**Chapter 1**

**INTRODUCTION**

* 1. **Statement Of the Problem**

iMAS is an android application placed in android mobile to Filter and rank the ads based on location based filtering but in existing Advertising Systems, all kind of ads cannot be viewed. Existing advertising systems has the problem of access delay. No effective targeting of customer. The ads does not define any location or venue. Lack of Location Based Filtering and Context Based Ranking mechanisms. Existing approach takes much time to detect the offer or ad which the user wants. When the number of ads is more, the user will miss the ad which he really wants.

* 1. **Objective Of the Work**

To develop an application that enables the user to view his nearby advertisements and get benefited. User can use this application to view the advertisements of any location and any product. He can use the link provided to view the complete details of the advertisement.

* 1. **Technology**

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android-SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

On July 2005, Google announced the redemption of Android Inc. The Android team created an operating system, named Android, based on the core of the operating system Linux. The target of this operating system was the mobile phones market, where SymbianOS and Windows Mobile dominated. The main benefit of Android is a rich library to enhance and ease communication and networking. Java is offered to software developers as the main selection for programming, enriched with software libraries developed by Google.

The first presentation of the Android platform was on November 5, 2007, along with the establishment of Open Handset Alliance. Google has issued most of the code of Android under the terms of Apache License. Eventually, Android is a platform that includes open source operating system, middleware and key applications for use on mobile devices based on the kernel Linux 2.6; in accordance with the principles of platform software design, based on available drivers hardware device. One level up, the libraries of the platform are found and the required virtual machine for converting and runtime applications. These are complemented by basic applications needed for the basic management unit for the initial communication and use of the device by the user. At the top of the platform architecture, lie the applications developed by software developers.

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Google Inc. purchased the initial developer of the software, Android Inc., in 2005. Android's mobile operating system is based on the Linux kernel. Google and other members of the Open Handset Alliance collaborated on Android's development and release. The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android. The Android operating system is the world's best-selling Smartphone platform. The first truly open and comprehensive platform for mobile devices, all of the software to run a mobile phone but without the proprietary obstacles that have hindered mobile innovation.

**1.5.1 Features of Android and why Android**

* **Handset layouts:** The platform is adaptable to larger, VGA, 2D graphics library, 3D graphics library based on OpenGL ES 2.0 specifications, and traditional smart phone layouts.
* **Storage:** The Database Software SQLite is used for data storage purposes.
* **Connectivity:** Android supports connectivity technologies including GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, WI-Fi, and WiMAX.
* **Messaging:** SMS and MMS are available forms of messaging including threaded text messaging.
* **Web browser:** The web browser available in Android is based on the open-source WebKit application framework. The browser scores a 93/100 on the Acid3 Test.
* **Java support:** Software written in Java can be compiled to be executed in the Dalvik virtual machine, which is a specialized VM implementation designed for mobile device use, although not technically a standard Java Virtual Machine. Android does not support J2ME, like some other mobile operating systems.
* **Media support:** Android supports the following audio/video/still media formats: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF and BMP.
* **Additional hardware support:** Android can use video/still cameras, touch screens, GPS, accelerometers, magnetometers, accelerated 2D bit blits (with hardware orientation, scaling, and pixel format conversion) and accelerated 3D graphics.
* **Development environment:** Includes a device emulator, tools for debugging, memory and performance profiling, and a plug-in for the Eclipse IDE.
* **Market:** Like many phone-based application stores, the Android Market is a catalog of applications that can be downloaded and installed to target hardware over-the-air, without the use of a PC. Originally only free applications were supported. Paid-for applications have been available on the Android Market in the United States since 19 February 2009.The Android Market has been expanding rapidly. As of April 30, 2010, it had over 50,000 Android applications for download.
* **Multi-touch:** Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero. The feature was initially disabled at the kernel level (possibly to avoid infringing Apple's patents on touch-screen technology). Google has since released an update for the Nexus One and the Motorola Droid which enables multi-touch natively.
* **Bluetooth:** Support for sending files over Bluetooth was added in version 2.0.
* **Video calling:** Android does not support video calling. However, it is possible if the phone runs an additional UI. This is proved on the HTC Evo 4G, which runs Sense and can support video calling.

Below are the reasons mentioned for choosing android.

* It is a simple and powerful SDKsupportingGSM, EDGE, and 3G networks, Wi-Fi, Bluetooth.
* It requires no licensing, distribution, or development fees.
* Development over many platform Linux, Mac OS, windows Excellent documentation
* Thriving developer community Java-based, easy to import 3rdparty Java library
* Funding (40+ G1 phones)Prize (amazon’skindle)Job opportunity is tremendous

**1.5.2 Android Architecture**

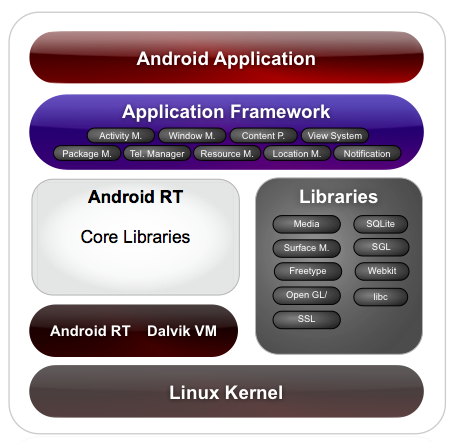


Fig 1.1

**1.5.3 Applications**

Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language.

