The complete design and development of the system is presented in this dissertation. The system has user friendly features and can be configured to fit to your organization's unique business process and workflow.

The programming techniques used in the design of the system provide a scope for further expansion and implementation of any changes, which may occur in the future. The system has been tested with many client side browsers and they provide satisfactory performance.

The main aim behind the development of this system is to facilitate the IT community to keep record of the commonly met errors in the software development process and provision of an easy and efficient mean of communication between the tester and developer.

38

---

---

## 8. APPENDIX

---

---

39

## 8.0 Appendix

### 8.1 Sample code:

#### Administration.jsp

```
<%@ include file="Common.jsp" %><%!  
static final String sFileName = "Administration.jsp";  
static final String PageBODY = "bgcolor=#F3F2E6" text=#000000"  
link=#800000" vlink=#000080" alink=#0000FF";  
static final String FormTABLE = "border=#1" cellspacing=#0" cellpadding=#2"  
bordercolorlight=#000000" bordercolordark=#FFFFFF";  
static final String FormHeaderTD = "align=#center" bgcolor=#669999";  
static final String FormHeaderFONT = "style=#font-size: 12pt; color: #FFFFFF; font-  
family: Arial, Tahoma, Verdana, Helvetica; font-weight: bold";  
static final String FieldCaptionTD = "bgcolor=#B3B300";  
static final String FieldCaptionFONT = "style=#font-size: 10pt; color: #000000; font-  
family: Arial, Tahoma, Verdana, Helvetica";  
static final String DataTD = "bgcolor=#F0F0F0";  
static final String DataFONT = "style=#font-size: 10pt; color: #000000; font-family:  
Arial, Tahoma, Verdana, Helvetica";  
static final String ColumnFONT = "style=#font-size: 10pt; color: #000000; font-family:  
Arial, Tahoma, Verdana, Helvetica; font-weight: bold";  
static final String ColumnTD = "bgcolor=#B3B300";  
%><%  
String cSec = checkSecurity(3, session, response, request);  
if ("sendRedirect".equals(cSec) ) return;  
boolean bDebug = false;  
String sAction = getParam( request, "FormAction");  
String sForm = getParam( request, "FormName");  
String sAdministrationErr = "";  
String sLoginErr = "";
```

40

```
<td valign="top">  
<% Administration_Show(request, response, session, out, sAdministrationErr, sForm,  
sAction, conn, stat); %>  
</td>  
</tr>  
</table>  
<table>  
<tr>  
<td valign="top">  
<% Login_Show(request, response, session, out, sLoginErr, sForm, sAction, conn, stat);  
%>  
</td>  
</tr>  
</table>  
<center>&nbsp;  </center>  
</body>  
</html>  
<%%>  
<%  
if ( stat != null ) stat.close();  
if ( conn != null ) conn.close();  
%>  
<%!  
void Administration_Show (javax.servlet.http.HttpServletRequest request,  
javax.servlet.http.HttpServletResponse response, javax.servlet.http.HttpSession session,  
javax.servlet.jsp.JspWriter out, String sAdministrationErr, String sForm, String sAction,  
java.sql.Connection conn, java.sql.Statement stat) throws java.io.IOException {  
try {  
out.println(" <table border=#1" cellspacing=#0" cellpadding=#2"  
bordercolorlight=#000000" bordercolordark=#FFFFFF">");
```

42

```
java.sql.Connection conn = null;  
java.sql.Statement stat = null;  
String sErr = loadDriver();  
conn = cn();  
stat = conn.createStatement();  
if ( ! sErr.equals("") ) {  
try {  
out.println(sErr);  
}  
catch (Exception e) {}  
}  
if ( sForm.equals("Login") ) {  
sLoginErr = LoginAction(request, response, session, out, sAction, sForm, conn, stat);  
if ( "sendRedirect".equals(sLoginErr) return;  
}  
%>  
<html>  
<head>  
<title>Administration</title>  
<meta name="GENERATOR" content="YesSoftware CodeCharge v.1.2.0 / JSP.ccp  
build 05/21/2001"/>  
<meta http-equiv="pragma" content="no-cache"/>  
<meta http-equiv="expires" content="0"/>  
<meta http-equiv="cache-control" content="no-cache"/>  
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">  
</head>  
<body bgcolor=#F3F2E6" text=#000000" link=#800000" vlink=#000080"  
alink=#0000FF">  
<jsp:include page="Header.jsp" flush="true"/>  
<table>  
<tr>
```

