A comprehensive study on Online Crime Investigation System.

Hemanth Erabelli

Abstract-- The main abstract of this system isto provide the solution for the bug posted by the customer within no time. The manager should be able to organize the entire bug efficiently and assign the bug to the concerned technical person. The system should provide the solution to the customer in possible proposed date. The objective of the proposed system is to build that provides the facility to the customer to post their complaints about the particular crime or criminal.

Index terms—Administrator, Crime investigation, Customer module, Department module, Java VM, JSP, JDBC, Whitebox testing.

1.1 Motivation

The motivation behind the system is to solve the customer problems easily and quickly and provides the facility to post their complaints about the particular crime or criminal. The system should keep track of complaints and within no time, the customer should get the solution for his problem.

1.2 Problem Definition

There is a communication gap between the customer and department. People generally don't know the sites of all the departments to post their complaints. Customer face problem while posting their complaints. As they have to post their complaints to different departments individually. There is no surety of quick response from the department. Few departments may not be interested in responding to the customers. The proposed system should provide the facilities for the following modules.

Administrator
Customer
Department

In administrator module, once an administrator logs on to the system he/she can register the departments and criminals. The Administrator can update the details of anyone on their demand. He analyzes the performance of each individual.

Author details :
Hemanth Erabelli, Student, B.tech 3rd year, Computer Science and Engineering, Anurag Group of Institutions, Hyderabad, India.
Mail id :hemanth.erabelli@gmail.com

In customer module, once a customer logs on to the system, he can post a complaint to the department. The customer can view the status to the complaint posted by him. They can edit their personnel details.

In department module, when department person logs on to the system, he can view the complaints posted by the customers. The department person can then change the status of the crime.

2.ANALYSIS

2.1 Introduction

Requirements analysis is done in order to understand the problem the software system is to solve. The problem could be automating an existing manual process, developing a new automated system, or a combination of the two.The emphasis in requirements analysis is on identifying whatis needed from the system, not how the system will achieve its goals. This task is complicated by the fact that there are often at least two parties involved in software development-a client and a developer. The developer usually does not understand the client's problem domain and the client does not understand the issues involved in the system software systems developed by the developers. Hence causes a communication gap between them.
This communication gap is bridged during the analysis. This analysis phase ends with a document describing all the requirements called as SRS (Software Requirements Specification).

2.2 Existing System

Now days, almost everyone is using computers. Some people use it for writing letters, sending mails etc. Few people use it for developing applications and few people use it for entertainment purpose. Like this computers are useful for almost everyone in something or the other way.When people are working with computers, they may

or may not be aware of computers and more importantly when they are using the software products of different vendors, they may face some problems. If such is the case, the users may not get a proper solution immediately and sometimes it would also take a very long time to get a solution for his/her problem.

For getting proper solutions, the users have to contact the concerned organization that has provided the software and tell their problem. The concerned organization will then provide a solution for his/her problem within a period of time.

2.3 Proposed System

Online Crime Investigation System interacts mainly with three entities i.e. Customer, Administrator and Department person. The proposed system is mainly used to provide the facility to customers to post their complaints. The main user of the system is the Administrator.

2.4 Software Requirement Specification

Operating System - Microsoft Windows 2000/XP.
Front End Tool - Servlets and Java Server Pages.
Scripting language - HTML, JavaScript.
RDBMS - Oracle 8i.
Application Server - Apache Tomcat Web Server .

2.4.1 Purpose

The purpose of "**ONLINE CRIME INVESTIGATION SYSTEM"** isto provide the solution for the bug posted by the customer within no time. The administrator should be able to organize the entire bug efficiently and assign the bug to the concerned technical person. The system should provide the solution to the customer in possible proposed date.

2.4.2 Scope

Presently we have to login to the system online and post the complaints. We can further develop the system by using "Voice Recognition System". So this project has got scope to get developed in the near future. The scope of the project exists throughout the life of the product development, testing and implementation.

2.4.3 Overall Description

Online Crime Investigation System is mainly used to provide the facility to the customer to post their complaints about the particular crime or criminal. The administrator should keep track of complaints should solve the solution for the problem posted by the customer quickly,

2.4.4 External Interface Requirements

RAM - 64 MB
Hard Disk - 20 GB
Processor - Intel Pentium 500MHz
Floppy Disk - 1.44 MB
Monitor - Color Monitor (256 colors)
Mouse - 3 button scroll mouse
Keyboard -Multimedia Keyboard

3. SYSTEM ARCHITECTURE

3.1 Modules

The proposed system consists of the following modules.
Administrator Module
Customer Module
Department Module

3.2 Module description

Administration

1. **New department-** The aim of this module is to add a new department so that the customers can post their complaints related to that particular department.
2. **New criminal-**The aim of this module is to add new criminal information to the department.
3. **Viewing Criminal information -** The aim of this module is to provide an interface to the customer to view the criminal information
4. **Viewing Department information-** The aim of this module is to provide the departments information to the customers.

3.2.1 Customers

1. **Customer Registration** - The aim of this module is to provide all the details of the customers so that by providing all the details, he can get registered with this web site. After registering, the customer will be getting a unique id. Using that unique id the customer can logon to this web site**.**
2. **Customer Login** - The aim of this module is that the customers with their login ID and password can now logon to the Systems and use the service.
3. **Posting Complaints -** The aim of this module is to provide all the details of the complaint by the customer and post the complete details to the manager.
4. **Viewing Complaint Status -** The aim of this module is to provide an interface to the customer to view the status of his posted complaints.

5. **Viewing department information -** The aim of this module is to provide an interface to the customer to view the department information.
6. **Viewing criminal information -** The aim of this module is to provide an interface to the customer to view the criminal information

3.2.2 Department

1. **Department Registration** - The aim of this module is to provide all the details of the managers so that by providing all the details, the Administrator can register the manager and after registering each manager provided with a unique ID.
2. **Department Login** - The aim of this module is that the department persons with their login ID and password can now logon to the System and use the services provided to him.
3. **Viewing complaints -** The aim of this module is to provide an interface to the manager to view the complaints along with the details posted by the customers.
4. **Viewing customer information -** The aim of this module is to provide an interface to the department to View the complaints posted by the customers.
5. **Viewing criminal information -** The aim of this module is to provide an interface to the department to View the criminal information.

5. DESIGN

4.1 Introduction

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate. Explore potential designs, and validate the architectural design of the software.

4.1.1 Goals of UML

The primary goals in the design of the UML were:

1. Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Supporthigher-level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

UML is specifically constructed through two different domains they are:

1. UML Analysis modeling, which focuses on the user model and structural model views of the system.UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

2. Use case Diagrams represent the functionality of the system from a user's point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

4.2 UML diagrams:

The UnifiedModeling Language (UML) is a general-purpose modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system. UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.

4.2.1 Use Case Diagram

Use case Diagram shows a set of uses cases and actors with their relationships. It addresses the static use case view of a system. Use case diagrams gives a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions are interacted. These are important in organizing and modeling the behavior of a system. A more detailed description might characterize a use case as:

1. A pattern of behavior the system exhibits
1. A sequence of related transactions performed by an actor and the system
1. Delivering something of value to the actor.