**1.5.4 Application Framework**

* By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more.
* Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

**1.5.5 Underlying All Applications Is a Set of Services and Systems**

* A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser.
* Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data.
* A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files.
* A Notification Manager that enables all applications to display custom alerts in the status bar.
* An Activity Manager that manages the lifecycle of applications and provides a common navigation back stack.

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

**1.5.6 Role of DVM**

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. The Dali VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

**1.5.7 Linux Kernel**

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

**1.5.8 Android SDK 1.5**

There are several new features and UI updates included in the 1.5 update:

* + Ability to record and watch videos with the camcorder mode.
  + Uploading videos to YouTube and pictures to Picasa directly from the phone.
  + A new soft keyboard with an "Auto complete" feature.
  + Bluetooth A2DP support which in turn broke Bluetooth connectivity with many popular cars and headsets.
  + [[update]](http://en.wikipedia.org/w/index.php?title=Android_(operating_system)&action=edit)Ability to automatically connect to a Bluetooth headset within a certain distance.
  + New widgets and folders that can populate the Home screens.
  + Animations between screens.
  + Expanded ability of Copy and paste to include web page.

**1.5.9 Android Runtime**

Android includes a set of core libraries that provides most of the functionality available in the Core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik

Virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently.

The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .decks format by the included "dx" tool.



**Fig 1.2 Android Architecture**

**1.5.10 Advantages and Disadvantages**

There are a host of advantages that Google’s Android will derive from being open source software. Some of the advantages include:

* 1. The ability for anyone to customize the Google Android platform will open up the applications playing field to small and new players who lack the financial muscle to negotiate with wireless carriers like AT&T and Orange. The consumer will benefit from having a wide range of mobile applications to choose from since the monopoly will be broken by Google Android.
  2. Although this will depend on the carrier, one will be able to customize a mobile phones using Google Android platform like never before, right down to the screen. Features like weather details, opening screen, live RSS feeds and even the icons on the opening screen will be able to be customized.
  3. In addition, as a result of many mobile phones carrying Google Android, companies will come up with such innovative products like the location– aware services that will provide users with any information they might be in need of. This information could include knowing the location of a nearby convenience store or filling station. In addition the entertainment functionalities will be taken a notch higher by Google Android being able to offer online real time multiplayer games.
  4. However, Google Android is likely to experience some problems as follows:

These problems might include rejection of Google Android by carriers who charge a fee for some of the web based applications which Google Android will enable mobile phone holders to access free of charge.

* 1. Since Google Android is an open source platform driving all Android phones, it could significantly increase the risk of hacking these devices. If this happens, then locking mobiles as a security measure will be a mockery.

**1.6 Existing System**

Existing advertising systems are web based such as web-classified advertisements which can be accessed through desktop computers. User can browse these ads through mobile browsers but he will face resolution problem and access delay problem. The existing system is not built up with location based filtering mechanism. Even there is no ranking concept to order the ads in the way user need. There is no effective mobile based advertisement system at present.

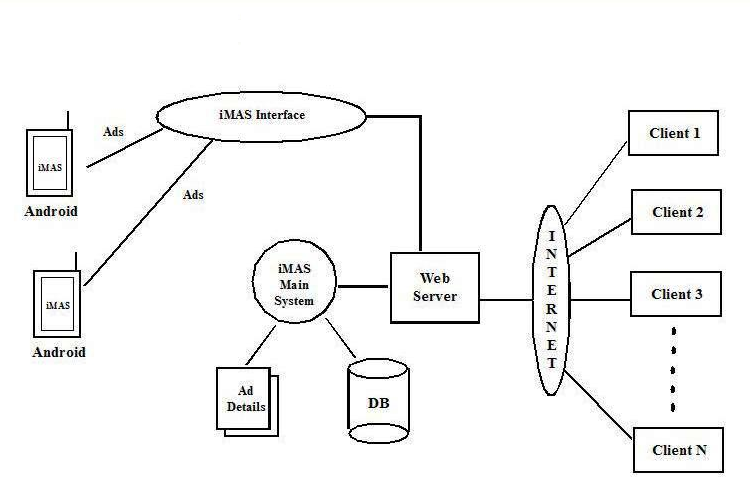
**1.6.1 Disadvantages of Existing System**

* In this system, advertising is time consuming.
* Advertising does not support targeting the needs of the user.
* User will miss the ads. When the number of ads are more.