41

```
out.println(" <tr>\n <td align=#center" bgcolor=#669999"><font  
style=#font-size: 12pt; color: #FFFFFF; font-family: Arial, Tahoma, Verdana, Helvetica;  
font-weight: bold">Administration</font></td>\n </tr>");  
out.print(" <tr>");  
// Set URLs  
String fldemployees = "EmployeeList.jsp";  
String fldpriorities = "PriorityList.jsp";  
String fldprojects = "ProjectList.jsp";  
String fldstatus = "StatusList.jsp";  
// Show fields  
out.print("\n <td bgcolor=#F0F0F0"><a href=#"+fldemployees+"><font  
style=#font-size: 10pt; color: #000000; font-family: Arial, Tahoma, Verdana,  
Helvetica">Employees</font></a></td>\n\n </tr>\n\n <tr>");  
out.print("\n <td bgcolor=#F0F0F0"><a href=#"+fldpriorities+"><font  
style=#font-size: 10pt; color: #000000; font-family: Arial, Tahoma, Verdana,  
Helvetica">Priorities</font></a></td>\n\n </tr>\n\n <tr>");  
out.print("\n <td bgcolor=#F0F0F0"><a href=#"+fldprojects+"><font  
style=#font-size: 10pt; color: #000000; font-family: Arial, Tahoma, Verdana,  
Helvetica">Projects</font></a></td>\n\n </tr>\n\n <tr>");  
out.print("\n <td bgcolor=#F0F0F0"><a href=#"+fldstatus+"><font  
style=#font-size: 10pt; color: #000000; font-family: Arial, Tahoma, Verdana,  
Helvetica">Statuses</font></a></td>");  
out.println("\n </tr>\n </table>");  
}  
catch (Exception e) { out.println(e.toString()); }  
}  
String LoginAction(javax.servlet.http.HttpServletRequest request,  
javax.servlet.http.HttpServletResponse response, javax.servlet.http.HttpSession session,  
javax.servlet.jsp.JspWriter out, String sAction, String sForm, java.sql.Connection conn,  
java.sql.Statement stat) throws java.io.IOException {  
String sLoginErr = "";
```

43

```

try {
    final int iLoginAction = 1;
    final int iLogoutAction = 2;
    String transitParams = "";
    String sQueryString = "";
    String sPage = "";
    String sSQL="";
    int iAction = 0;
    if ( sAction.equals("login") ) iAction = iLoginAction;
    if ( sAction.equals("logout") ) iAction = iLogoutAction;
    switch (iAction) {
        case iLoginAction: {
            // Login action
            String sLogin = getParam( request, "Login");
            String sPassword = getParam( request, "Password");
            java.sql.ResultSet rs = null;
            rs = opens( stat, "select employee_id, security_level from employees where login
            =" + toSQL(sLogin, adText) + " and pass=" + toSQL(sPassword, adText));
            if ( rs.next() ) {
                // Login and password passed
                session.setAttribute("UserID", rs.getString(1));
                session.setAttribute("UserRights", rs.getString(2));
                sQueryString = getParam( request, "querystring");
                sPage = getParam( request, "ret_page");
                if ( ! sPage.equals(request.getRequestURI()) && ! "".equals(sPage) ) {
                    try {
                        if ( stat != null ) stat.close();
                        if ( conn != null ) conn.close();
                    }
                    catch ( java.sql.SQLException ignore ) {}
                    response.sendRedirect(sPage + "?" + sQueryString);
                }
            }
        }
    }
}

```

44

```

style="font-size: 12pt; color: #FFFFFF; font-family: Arial, Tahoma, Verdana, Helvetica;
font-weight: bold">Login</font></td></tr>";
if ( sLoginErr.compareTo("") != 0 ) {
    out.println(" <tr>\n <td colspan="2" bgcolor="#F0F0F0"><font
style="font-size: 10pt; color: #000000; font-family: Arial, Tahoma, Verdana,
Helvetica">"+sLoginErr+"</font></td>\n </tr>");
}
sLoginErr="";
out.println(" <form action="+sFileName+" method="POST">");
out.println(" <input type="hidden" name="FormName" value="Login">");
if ( session.getAttribute("UserID") == null || ((String)
session.getAttribute("UserID")).compareTo("") == 0 ) {
    // User did not login
    out.println(" <tr>\n <td bgcolor="#B3B300"><font style="font-size: 10pt;
color: #000000; font-family: Arial, Tahoma, Verdana, Helvetica">Login</font></td><td
bgcolor="#F0F0F0"><input type="text" name="Login" maxlength="50"
value="+toHTML(getParam( request, "Login"))+"></td>\n </tr>");
    out.println(" <tr>\n <td bgcolor="#B3B300"><font style="font-size: 10pt;
color: #000000; font-family: Arial, Tahoma, Verdana,
Helvetica">Password</font></td><td bgcolor="#F0F0F0"><input type="password"
name="Password" maxlength="50"></td>\n </tr>");
    out.println(" <tr>\n <td colspan="2"><input type="hidden"
name="FormAction" value="login"><input type="submit" value="Login">");
    out.println("<input type="hidden" name="ret_page" value="+sPage+"><input
type="hidden" name="querystring" value="+sQueryString+"></td>\n </form>\n
</tr>");
}
else {
    // User logged in
    String sUserID = dLookup( stat, "employees", "login", "employee_id =" +
session.getAttribute("UserID"));