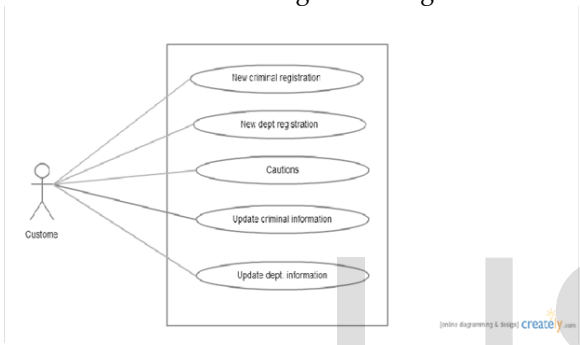


Fig:4.2.1.1 Usecase Diagram for Customer

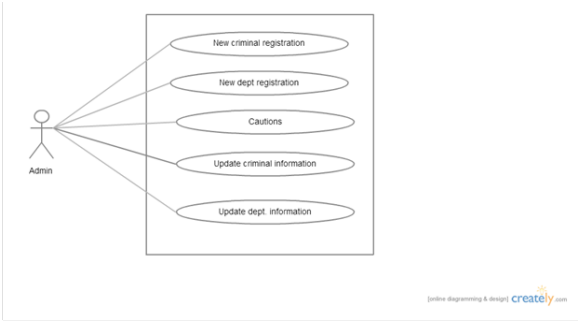


Fig:4.2.1.2 Usecase Diagram for Adminstrator

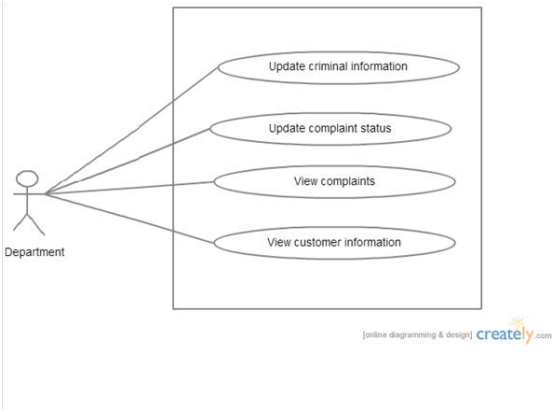


Fig.4.2.1.3 Usecase Diagram for Department

4.2.2 Class Diagram

A class diagram is a picture for describing generic descriptions of possible systems. It is the main building block of any object oriented solution. It shows the classes in a system, attributes and operations of each class and the relationship between each class. Class diagrams contain classes and object diagrams contain objects, but it is possible to mix classes and objects when dealing with various kinds of metadata, so the separation is not rigid.

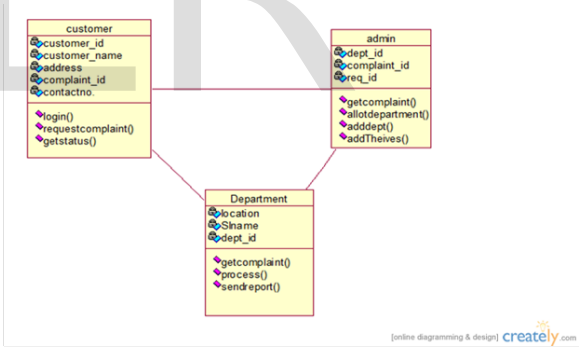


Fig: 4.2.2 Class Diagram

4.2.3 Sequence Diagram

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces. Sequence diagram shows how object interact with each other and the order those interactions occur.

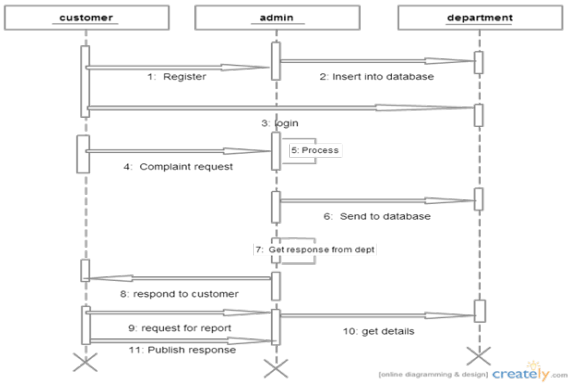


Fig: 4.2.3 Sequence Diagram

4.2.4 Collaboration Diagram

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

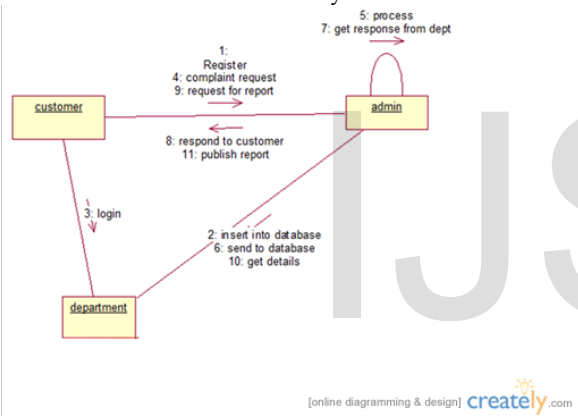


Fig: 4.2.4 Collaboration Diagram

4.2.5 Activity Diagram

Activity diagrams provide a way to model the workflow of a business process. Activity diagrams are very similar to a flowchart because you can model a workflow from activity to activity. An activity diagram is typically used for modeling the sequence of activities in a process, whereas a state chart is better suited to model the discrete stages of an object's lifetime.

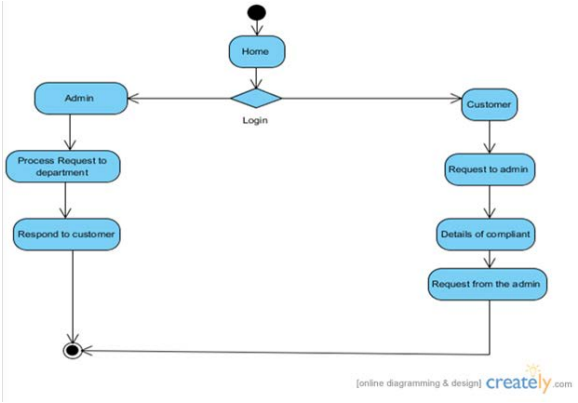


Fig: 4.2.5 Activity Diagram

4.2.6 Component Diagram

Component diagrams provide a physical view of the current model. A component diagram shows the organizations and dependencies among software components, including source code components, binary code components, and executable components. These diagrams also show the externally-visible behavior of the components by displaying the interfaces of the components.

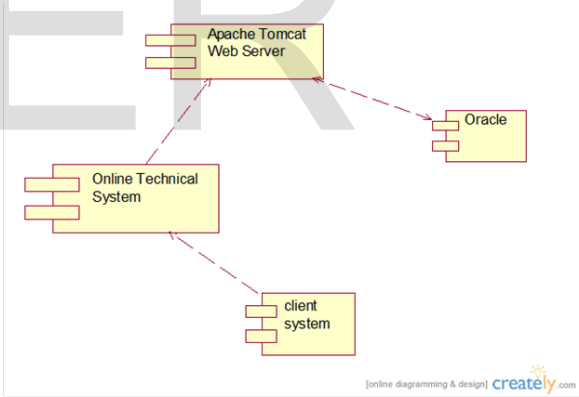


Fig:4.2.6 Component Diagram

4.2.7 Deployment Diagram

A deployment diagram is a visual representation of devices and processors. The deployment diagram class exposes properties and methods that allow you to add, retrieve and delete devices and processors in a deployment diagram.

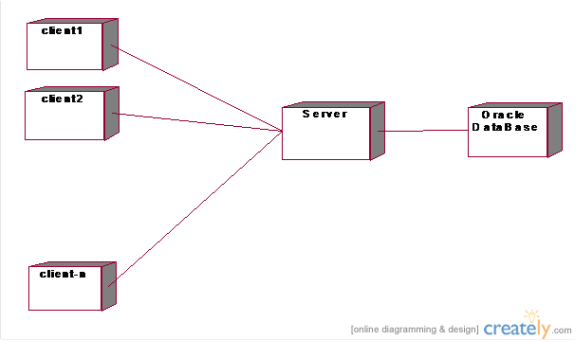


Fig: 4.2.7 Deployment Diagram

6. TECHNOLOGIES USED

5.1.1 Introduction to HTML

Hyper Text Markup Language is a structural markup language used to create and format a web document. A markup language such as HTML is simply a collection of codes, called Elements that are used to indicate the structure and format of a document. A user agent, usually a web browser that renders the document, interprets the meaning of these codes to figure how to structure or display a document. HTML is not invention but it is an improved version of Standard Generalized Markup Language (SGML).