**1.7 Proposed System**

The intelligent Mobile Advertising system uses Context Based Ranking to rank the ads based on the priority which is set by the user. It also uses Location Based Filtering which will be helpful for the user to find out the ads of the particular location.

**System Architecture:**

****

**1.7.1 Advantage in Proposed System**

* It provides an effective basis upon which personalization can be achieved.
* Defines the location and venue to serve the user in an effective way.
* More effective targeting of customers.
* Time taken to detect the offer or ad will be less.
* The user will find all the ads which he really wants.

**1.8 Steps to Configure Application Development**

**1.8.1 Install JDK**

Go to http://developers.sun.com/downloads/, Expand choice Java SE and Click on: Java SE (JDK) 6.

**1.8.2 Setting up the Android SDK**

Perform the following steps to set-up Android SDK

1. Prepare your development computer and ensure it meets the system requirements.
2. Install the SDK starter package from either one of this depending on the platform OS.
3. Windows installer\_r08-windows.exe, MacOSX(Intel)android-sdk\_r08-mac\_86.zip, Linux (i386)android-sdk\_r08-linux\_86.tgz
4. Install the ADT Plugin for Eclipse
5. Add Android platforms and other components to your SDK.
6. Explore the contents of the Android SDK

**1.8.3 Install Eclipse IDE**

Eclipse is a multi-language software development platform comprising an IDE and a plug-in system to extend it. It can be used to develop applications in Java and, by means of the various plugin, in other languages.

Go to http://www.eclipse.org/downloads/, Download the current and save it to drive C:\, Unzip the compress file to your hard drive (c:\eclipse)

**1.8.4 Installing the Eclipse ADT Plugin**

The **Android Development Tools** (ADT) plugin for Eclipse adds extensions to the Eclipse IDE. It allows you to create and debug Android applications easier and faster.

**Advantages**

1. It gives you access to other Android development tools from inside the Eclipse IDE. For example:
   1. take screenshots,
   2. Debug / set breakpoints, and
   3. View thread and process information directly from Eclipse.
2. It provides a New Project Wizard, which helps you quickly create and set up all of the basicfiles you'll need for a new Android.

**Steps involved are**

1. Start Eclipse, then select Help>Install New Software.
2. Click Add, in the top-right corner.
3. In the Add Repository dialog that appears, enter "ADT Plugin" for the Name and the following URL for the Location: https://dl-ssl.google.com/android/eclipse/ and ClickOK.
4. When the installation completes, restart Eclipse.

**1.4.5 Creating an Android Virtual Device (AVD)**

1. Android Virtual Devices (AVDs) are configurations of emulator options that let you better model an actual device.
2. In Eclipse, choose Window > Android SDK and AVD Manager.
3. Select Virtual Devices in the left panel and Click New.
4. The Create New AVD dialog appears and then Type the name of the AVD, such as “AVD23API9".
5. Choose a target (such as “Android 2.3 –API Level9”).
6. Optionally specify any additional settings (SD, camera, trackball, ….) YES to all.
7. Click Create AVD.

**1.8.6 Testing the Emulator**

Android Virtual Devices (AVDs) are configurations of emulator options that let you better model an actual device.

1. In Eclipse, choose Window > Android SDK and AVD Manager.
2. Select Virtual Devices in the left panel.
3. Click on an AVD and Click Start.



**Fig-1.3 Emulator**

**1.8.7 Creating an Android Project**

To create a new project

1. Start Eclipse and Select File> New> Project.
2. Select Android> Android Project, and click Next.
3. Enter Project name: AndHolaMundo.
4. Select Target Android 1.5.
5. Application name: Hola.
6. Package name: cis493.demo.
7. Create Activity: HolaMundo.
8. Min SDK Version: 3 and Click Finish.

Creating an Android Project Once you complete the New Project Wizard, ADT creates the following folders and files in your new project:

1. Src/ Includes your stub Activity Java file. All other Java files for your application go here.
2. <Android Version>/ (e.g., Android 1.5/) Includes the android.jar file that your application will build against.
3. Gen/ This contain the Java files generated by ADT, such as your R.java file and interfaces created from AIDL files.
4. Assets/This is empty. You can use it to store raw asset files.
5. Res/A folder for your application resources, such as drawable files, layout files, string values etc.
6. AndroidManifest.xml The Android Manifest for your project.
7. Default.properties This file contains project settings, such as the build target.

**Chapter 2**

**REQUIRMENT SPECIFICATION**

**2.1 Hardware and software requirements**

**2.1.1 Hardware Requirements**

1. Pentium 2 GHz or higher series
2. At least 512 MB or higher
3. At least 500 gigabytes (GB) of available space on the hard disk
4. Android Mobile.
5. CD-ROM or DVD-ROM drive
6. Keyboard and a Microsoft Mouse or some other compatible pointing device.