```

46

```

return "sendRedirect";
}
}
else sLoginErr = "Login or Password is incorrect.";
rs.close();
break;
}
case iLogoutAction: {
    // Logout action
    session.setAttribute("UserID", "");
    session.setAttribute("UserRights", "");
    break;
}
}
}
catch (Exception e) { out.println(e.toString()); }
return (sLoginErr);
}
void Login_Show(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response, javax.servlet.http.HttpSession session,
javax.servlet.jsp.JspWriter out, String sLoginErr, String sForm, String sAction,
java.sql.Connection conn, java.sql.Statement stat) throws java.io.IOException {
    try {
        String sSQL="";
        String transitParams = "";
        String sQueryString = getParam( request, "querystring");
        String sPage = getParam( request, "ret_page");
        out.println(" <table border="1" cellspacing="0" cellpadding="2"
bordercolorlight="#000000" bordercolordark="#FFFFFF" border="1">");
out.println(" <tr>\n <td align="center" bgcolor="#669999" colspan="2"><font

```

45

```

out.println(" <tr><td bgcolor="#F0F0F0"><font style="font-size: 10pt; color:
#000000; font-family: Arial, Tahoma, Verdana,
Helvetica">"+sUserID+"&nbsp;&nbsp;&nbsp;"+</font><input type="hidden"
name="FormAction" value="logout"><input type="submit" value="Logout"/>");
    out.println("<input type="hidden" name="ret_page" value="+sPage+"><input
type="hidden" name="querystring" value="+sQueryString+">");
    out.println("</td>\n </form>\n </tr>");
}
out.println(" </table>");
}
catch (Exception e) { out.println(e.toString()); }
}
%>