Importance of HTML:

HTML can be used to display any type of document on the host computer, which can be geographical at a different location.

1. It is a versatile language and can be used on any platform or desktop.

2. The appearance of a Web page is important, and HTML provides tags to make the document look attractive. Using graphics, fonts, different sizes, color, etc. can enhance the presentation of the document.

Functionality of HTML in the project:

As we know this is purely web-based project. This helps to embed Java Server Pages within the page using some simple tags.

1.Used to design the forms.

2.User can communicate easily with server.

5.1.2 JAVA

Initially the language was called as "oak" but it was renamed as "Java" in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.Java is a programmer's language.Java is cohesive and consistent.Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.Finally, Java is to Internet programming where C was to system programming.

1. **Importance of Java to the Internet**

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause and by doing so, has opened the door to an exciting new form of program called the Applet.

2. **Java can be used to create two types of programs**

Applications and Applets: An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Java's ability to create Applets makes it important. An Applet is an application designed to be transmitted over the Internet and executed by a Java – compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

Features of Java Security

When you use a Java-compatible Web browser, you can safely download Java applets without fear of virus infection or malicious intent.

Portability

For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed .As you will see, the same mechanism that helps ensure security also helps create portability. Indeed, Java's solution to these two problems is both elegant and efficient.

The Byte code

The key that allows the Java to solve the security and portability problems is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM

is an interpreter for byte code.Translating a Java program into byte code helps makes it much easier to run a program in a wide variety of environments. The reason is, once the run-time package exists for a given system, any Java program can run on it.Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-flycompilation of byte code into native code. Sun has just completed its Just In Time (JIT) compiler for byte code. When the JIT compiler is a part of JVM, it compiles byte code into executable code in real time, on a piece-by-piece, demand basis. It is not possible to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. The JIT compiles code, as it is needed, during execution.

Java Virtual Machine (JVM)

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that's has been generated by the compiler will not corrupt the machine that it's loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code.

Overall Description

Fig5.1.3.1 Compilation of Java Program

Picture showing the development process of JAVA Program.Java programming uses to produce byte codes and executes them. The first box indicates that the Java source code is located in a. Java file that is processed with a Java compiler called javac. The Java compiler produces a file called a. class file, which contains the byte code. The .Class file is then loaded across the network or loaded locally on your machine into the execution environment is the Java virtual machine, which interprets and executes the byte code.

Java Architecture

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

Compilation of code

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

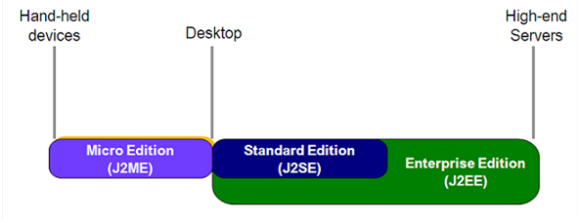
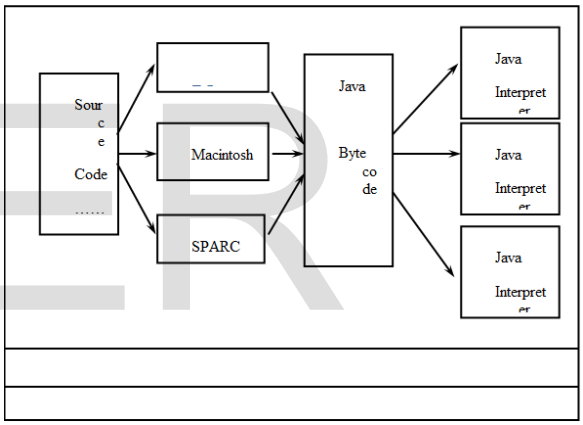


Fig.5.1.3.2 Compiling and interpreting Java Source Code



During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be a Intel Pentium Windows 95 or SunSARC station running Solaris or Apple Macintosh running system and all could receive code from any computer through Internet and run the Applets.

Features of Java

Simple

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ programmer, learning Java will be even easier. Because Java inherits the C/C++ syntax and many of the object oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small

number of clearly defined ways to accomplish a given task.

Object-Oriented

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

Robust

The multi-platform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and run time.Java virtually eliminates the problems of memory management and de-allocation, which is completely automatic. In a well-written Java program, all run time errors can –and should –be managed by your program.

Java Collections

A collection — sometimes called a container — is simply an object that groups multiple elements into a single unit. Collections are used to store, retrieve, manipulate, and communicate aggregate data. Typically, they represent data items that form a natural group, such as a poker hand (a collection of cards), a mail folder (a collection of letters), or a telephone directory (a mapping of names to phone numbers). If you've used the Java programming language — or just about any other programming language — you're already familiar with collections. Collection implementations in earlier (pre-1.2) versions of the Java platform included Vector, Hashtable, and array. However, those earlier versions did not contain a collections framework. A collections framework is a unified architecture for representing and manipulating collections. All collections frameworks contain the following:

Interfaces: These are abstract data types that represent collections. Interfaces allow collections to be manipulated independently of the details of their representation. In object-oriented languages, interfaces generally form a hierarchy.

Implementations: These are the concrete implementations of the collection interfaces. In essence, they are reusable data structures.

Algorithms: These are the methods that perform useful computations, such as searching and sorting, on objects that implement collection interfaces. The algorithms are said to be polymorphic: that is, the same method can be used on many different implementations of the appropriate collection interface. In essence, algorithms are reusable functionality.

Benefits of the Java Collections Framework

The Java Collections Framework provides the following benefits:

1. **Reduces programming effort:** By providing useful data structures and algorithms, the Collections Framework frees you to concentrate on the important parts of your program rather than on the low-level "plumbing" required to make it work. By facilitating interoperability among unrelated APIs, the Java Collections Framework frees you from writing adapter objects or conversion code to connect APIs.
2. **Increases program speed and quality:** This Collections Framework provides high performance, high-quality implementations of useful data structures and algorithms. The various implementations of each interface are interchangeable, so programs can be easily tuned by switching collection implementations. Because you're freed from the drudgery of writing your own data structures, you'll have more time to devote to improving programs' quality and performance.
3. **Allows interoperability among unrelated APIs:** The collection interfaces are the vernacular by which APIs pass collections back and forth. If my network administration API furnishes a collection of node names and if your GUI toolkit expects a collection of column headings, our APIs will interoperate seamlessly, even though they were written independently.
4. **Reduces effort to learn and to use new APIs:** Many APIs naturally take collections on input and furnish them as output. In the past, each such API had a small sub-API devoted to manipulating its collections. There was little consistency among these ad hoc collections sub-APIs, so you had to learn each one from scratch, and it was easy to make mistakes when using them. With the advent of standard collection interfaces, the problem went away.
5. **Reduces effort to design new APIs:** This is the flip side of the previous advantage. Designers and implementers don't have to reinvent the wheel each time they create an API that relies on collections; instead, they can use standard collection interfaces.

5.1.3 Introduction of JSP:

Java Server Pages (JSP) lets you separate the dynamic part of your pages from the static HTML. You simply write the regular HTML in the normal manner, using whatever Web-page-building tools you normally use. You then enclose the code or the dynamic parts in special tags, most of which start with "<%" and end with %>". we normally give the file a .jsp extension.