**2.1.2 Software Requirements:**

**Operating System:**

1. Ubuntu 10.10 Operating System.
2. Android 4.2 Operating System (Client side)

**Client Side:**

1. Emulator
2. For Development : Eclipse 3.6.2

**Server Side:**

1. Java Enabled Web Server / App Server(Tomcat)

**2.2 Functional Requirements**

This system having two applications one is web application which is going to developed using J2EE technology in MVC architecture. Another application is android application which is developed with android SDK tools. The web application has following three applications

* Admin Application
* Android Application

**Admin Application:** Admin is responsible for maintain the cloud configurations, details of advertisements and location details.

**Android User Application:** This is an android application which can be installed in any Android based mobile phones.

**2.3 Non Functional Requirements**

Non functional requirements a description and, where possible, target values of associated non-functional requirements. Non-functional requirements detail constraints, targets or control mechanisms for the new system. They describe how, how well or to what standard a function should be provided. For example, levels of required service such as response times; security and access requirements; technical constraints; required interfacing with users' and other systems; and project constraints such as implementation on the organization’s hardware/software platform. Service level requirements are measures of the quality of service required, and are crucial to capacity planning and physical design.

Identify realistic, measurable target values for each service level. These include service hours, service availability, responsiveness, throughput and reliability. Security includes defining priority and frequency of backup of data, recovery, fallback and contingency planning and access restrictions. Access restrictions should deal with what data needs protected; what data should be restricted to a particular user role; and level of restriction required, eg physical, password, view only. Non-functional requirements may cover the system as a whole or relate to specific functional requirements.

[**Accessibility**](https://en.wikipedia.org/wiki/Accessibility)**:** The accessibility factor to the system depends on the availability of a broad band connection. The access to our system is possible in places where the internet is available. Since usually the login and password hacking happens in most of the domains which are implemented on a global server. For our Opass system to be accessible anywhere and anytime we have deployed our system on a global server space and this global server has to always be up and running. The global server is provided with anti-virus, fire wall security and safe internet usage mode in order to avoid the server being hacked or intruded.

[**Availability**](https://en.wikipedia.org/wiki/Availability)**:** The system has to be available anywhere and anytime whenever the user requires the use of the system. Here we consider the issue of the service level agreement with various domains in order to embed our system with theirs. So as for as now we have created our own domain called as Exclusivebuy.com where we demonstrate our Opass password authentication attack prevention mechanism.

**Deployment:** The deployment is carried out in various stages such as Android deployment, Database deployment, and deployment of the source code.

[**Documentation**](https://en.wikipedia.org/wiki/Documentation)**:** the documentation is carried out for every stage in order to summarize and record the result of every stage since the time the system development was started.

**Efficiency:** The efficiency of the system is now related in terms of the number of users able to login. We have not provided the factor of simultaneous login because our system is built under the cost constraint and this can be further extended and developed.

[**Failure management**](https://en.wikipedia.org/w/index.php?title=Failure_management&action=edit&redlink=1)**:** At times of system failure the users may be puzzled about what is the problem that is taking place in the system. So we have created various exceptions in order to provide notifications to the users to help them assist and guide them in rectifying the errors.

[**Security**](https://en.wikipedia.org/wiki/Security)**:** We are developing our system with the view of making the password authentication in various websites and domain. So sufficient security is provided to our system in order to make it safe for the users [Usability](https://en.wikipedia.org/wiki/Usability) by target user community: The usability factor can be extended to all the domains where the password authentication phase is used and thus the security of the password and the confidence of the password being safe from the hackers and phishing software’s is enhanced.

These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process etc. Few of them are:

**Communication Interfaces:** The application requires an internet connection to perform browsing And downloading files from the internet.

**Safety requirements:** Application ensures that it does not download malicious and infected links from web sites.

**Security requirements:** Provides automatic file checking capability to ensure the security of mobile platform and the application.

**Software quality attributes:** The source code of the application is going to be open as this is going to be open source software. It will be free for further modifications and improvements.

**Chapter 3**

**DESIGN**

**3.1 Introduction to System Analysis**

A system is an orderly group of interdependent components linked together according to a plan to achieve a specific objective. Its main characteristics are organization, interaction, interdependence, integration and a central objective.

**3.1.1 System Analysis**

System analysis and design are the application of the system approach to problem solving generally using computers. To reconstruct a system the analyst must consider its elements output and inputs, processors, controls feedback and environment.

Analysis is a detailed study of the various operations performed by a system and their relationships within and outside of the system. One aspect of analysis is defining the boundaries of the system and determining whether or not a candidate system should consider other related systems. During analysis data are collected on the available files decision points and transactions handled by the present system. This involves gathering information and using structured tools for analysis.

**3.2 Feasibility study**

Feasibility is the determination of whether or not a project is worth doing. The process followed in making this determination is called feasibility Study. This type of study if a project can and should be taken. In the conduct of the feasibility study, the analyst will usually consider seven distinct, but inter-related types of feasibility.