```

#### Mail Composer

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>
<%@page import="javax.swing.*"%>
<%@page import="java.util.*"%>
<%!
String uid;
%>
<%
try{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    System.out.println("Driver Registered in matching");
    Connection
con=DriverManager.getConnection("jdbc:odbc:bugtrack","","");
    System.out.println("Connection Established matching");
    Statement st=con.createStatement();

```

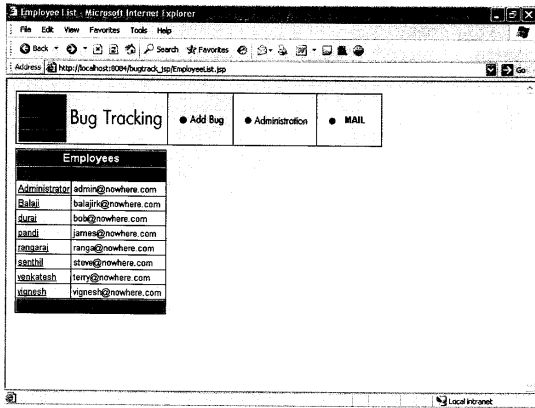
47



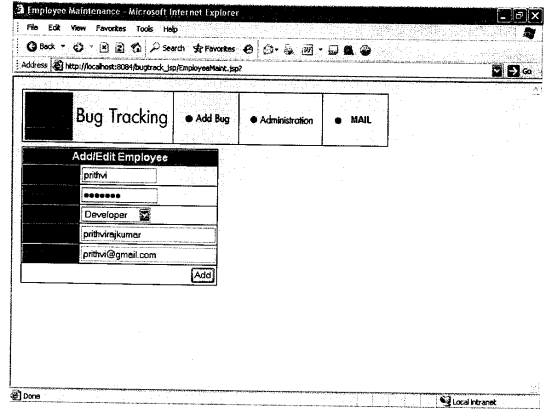






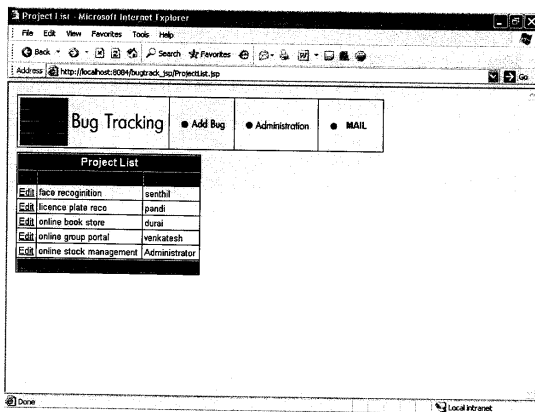


60

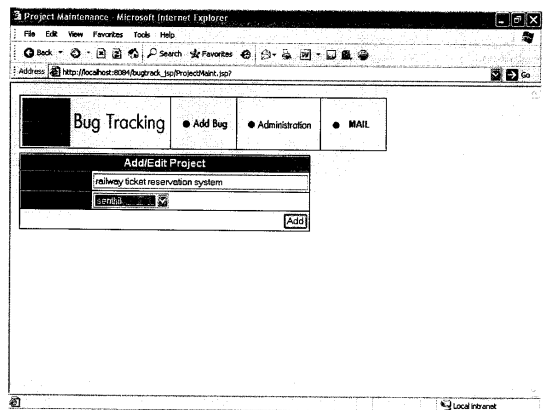


61

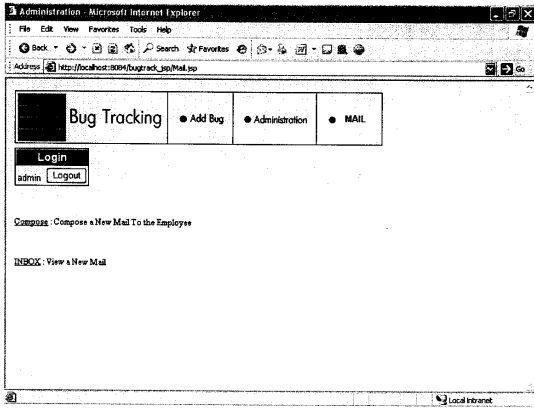
P-1637



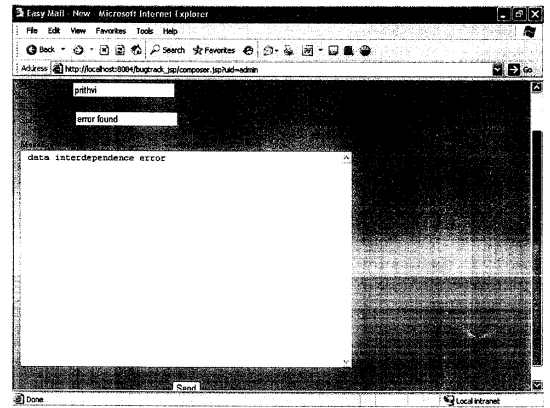
62



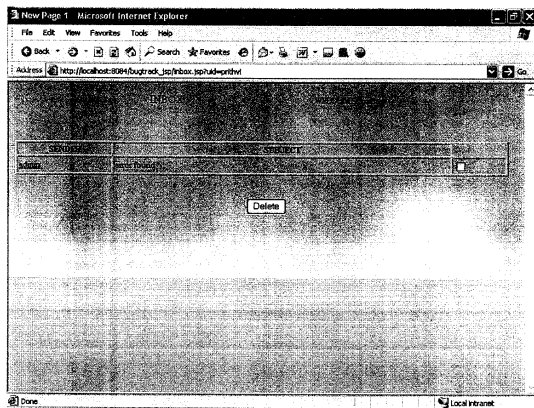
63



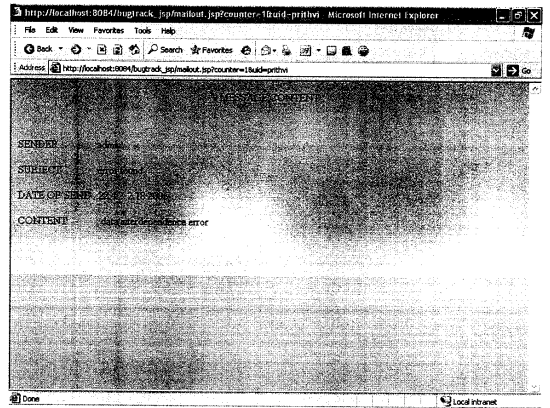
64



65



66



67

## 9.0 References

### 9.1 Books:

- 1) "The Complete Reference Java2", Herbert Schildt, Edition 5
- 2) "Java Script Bible", Danny Goodman, Michael Moorison, Edition 5
- 3) "Database Programming with JDBC and Java", George Reese
- 4) "Java Servlet programming", Jason Hunter, Edition 2

### 9.2 Web Sites:

1. <http://www.java.sun.com/>
2. <http://www.servlets.com/>
3. <http://www.javascript-coder.com/>
4. <http://www.webreference.com/>

---

---

## 9. REFERENCES

---

---