There are three main types of JSP constructs that you embed in a page: scripting elements, directives, and actions. Scripting elements let you specify Java code that will become part of the resultant servlet, directives let you control the overall structure of the servlet, and actions let you specify existing components that should be used, and otherwise control the behavior of the JSP engine.

5.1.3.1 JSP Scripting Elements

JSP scripting elements let you insert Java code into the servlet that will be generated from the current JSP page. There are three forms:

Expressions of the form <%= expression %> that are evaluated and inserted into the output,

Scriptlets of the form <% code %> that are inserted into the servlet's service method, and

Declarations of the form <%! code %> that are inserted into the body of the servlet class, outside of any existing methods.

5.1.3.1.1 JSP Expressions

A JSP expression is used to insert Java values directly into the output. It has the following form:

<%= Java Expression %>

The Java expression is evaluated, converted to a string, and inserted in the page. This evaluation is performed at run-time (when the page is requested), and thus has full access to information about the request.

5.1.3.1.2 JSP Scriptlets

If you want to do something more complex than insert a simple expression, JSP scriptlets let you insert arbitrary code into the servlet method that will be built to generate the page. Scriptlets have the following form:

<% Java Code %>

Scriptlets have access to the same automatically defined variables as expressions.

5.1.3.1.3 JSP Declarations

A JSP declaration lets you define methods or fields that get inserted into the main body of the servlet class (outside of the service method processing the request). It has the following form:

<%! Java Code %>

Since declarations do not generate any output, they are normally used in conjunction with JSP expressions or scriptlets.

5.1.3.2 JSP Directives

A JSP directive affects the overall structure of the servlet class. It usually has the following form:

<%@ directive attribute="value" %>

However, you can also combine multiple attribute settings for a single directive, as follows:

```
<%@ directive attribute1="value1"
      attribute2="value2"
    attributeN="valueN" %>
```

To simplify code in JSP expressions and scriptlets, you are supplied with eight automatically defined variables, sometimes called implicit objects. The available variables are request, response, out, session, application, config, page Context, and page. Details for each are given below.

1. Request

This is the HttpServletRequest associated with the request, and the request parameters (via get Parameter), the request type (GET, POST, HEAD, etc.), and the incoming HTTP headers (cookies, Referrer, etc.). Request is allowed to be a subclass of ServletRequest other than HttpServletRequest, if the protocol in the request is something other than HTTP.

2. Response

This is the HttpServletResponse associated with the response to the client. Note that, since the output stream is buffered, it is legal to set HTTP status codes and response headers, even though this is not permitted in regular servlets once any output has been sent to the client.

3. Out

This is the PrintWriter used to send output to the client. In order to make the response object useful, this is a buffered version of PrintWriter called JspWriter.

4. Session

This is the HttpSession object associated with the request. Recall that sessions are created automatically, so this variable is bound even if there was no incoming session reference.

5. Application

This is the ServletContext as obtained via getServletConfig ().getContext ().

6. Config

This is the ServletConfig object for this page.

7. PageContext

JSP introduced a new class called PageContext to encapsulate use of server-specific features like higher performance JspWriters. The idea is that, if you access them through this class rather than directly, your code will still run on "regular" servlet/JSP engines.

8. Page

This is simply a synonym for this, and is not very useful in Java. It was created as a placeholder for the time when the scripting language could be something other than Java.

5.1.3.3 Actions

JSP actions use constructs in XML syntax to control the behavior of the servlet engine. You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin. Available actions include:

jsp:include - Include a file at the time the page is requested.

jsp:useBean - Find or instantiate a JavaBean.
jsp:setProperty - Set the property of a JavaBean.
jsp:getProperty - Insert the property of a JavaBean into the output.
jsp:forward - Forward the requester to a new page.

5.1.3 Introduction to ORACLE

A modern Relational Database Management System can perform a wide array of tasks. It acts as a transparent interface between the physical storage and the logical presentation of data .In practice; it provides a set of more or less flexible and sophisticated tools for handling information.

A **DBMS** must also be secure from unauthorized access and provide efficient solutions for failure recovery. The **ORACLE** server provides efficient solutions for the database features.

Features of ORACLE

Oracle provides a good security by providing the capability to make users with their own passwords with different privileges.

- It includes the provision to define table level or column level constraints
- It is easy to retrieve data needed by giving proper SQL commands.
- We can bunch the SQL commands and make the bunch executed once by means of PL/SQL (procedures).
- It provides to store our procedures as a library functions by means of Stored Procedures.
- It includes the facility for corresponding updates by means of triggers.
- It gives the capability to access record by means of Cursors.
- It supports client handler, which is a section of code written specifically to deal with errors.
-

5.1.4 Introduction to JDBC

JDBC is a java API for executing SQL statements. JDBC is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the Java programming language. JDBC provides standard API tool/database developers and makes it possible to write database application using a pure Java API. Using JDBC, it is easy to send SQL statements to virtually any relation database.

One can write a single program using the JDBC API, and the program will be able to send SQL statements to the appropriate database. The combination of Java and JDBC lets a programmer to write it once and run it anywhere.

Java being robust, secure, easy to understand and automatically downloadable on a network, is an excellent language basis for database applications. A programmer can write or update once, put it on the server, and everybody has access to the latest version.

JDBC makes it possible to do three things:

1. Establishes a connection with a database.
1. Send SQL statements.
1. Process the results.

JDBC Driver Types

The JDBC drivers that we are aware of at this time fit into one of four categories:

1.JDBC-ODBC bridge plus ODBC driver
2.Native-API partly-Java driver
3.JDBC-Net pure Java driver
4.Native-protocol pure Java driver

The JDBC-ODBC Bridge is a JDBC driver, which implements JDBC operations by translating them into ODBC operations. To ODBC it appears as a normal application program. The Bridge implements JDBC for any database for which an ODBC driver is available. The Bridge is implemented as the Sun.jdbc.odbc Java package and contains a native library used to access ODBC. The Bridge is a joint development of Innersole and Java Soft.

JDBC connectivity

The JDBC provides database-independent connectivity between the J2EE platform and a wide range of tabular data sources. JDBC technology allows an Application Compsonent Provider to:

1. Perform connection and authentication to a database server
1. Manager transactions
1. Move SQL statements to a database engine for preprocessing and execution
1. Execute stored procedures
1. Inspect and modify the results from Select statements

5.2 Sample code

```
   package CMST;
import java.sql.*;
public class ConnectionBean
{
private Connection connection;
private Statement statement;
private static final String
     driver="sun.jdbc.odbc.JdbcOdbcDriver";
private static final String  dbURL="jdbc:odbc:ocr";
private static final String login="ocr";
private static final String password="ocr";
public ConnectionBean()
{ try{
Class.forName(driver);
System.out.println("ok");
connection=DriverManager.getConnection(dbURL,login,pa
     ssword);
System.out.println("ok");
statement=connection.createStatement();}
catch (ClassNotFoundException e)
{System.err.println("ConnectionBean: driver unavailable");
connection = null;}
catch (SQLException e)
{System.err.println("ConnectionBean: driver not loaded");
connection = null;}}
public Connection getConnection()
{return connection;}
public void commit() throws SQLException
{connection.commit();}
public void rollback() throws SQLException
{connection.rollback();}
public void setAutoCommit(booleanautoCommit) throws
SQLException
{connection.setAutoCommit(autoCommit );}
public ResultSetexecuteQuery(String sql) throws
SQLException
{return statement.executeQuery(sql);}
public intexecuteUpdate(String sql) throws SQLException
{return statement.executeUpdate(sql);}
public booleanexecute(String sql) throws SQLException
{return statement.execute(sql);
}protected void finalize()
{
try{connection.close();
}catch (SQLException e){}}}
```