**3.2.1 Technical Feasibility**

This is considered with specifying equipment and software that will successful satisfy the user requirement the technical needs of the system may vary considerably but might include

1. The facility to produce outputs in a given time.
2. Response time under certain conditions.
3. Ability to process a certain column of transaction at a particular speed.

**3.2.2 Economic Feasibility**

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as cost / benefit analysis. The procedure is to determine the benefits and savings are expected form a proposed system and a compare them with costs. It benefits outweigh costs; a decision is taken to design and implement the system will have to be made if it is to have a chance of being approved. There is an ongoing effort that improves in accuracy at each phase of the system life cycle.

**3.2.3 Operational Feasibility**

It is mainly related to human organization and political aspects. These points are considered are

1. What changes will be brought with the system?
2. What organizational structures are distributed?
3. What new skills will be required?
4. Do the existing system staff members have these skills?
5. If not, can they be trained in the course of time?

**Chapter 4**

**Modules and Description:**

iMAS application has two applications. One is developed using J2EE technology and another is developed using Android Technology. iMAS J2EE application has to run in Web Server, iMAS Android app has to be installed in consumer mobile. iMAS has two type of users one is Admin user and another one is Android User. Admin user is a main user who will feed the Ads into the system and he is responsible for Location and Priority settings

Another user in this system is Android user who we can call consumer. When Consumer request for the ads, the request will send to iMAS J2EE application which is running in web server. iMAS J2EE application has two main division, one is iMAS Interface and another is iMAS Scheduler Service. iMAS interface is responsible for interacting with consumer android app, which will get the request from android app along with Geo Coordinates and using Geo-Location service it will find the location of the consumer and pass the location details and consumer id to iMAS Scheduler, which responsible for filtering ads and ranking the ads based user context details and location details. iMAS Scheduler has to do Ads Expiry management, Location Based ads filtering, ranking the ads based on user context.

Once ads are generated it will given to iMAS report which will pass it to iMAS interface. iMAS interface will transfer the ads to respective android apps where consumer can view the ads and get more benefit.

The proposed system modules are:

**MODULE 1- ADMIN SESSION (J2EE)**

**LOGIN**

Admin has to login using valid user name and password.If not, it will display error

**ADS**

The admin posts the different kinds of ads in the web server.It includes Ad Heading, Message, Location, Start Date, End Date, Link, Venue and Priority.

**LOCATION**

The admin sets the location of the ads. It is based on latitude and longitude values.

**PRIORITY SETTING**

The admin sets the priority for the particular ad. For e.g. IT Professional, House wife etc..

**CHANGE PASSWORD**

The admin can change his password if required. The password can be changed only when the current password is entered.

**MODULE 2- USER SESSION (ANDROID)**

**USER REGISTRATION**

The user has to register himself with valid user details. They are Name, User Name, Password, Sex, DOB, Address, Location, Priority, Mobile No, Phone No and Email id

**LOGIN**

The user has to login with valid user name and password. If not, it will display error.

**VIEW PRIORITY ADS**

The ads will be ranked based on the priority of the user.

**VIEW LOCATION BASED ADS**

The ads will be ranked based on the priority of the user. The ranked ads will be filtered based on the location set by the user.

**VIEW CURRENT LOCATION ADS (GPS)**

The user’s current location will be considered based on the latitude and longitude values. Then the ads will be ranked and filtered based on his current location.

**VIEW ALL ADS**

If required, then the user can view all the ads which is stored in the database.

**CHANGE PASSWORD**

The user can also change his password. Password can be changed only when the current password is entered.

**CHAPTER 5**

**5.1 Data Flow Diagram**

A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities. DFD's show the flow of data from external entities into the system, how the data moves from one process to another, as well as its logical storage.

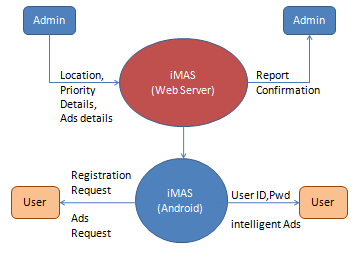
There are only four symbols

* + 1. Squares representing external entities, which are sources and destinations of data.
    2. Rounded rectangles representing processes, which take data as input, do some

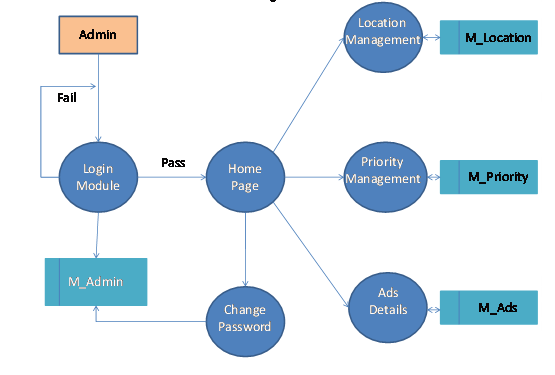
Operation and output the data.