1) Admin Login

```
public class AdminLogin extends HttpServlet
{
public void init(ServletConfig config) throws
ServletException
{ super.init(config); }
 public void destroy() { }
protected void service(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
java.io.IOException
{ RequestDispatcherrd=null;
  response.setContentType("text/html");
  try  {
ConnectionBeanCBean=new ConnectionBean();
   Connection con=CBean.getConnection();
            String loginName=request.getParameter("cLid");
    System.out.println("loginname" + loginName);
  String adminPwd=request.getParameter("cPwd");
    System.out.println("adminPwd" + adminPwd);
    System.out.println("In Admin Login");
    ResultSetadminrs_found=CBean.executeQuery("select   *
from admin_master where admin_id='" + loginName + "'
and admin_pwd='" + adminPwd + "' ");
    if(adminrs_found.next())
            {
  System.out.println("Record found");
  rd=request.getRequestDispatcher("AdminHome.html");
    }else
    {  rd=request.getRequestDispatcher("ALogin.jsp");
            }
    rd.forward(request, response);
}
```

2) Admin Registration

```
public class AdminRegister extends HttpServlet
{
public void init(ServletConfig config) throws
ServletException
{ super.init(config); }
 public void destroy() { }
 protected void service(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
java.io.IOException
 {

 RequestDispatcherrd=request.getRequestDispatcher("Inser
ted.jsp");
  response.setContentType("text/html");
  String aid=request.getParameter("aid");
  String apwd=request.getParameter("apwd");
```

```
String AdminAdd=request.getParameter("AdminAdd");
intcno=Integer.parseInt(request.getParameter("cno"));
String EmailId=request.getParameter("EmailId");
System.out.println(EmailId);
try {
ConnectionBeanCBean=new ConnectionBean();
        Connection con=CBean.getConnection();
 int inserted=CBean.executeUpdate("insert into admin_mastervalues('" + aid + "','" + apwd + "','" + AdminAdd + "'," + cno + ",'" + EmailId + "') ");
  if(inserted==1)
        {
    rd.forward(request, response);
        }
```

3) Customer Login

```
public class CustLogin extends HttpServlet
{
public void init(ServletConfig config) throws ServletException
{ super.init(config); }
public void destroy()
{ }
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, java.io.IOException
{

RequestDispatcherrd=request.getRequestDispatcher("DetailsInserted.jsp");
response.setContentType("text/html");
String crno=request.getParameter("crno");
String cLid=request.getParameter("cLid");
String cPwd=request.getParameter("cPwd");
System.out.println("In CustLogin");
try
{
 ConnectionBeanCBean=new ConnectionBean();
        Connection con=CBean.getConnection();
 int inserted=CBean.executeUpdate("insert into customer_loginvalues('" + cLid + "','" + cPwd + "'," + crno + ",sysdate) ");
  if(inserted==1)
        {
   rd.forward(request, response);
```

4) Customer Registration

```
public class CustRegister extends HttpServlet
{
public void init(ServletConfig config) throws ServletException
{ super.init(config); }
public void destroy() { }
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, java.io.IOException
{
RequestDispatcherrd=request.getRequestDispatcher("Inserted.jsp");
response.setContentType("text/html");
PrintWriter out=response.getWriter();
String cregno=request.getParameter("regno");
String CustName=request.getParameter("CustName");
String CustAdd=request.getParameter("CustAdd");
String Code=request.getParameter("Code");
String CustCno=request.getParameter("CustCno");
String lid=request.getParameter("txtlogin");
String pwd=request.getParameter("txtpwd");
try {
ConnectionBeanCBean=new ConnectionBean();
        Connection con=CBean.getConnection();
 intcust_inserted=CBean.executeUpdate("insert into customer_mastervalues('" + cregno + "','" + CustName + "','" + CustAdd + "','"+ Code + "','" + CustCno + "')
");
 intcust_login_inserted=CBean.executeUpdate("insert into login_mastervalues('" + lid + "','" + pwd + "','C' ,'" + cregno + "') ");
 if(cust_inserted==1 &&cust_login_inserted==1)
 { rd.forward(request, response);
        }
```

5) Department Login

```
public class Login extends HttpServlet
{
public void init(ServletConfig config) throws ServletException
{ super.init(config); }
public void destroy()
{ }
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, java.io.IOException
{
RequestDispatcherrd=null;
response.setContentType("text/html");
try {
 ConnectionBeanCBean=new ConnectionBean();
        Connection con=CBean.getConnection();
        String loginName=request.getParameter("cLid");
        String cPwd=request.getParameter("cPwd");
 ResultSetcustrs=CBean.executeQuery("select reg_id,login_type from login_master where user_id='" + loginName + "' and user_pwd='" + cPwd + "' ");
 if(custrs.next())
```

```
        {
                String uid=custrs.getString(1);
   request.setAttribute("uid",uid);
   String login_type=custrs.getString(2);
   if(login_type.equals("C"))
   {
                        System.out.println("Is  Valid");
   rd=request.getRequestDispatcher("CustomerHome.jsp");
   }else
   {
    rd=request.getRequestDispatcher("Login.jsp");
                } }else
        {
                rd=request.getRequestDispatcher("Login.jsp"); }
   rd.forward(request, response);
```

6) Thieve Registration

```
public class ThieveRegistration extends HttpServlet
{
 String noy="0";
  String nom="0";
  String nod="0";
  String ws="";
 public void init(ServletConfig config) throws ServletException
  { super.init(config);}
  public void destroy()
  { }
  protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, java.io.IOException
  {

 RequestDispatcherrd=request.getRequestDispatcher("AdminInserted.jsp");
  response.setContentType("text/html");
  PrintWriter out=response.getWriter();
  int  thid=Integer.parseInt(request.getParameter("thid"));
  String thName=request.getParameter("thName");
  String tloc=request.getParameter("tloc");
  noy=request.getParameter("noy");
  nom=request.getParameter("nom");
  nod=request.getParameter("nod");
System.out.println("values id "+nom);
 if(noy=="")
  { noy="0";  }
 if(nom=="")
  { nom="0"; }
  if(nod=="")
  { nod="0";  }
  ws=noy+","+nom + "," +nod;
  String tth=request.getParameter("tth");
  String image=request.getParameter("photo");
  try {
ConnectionBeanCBean=new ConnectionBean();
        Connection con=CBean.getConnection();
PreparedStatementpstmt=con.prepareStatement("insert into thieves values(?,?,?,?,?,?)");
   pstmt.setInt(1,thid);
   pstmt.setString(2,thName);
   pstmt.setString(3,tloc);
   pstmt.setString(4,ws);
   pstmt.setString(5,tth);
        File f=new File(image);
FileInputStreamfis=new FileInputStream(f);
        pstmt.setBinaryStream(6,fis,(int) f.length());
   int rows=pstmt.executeUpdate();
// intth_inserted=CBean.executeUpdate("insert into thieves values(" + thid + ",'" + thName + "','" + tloc + "','" + noy + "','" + tth + "') ");
   if(rows==1)
        {
rd.forward(request, response);
        }
```

7) Update Thieve Details

```
public class UpdateThieve extends HttpServlet
{
public void init(ServletConfig config) throws ServletException
{  super.init(config);
}
public void destroy() { }
 protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, java.io.IOException
 {
PrintWriter out=response.getWriter();
  String noy="0";
  String nom="0";
  String nod="0";
  String ws="";

//RequestDispatcherrd=request.getRequestDispatcher("DetailsInserted.jsp");
  response.setContentType("text/html");
  String tid=request.getParameter("tid");
  String ThName=request.getParameter("ThName");
  String loc=request.getParameter("loc");
noy=request.getParameter("noy");
 nom=request.getParameter("nom");
  nod=request.getParameter("nod");
String tht=request.getParameter("tht");
  String image=request.getParameter("photo");
  System.out.println("Image si"+image);
```

```
 if(noy=="")
  { noy="0"; }
  if(nom=="")
  { nom="0"; }
   if(nod=="")
  {  nod="0";}
ws=noy+","+nom + "," +nod;
  try {
  ConnectionBeanCBean=new ConnectionBean();
            Connection con=CBean.getConnection();
    PreparedStatementpstmt=null;
if(image.equals(""))
  { System.out.println("Null");
    pstmt=con.prepareStatement("update thieves set
th_name=?,location=?,no_of_years=?,type_th=? where
thieve_id='"+ tid + "'");
    pstmt.setString(1,ThName);
                pstmt.setString(2,loc);
                pstmt.setString(3,ws);
                pstmt.setString(4,tht);
  } else
  {
   System.out.println("Not Null");
   pstmt=con.prepareStatement("update thieves set
th_name=?,location=?,no_of_years=?,type_th=?,photo=?
where thieve_id='"+ tid + "'");
    pstmt.setString(1,ThName);
    pstmt.setString(2,loc);
    pstmt.setString(3,ws);
    pstmt.setString(4,tht);
File f=new File(image);
            FileInputStreamfis=new FileInputStream(f);
            pstmt.setBinaryStream(5,fis,(int) f.length());
}
  int rows=pstmt.executeUpdate();
//int inserted=CBean.executeUpdate("update thieves set
th_name='"+ ThName + "',location='" + loc + "',no_of_years='"
+ noy + "',type_th='" + tht + "' where thieve_id='"+ tid + "' ");
if(rows==1) {
out.println("<html><script>function
check(){self.close();opener.location='DThievesInformation.js
p';          }</script><center><h1><b>Details          Updated
Successfully</b></h1></center></html>");
    out.println("<center><input type=button value='Close'
onclick='return check()'></center>");
//     rd.forward(request, response);
  } else
            {  System.out.println("Not Updated"); }
```

6. TESTING

6.1 Introduction

Testing is the major quality control measure employed for software development.Its basic function is to detect errors in the software. During requirement analysis and design, the output is document that is usually textual and non-textual. After the coding phase, computer programs are available that can be executed for testing purpose. This implies that testing has to uncover errors introduced during coding phases. Thus, the goal of testing is to cover requirement, design, or coding errors in the program. The starting point of testing is unit testing. In this a module is tested separately and often performed by the programmer himself simultaneously while coding the module.

The purpose is to exercise the different parts of the module code to detect coding errors. After this the modules are gradually integrated into subsystems, which are then integrated themselves too eventually forming the entire system. During integration of module integration testing is performed. The goal of this is to detect designing errors, while focusing the interconnection between modules. After the system was put together, system testing is performed. Here the system is tested against the system requirements to see if all requirements were met and the system performs as specified by the requirements. Finally accepting testing is performed to demonstrate to the client for the operation of the system.

Testing is an extremely critical and time-consuming activity. It requires proper planning of the overall testing process. Frequently the testing process starts with the test plan. This plan identifies all testing related activities that must be performed and specifies the schedule, allocates the resources, and specifies guidelines for testing. The test plan specifies conditions that should be tested; different units to be tested, and the manner in which the module will be integrated together. Then for different test unit, a test case specification document is produced, which lists all the different test cases, together with the expected outputs, that will be used for testing. During the testing of the unit the specified test cases are executed and the actual results are compared with the expected outputs. The final output of the testing phase is the testing report and the error report, or a set of such reports. Each test report contains a set of test cases and the result of executing the code with the test cases. The error report describes the errors encountered and the action taken to remove the error.

Error Messages

The term error is used in two different ways. Errors refer to the discrepancy between computed and observed values. That is error refers to the difference between the actual output of the software and the correct output. In this interpretation, error essentially is a measure of the difference between the actual and the ideal. Error is also used to refer to human action that results in the software containing a defect or a fault.

Testing Techniques

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test cases and the output is evaluated to determine whether the program is performing as expected. In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

Unit Testing

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

Each Module can be Tested using the following two Strategies:

Black Box Testing

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been used to find errors in the following categories:

1. Incorrect or missing functions
1. Interface errors
1. Errors in data structure or external database access
1. Performance errors
1. Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

White Box Testing

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been used to generate the test cases in the following cases:

Guarantee that all independent paths have been executed.

Execute all logical decisions on their true and false sides.

Execute all loops at their boundaries and within their operational

Execute internal data structures to ensure their validity.

Integration Testing

Integration testing ensures that software and subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

System Testing

Involves in-house testing of the entire system before delivery to the user. Its aim is to satisfy the user the system meets all requirements of the client's specifications.

Acceptance Testing

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

Validation Testing

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

Compiling Test

It was a good idea to do our stress testing early on, because it gave us time to fix some of the unexpected deadlocks and stability problems that only occurred when components were exposed to very high transaction volumes.

Execution Test

This program was successfully loaded and executed. Because of good programming there was no execution error.

Output Test

The successful output screens are placed in the output screens section.

Testing is the process of finding differences between the expected behavior specified by system models and the observed behavior of the system.

6.2 Test Cases

6.2.1 Testcases for Customer

S.no	Testcase	Input	Output	Remarks
1	Customer registration	Enter the customer details and enter login id and password	a. Registration successful if details are correct b.Registration fails if login id and password are same	Success Failure
2	Customer Login	Login id and password	a. Success if both are correct b.fails if both are same	Success Failure
3	Complaint Registration	Customer needs to enter details of the complaint	Details registered Succesfully	Success
4	Criminals Information	Select the Criminal Information	Displays all the criminals information	Success
5	Complaint Status	Select complaint status	Displays the status of the complaint	Success
6	Department	Select department	Displays this department information	Success
7	Change Profile	Select the profile to update	Successful if details are updated.	Success

6.2.2 Test cases for Adminstrator

S.no	Testcase	Input	Output	Remarks
1	Login	Enter username and password as admin	a. Login successful if login details are correct. b. Login failed if login details are not correct.	Success Failure
2	New Department	Select new department to enter details.	Details entered succesfully	Success
3	All Departments	Select all departments	Displays the list of all departments	Success
4	Criminals Information	Select the Criminal Information	Displays all the criminals information	Success
5	New Criminal Information	Criminal information is added	Details entered correctly	Success

6.2.3 Testcases for Department

S.no	Testcase	Input	Output	Remarks
1	Login	Enter the username and password of the department	a. Login successful if login details are correct. b. Login failed if login details are not correct.	Success Failure
2	Requested Complaints	Select the Requested complaints that are posted by the customer and change the status of complaints	Displays the list of all complaints	Success
3	Criminal Information	Select Criminal Information	Displays the list of all criminal information	Success
4	Customer Information	Select Customer Information	Displays the list of all Customers who registered with thie system	Success