* + 1. Arrows representing the data flows, which can either, be electronic data or physical Items.
    2. Cylinder representing the database.

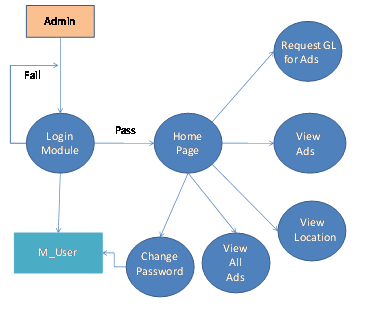
CONTEXT ANALYSIS DIAGRAM

****

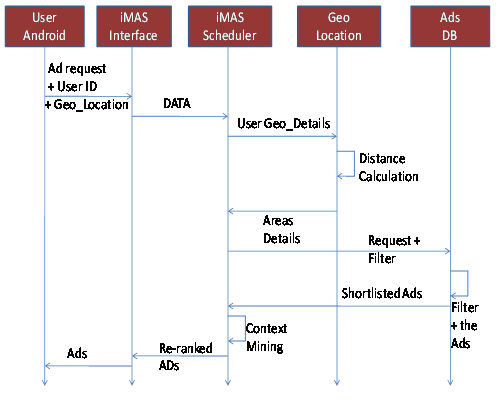
**ADMIN SESSION**

****

**USER SESSION**

****

**5.2 Sequence Diagram**

****

**Chapter 6**

**TESTING**

Testing performs a very critical role for quality assurance and for ensuring the reliability of the software. The success of testing for errors in programs depends critically on the test cases.

**6.1 Testing phase**

The completion of the system is achieved only after it has been thoroughly tested. Though this gives a feel that project is completed there cannot be any project without going through this stage. Hence in this stage it is decided whether this project can undergo real time environment execution without any breakdowns, therefore the package can be rejected even at this stage.

A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. This is a non-trivial pursuit. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed.

**6.2 System testing**

Testing is a set of activities that can be planned in advance and conducted systematically. The proposed system is tested in parallel with the software that consists of its own phases of its analysis, implementation, testing and maintenance. Following are the tests conducted on the system.

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

**6.3 Unit testing**

During this implementation of the system each module of the system was tested separately to uncover errors within its boundaries. User interface is used as a guide in the process

In computer programming, unit testing is a method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure. In object-oriented programming a unit is usually a method. Unit tests are created by programmers or occasionally by white box testers during the development process.

Ideally, each test case is independent from the others: substitutes like method stubs, mock objects,fakes and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended. Its implementation can vary from being very manual (pencil and paper) to being formalized as part of build automation.

**6.4 Module testing**

A module is composed of various programs related to that module. Module testing is done to check the module functionality and interaction between units within a module. It checks the functionality of each program with relation to other programs within the same module. It then tests the overall functionality of each module.

This module introduces the technique of functional (black box) unit testing to verify the correctness of classes. It shows how to design unit test cases based on a class specification within a contract programming approach. The laboratory exercises then guide students through creating and running tester classes in Java from a test case design, utilizing the Joint unit test framework. It also contains a worked example on how to unit test GUI and event handling classes.

Testing is an essential part of any software development process, but it is often poorly understood. In this module, we take a simple approach and look at two key approaches to testing. Unit testing is done at the class level, and is designed with the aid of method pre- and post-conditions. System testing is done at the program level, and is designed based on documented use cases. We look at how your approach to object-oriented design is influenced by the need to design and execute tests, and close with some other issues in testing.

**6.5 Integration testing**

Integration testing is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing. The object is to take unit tester module and build a program structure that has been dictated by the design.

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using Black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

Test cases are constructed to test that all components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

**6.6 Acceptance testing**

This software has been tested with realistic data given by the client and produced results. Then the client satisfying all the requirements specified by them has also developed the software within the time limitations specified. A demonstration has been given to the client and the end user giving all the operational features.

Acceptance testing generally involves running a suite of tests on the completed system. Each individual test, known as a case, exercises a particular operating condition of the user's environment or feature of the system, and will result in a pass or fail, or [Boolean](http://en.wikipedia.org/wiki/Boolean_logic), outcome. There is generally no degree of success or failure. The test environment is usually designed to be identical, or as close as possible, to the anticipated user's environment, including extremes of such. These test cases must each be accompanied by test case input data or a formal description of the operational activities (or both) to be performed—intended to thoroughly exercise the specific case—and a formal description of the expected results.

**6.6.1Test plan**

The test plan contains the following:

* Features to be tested.
* Approach to be tested.
* Test deliverables.

**6.7 Features to be tested**

All the functional features specified in the required document will be tested.Test coverage in the test plan states what requirements will be verified during what stages of the product life. Test Coverage is derived from design specifications and other requirements, such as safety standards or regulatory codes, where each requirement or specification of the design ideally will have one or more corresponding means of verification.