6.3 Screen Shots

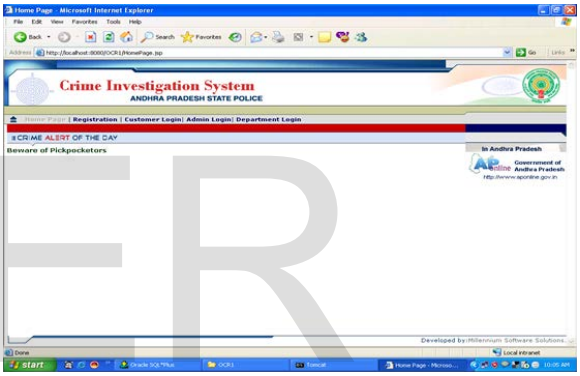


Fig.6.3.1 Home Page of Online Crime Investigation System

Administrator Module

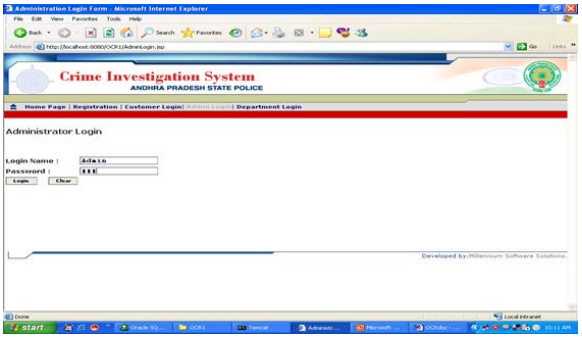


Fig.6.3.2 Administrator Login Screen

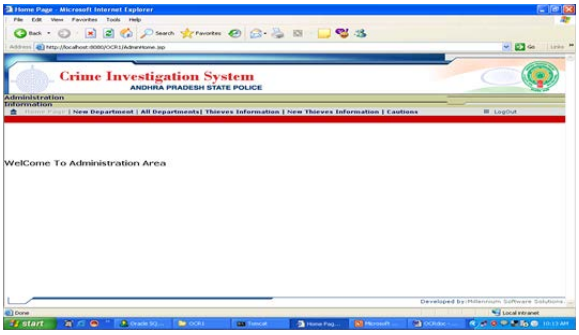


Fig.:6.3.3 Administrator HomePage

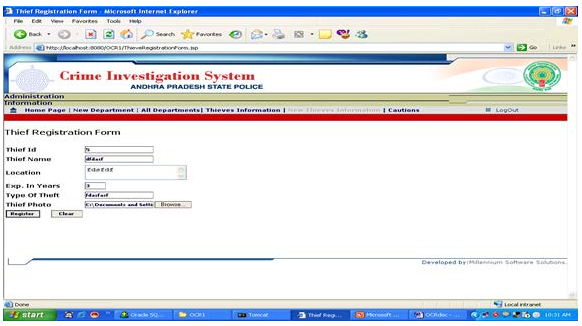


Fig.6.3.6 Thief Registration Screen

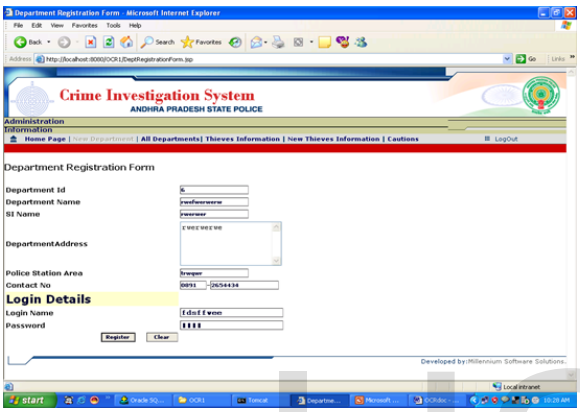


Fig.:6.3.4 Department Registration Form

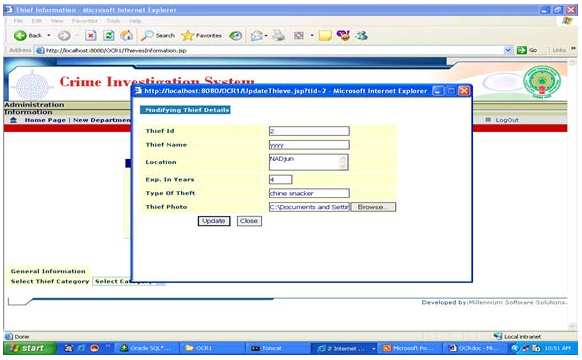


Fig.6.3.7 Updating thieves Information

Fig.6.3.5 Updating Department Details Screen

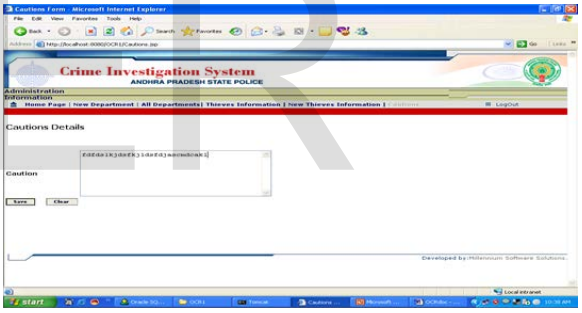


Fig.6.3.8 Caution Registration form

Department Module

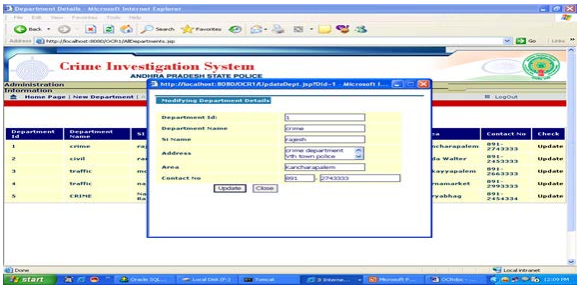


Fig.6.3.6 Modifying Department Details Screen

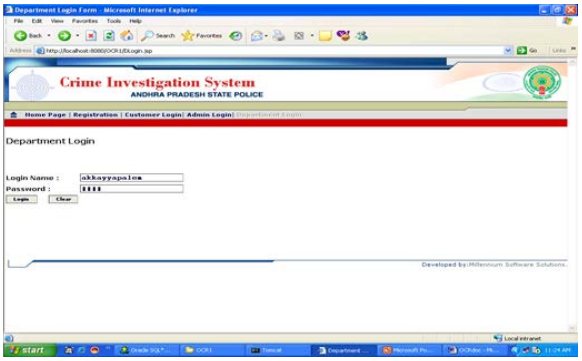


Fig.6.3.9 Department Login Screen

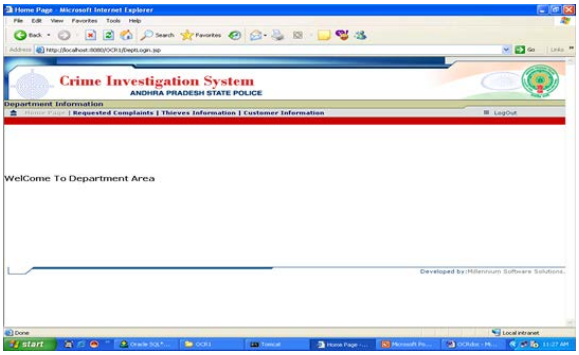


Fig.6.3.10 Department Home Page

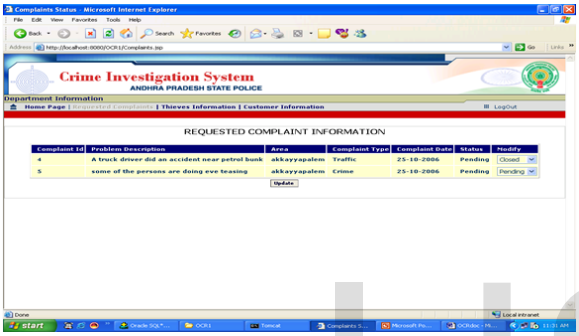


Fig.6.3.11 Requested Complaints

Fig.6.3.12 Thief Details Screen

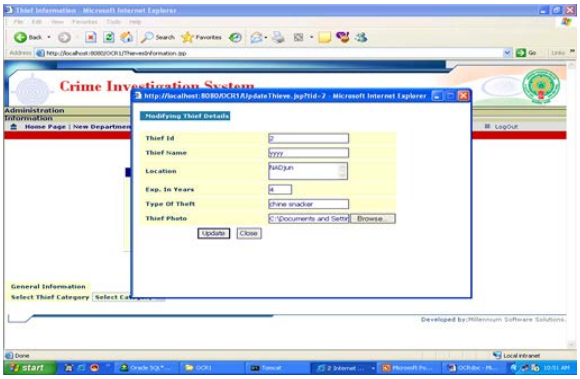


Fig.6.3.13 Modifying Thief Details Screen

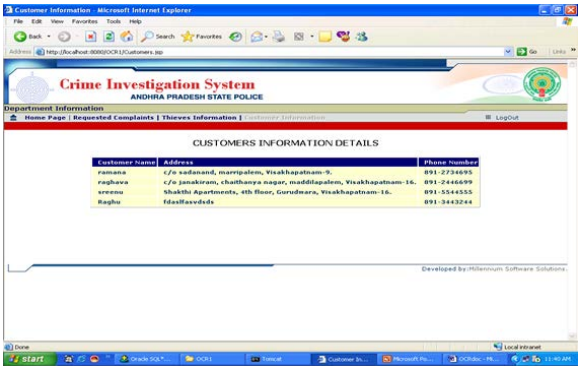


Fig.6.3.14 Customer Information

Customer Module

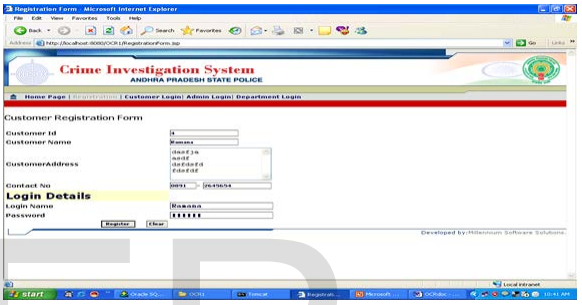


Fig.6.3.15:Customer registration Screen

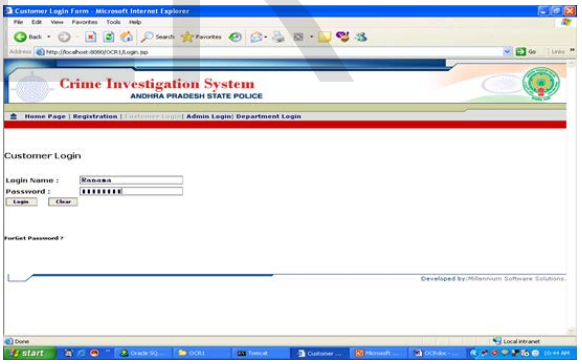


Fig.6.3.16 Customer Login Screen

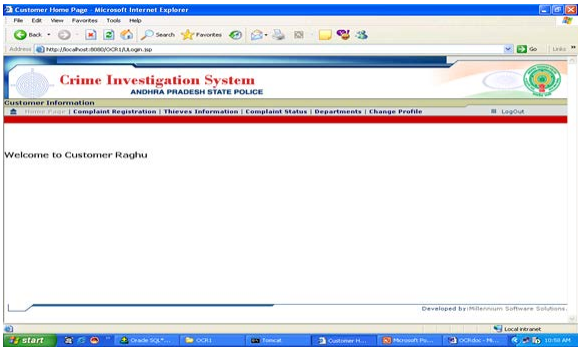


Fig.6.3.17 Customer Home page

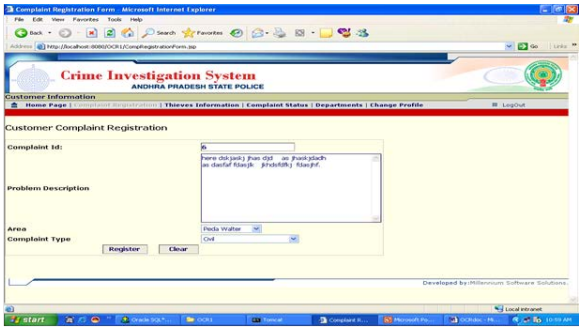


Fig.6.3.18 Complaint Registration Screen

Fig.6.3.19 Thief Details Screen

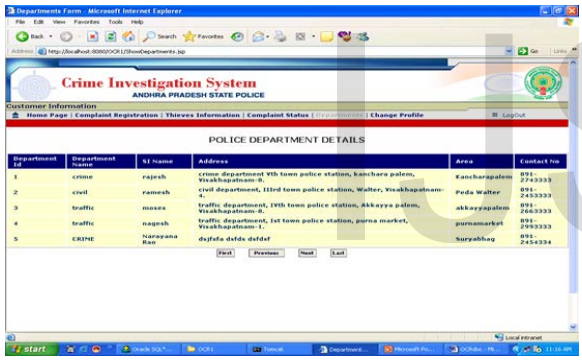


Fig.6.3.20 Department Details Screen

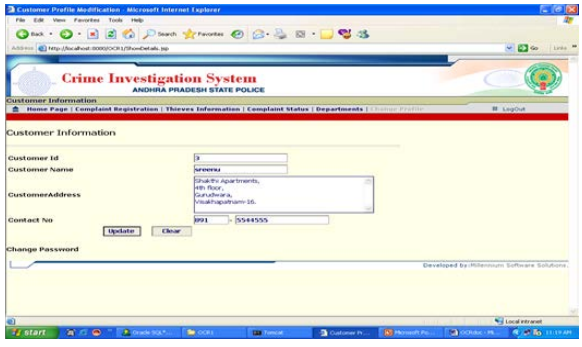


Fig.6.3.21 Changing Customer profile

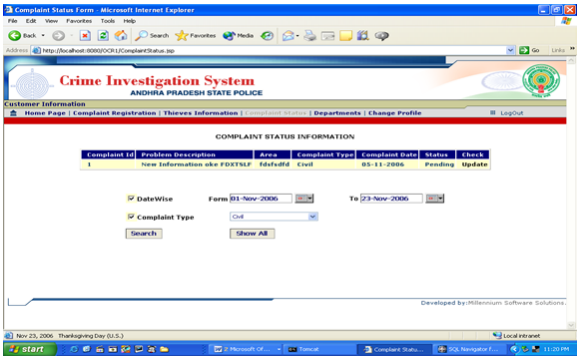


Fig.6.3.22 Complaint Status Details Screen

7. CONCLUSION