Test coverage for different product life stages may overlap, but will not necessarily be exactly the same for all stages. For example, some requirements may be verified during Design Verification test, but not repeated during Acceptance test. Test coverage also feeds back into the design process, since the product may have to be designed to allow test access .

Test methods in the test plan state how test coverage will be implemented. Test methods may be determined by standards, regulatory agencies, or contractual agreement, or may have to be created new. Test methods also specify test equipment to be used in the performance of the tests and establish pass/fail criteria. Test methods used to verify hardware design requirements can range from very simple steps, such as visual inspection, to elaborate test procedures that are documented separately.

**6.8 Approach for testing**

For unit testing structured testing based on branch coverage criteria will be used. The goal is to achieve branch coverage of more than 95% system testing will be functional in nature. A test plan documents the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements. A test plan is usually prepared by or with significant input from Test Engineers.

Depending on the product and the responsibility of the organization to which the test plan applies, a test plan may include one or more of the following:

1. **Design Verification or Compliance test** - to be performed during the development or approval stages of the product, typically on a small sample of units.
2. **Manufacturing or Production test** - to be performed during preparation or assembly of the product in an ongoing manner for purposes of performance verification and quality control.
3. **Acceptance or Commissioning test** - to be performed at the time of delivery or installation of the product.
4. **Service and Repair test** - to be performed as required over the service life of the product.
5. **Regression test** - to be performed on an existing operational product, to verify that existing functionality didn't get broken when other aspects of the environment are changed (e.g., upgrading the platform on which an existing application runs).
6. **Test deliverables**
   1. The following documents will be required:
   2. Unit tests report for each unit.
   3. Test case specification for system testing.
   4. Test reports for system testing.

**Chapter 7**

**Table Structure**

|  |  |
| --- | --- |
| **m\_ad** |  |
|  |  |

|  |
| --- |
| **Fields** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Key** | **Default** | **Extra** |
| ad\_id | int(10) | PRI | (NULL) | auto\_increment |
| ad\_head | varchar(150) |  |  |  |
| ad\_link | varchar(150) |  |  |  |
| ad\_msg | varchar(200) |  |  |  |
| ad\_venue | varchar(100) |  |  |  |
| ad\_location | int(10) |  |  |  |
| priority | varchar(100) |  |  |  |
| start\_date | varchar(100) |  |  |  |
| end\_date | varchar(100) |  |  |  |
| status | varchar(50) |  | Active |  |

|  |  |
| --- | --- |
| **m\_admin** |  |
|  |  |

|  |
| --- |
| **Fields** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Key** | **Default** | **Extra** |
| id | int(10) | PRI | (NULL) | auto\_increment |
| username | varchar(100) |  | (NULL) |  |
| password | varchar(100) |  | (NULL) |  |
| name | varchar(100) |  | (NULL) |  |

|  |  |
| --- | --- |
| **m\_location** |  |
|  |  |

|  |
| --- |
| **Fields** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Key** | **Default** | **Extra** |
| loc\_id | int(10) | PRI | (NULL) | auto\_increment |
| location | varchar(150) |  |  |  |
| latitude | varchar(150) |  |  |  |
| longitude | varchar(150) |  |  |  |

|  |  |
| --- | --- |
| **m\_priority** |  |
|  |  |

|  |
| --- |
| **Fields** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Key** | **Default** | **Extra** |
| priority\_id | int(10) | PRI | (NULL) | auto\_increment |
| priority\_name | varchar(150) |  |  |  |
| remarks | varchar(225) |  |  |  |

|  |  |
| --- | --- |
| **m\_user** |  |
|  |  |

|  |
| --- |
| **Fields** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Key** | **Default** | **Extra** |
| id | int(5) | PRI | (NULL) | auto\_increment |
| userid | varchar(50) | PRI |  |  |
| password | varchar(50) |  |  |  |
| username | varchar(20) |  |  |  |
| gender | varchar(50) |  |  |  |
| dob | varchar(100) |  |  |  |
| address | varchar(50) |  |  |  |
| location | varchar(100) |  |  |  |
| priority | varchar(100) |  |  |  |
| mobile | varchar(12) |  |  |  |
| phone | varchar(12) |  |  |  |
| email | varchar(100) |  |  |  |

**Chapter 8**

**RESULT AND RESULT ANALYSIS**

**8.1 Implementation**

**ANDROID**

HOME PAGE

package com.dev;

import java.util.ArrayList;

import java.util.Arrays;

import org.apache.http.NameValuePair;

import org.apache.http.message.BasicNameValuePair;

import com.HttpClient.CustomHttpClient;

import com.dev.R;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.Dialog;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.widget.EditText;

import android.widget.LinearLayout;

import android.widget.TextView;

import android.widget.Toast;

public class HomePage extends Activity

{

Dialog dialog=null;

AlertDialog.Builder settingsDialog;

LinearLayout settingSLayout;

EditText un;

EditText old\_pwd;

EditText new\_pwd;

EditText confirm\_pwd;