This application aims at providing the facility to the customers to pos their complaints. Online tendering as a tool to specify decision criteria, issue on line tenders and evaluating responses using this methodology. In all these cases the feedback was extremely positive and the users claimed they had. This software presents the part of an organization work as per the requirements, specifications and conditions mentioned in the user manual. This application s/w has been developed and completed successfully and also tests successfully by taking "Test Cases". It is user friendly and has all the needed menu options, which can be utilized by the user to perform the desired operations. Moreover help messages are provided which will help any authorized user from using the system without trouble.

1. FUTURE ENHANCEMENT

The visibility of the online tender handling solution is now increasing dramatically worldwide in recent years Now-a-days technology has been increasing and more are getting habituate to technology and mobile phone has been increasing rapidly and this application can be further developed as mobile application. So that it can be used in hands of the Customer and it becomes easy to access. . We can also further develop the system by using "Voice Recognition System". So this project has got scope to get developed in the near future. The scope of the project exists throughout the life of the product development, testing and implementation.

9. REFERENCES

[1.] JAVA 2: The Complete Reference, by Herbert Schildt.
[2.] Professional Java Server Programming by a Press
[3.] Object Oriented Analysis and Design with Application by Grady Booch.
[4.] Understanding JavaServer Pages Model 2 architecture (Java World)
[5.] Software Engineering Concepts by R. Fairly

[6.] http://www.w3schools.com/sql/
[7.] www.w3schools.com/html/

Author name : Hemanth Erabelli
Education : 10th - CALPS (cbse) cgpa=9.8
12th - CHAITANYA (state) percent=96%
Btech – Anurag group of institutions cgpa=9.41
Research : 1.Cloud computing
2. world wide web status analyzer
3.Online crime investigation system.
Membership : CSI (computer society of india)