String oldPwd="";

String newPwd="";

String confirmPwd="";

String oldPwdValue="";

String newPwdValue="";

String confirmPwdValue="";

String validationMsg="";

String username="";

String result[];

String locationList[];

TextView exception;

String selectedOption="",selectedLocation="";

String response = "",res = "";

@Override

public void onCreate(Bundle savedInstanceState)

{

// TODO Auto-generated method stub

super.onCreate(savedInstanceState);

setContentView(R.layout.home\_page);

Intent intent = getIntent();

username = (String) intent.getSerializableExtra("un");

System.out.println("\*\*\*\*\*\*\*\*\*\*\* Login Info \*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Username : " +username);

exception = (TextView)findViewById(R.id.exception);

}

//All About Home Menu

@Override

public boolean onCreateOptionsMenu(Menu menu)

{

menu.add(0,0,0,"View Ad").setIcon(R.drawable.view2);

menu.add(0,2,0,"Settings").setIcon(R.drawable.setting1);

menu.add(0,3,0,"Logout").setIcon(R.drawable.logout);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item)

{

switch (item.getItemId())

{

case 0:

viewAd();

break;

case 1:

break;

case 2:

ChangePassword();

break;

case 3:

Logout();

break;

default:

break;

}

return true;

}

public void viewAd()

{

//'View Ad' Option : Ads that belongs to user priority Only

result = new String[]{"View Ad","View All Ad","Location Specific Ad","Current Location(GPS)"};

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle("Select Your Choice");

builder.setIcon(R.drawable.view);

builder.setSingleChoiceItems(result, -1, new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int pos)

{

selectedOption=(String) result[pos];

selectedOption=selectedOption.trim();

}

});

builder.setPositiveButton("View Ad",new DialogInterface.OnClickListener()

{

public void onClick(DialogInterface dialog, int pos)

{

//'View Ad' Option : Ads that belongs to user priority Only

if(selectedOption.equals("View Ad"))

{

Intent viewAdIntent = new Intent(HomePage.this,ViewAd.class);

viewAdIntent.putExtra("selected\_option",selectedOption);

startActivity(viewAdIntent);

}

else if(selectedOption.equals("View All Ad"))

{

Intent viewAdIntent = new Intent(HomePage.this,ViewAd.class);

viewAdIntent.putExtra("selected\_option",selectedOption);

startActivity(viewAdIntent);

}

else if(selectedOption.equals("Location Specific Ad"))

{

// Getting The All Locations

ArrayList<NameValuePair> postParameters = new ArrayList<NameValuePair>();

try

{

response=CustomHttpClient.executeHttpPost(Global.URL+"Res/JSP/User/JSP-For-Android/get\_locations.jsp",postParameters);

res=response.toString();

res = res.trim();

res = res.replaceAll("\'","\\\\'"); // replacing ' to \\'

locationList = res.split("~");

System.out.println("Location List : " + Arrays.toString(locationList));

}

catch (Exception e)

{

System.out.println("Opps,While Getting Location : ");

e.printStackTrace();

}

AlertDialog.Builder bldr = new AlertDialog.Builder(HomePage.this);

bldr.setTitle("Select The Ad Location");

bldr.setSingleChoiceItems(locationList, -1, new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int pos)

{

selectedLocation=(String) locationList[pos];

selectedLocation=selectedLocation.trim(

}

});

bldr.setPositiveButton("View Ad",new DialogInterface.OnClickListener()

{

public void onClick(DialogInterface arg0, int arg1)

{

Toast.makeText(getApplicationContext(),"Selcted Location : " + selectedLocation,Toast.LENGTH\_LONG).show();

Intent viewAdIntent = new Intent(HomePage.this,ViewAd.class);

viewAdIntent.putExtra("selected\_option",selectedLocation);

startActivity(viewAdIntent);

}

});

bldr.setNegativeButton("Cancel",new DialogInterface.OnClickListener()

{

public void onClick(DialogInterface dialog, int pos)

{

Toast.makeText(getApplicationContext(),"View Ad Process Canceled...",Toast.LENGTH\_LONG).show();

dialog.dismiss();

}

});

bldr.show();

}

else if(selectedOption.equals("Current Location(GPS)"))

{

Intent viewAdByCLIntent = new Intent(HomePage.this,GPSTracking.class);

startActivity(viewAdByCLIntent);

}

dialog.dismiss();

}

});

builder.setNegativeButton("Cancel",new DialogInterface.OnClickListener()

{

public void onClick(DialogInterface dialog, int pos)

{

Toast.makeText(getApplicationContext(),"View Ad Process Canceled...",Toast.LENGTH\_LONG).show();

dialog.dismiss();

}

});

builder.show();

}

public void settings()

{

Intent settingIntent = new Intent(this,ChangePassword.class);

startActivity(settingIntent);

}

public void ChangePassword()

{

SharedPreferences pref =getApplicationContext().getSharedPreferences("any\_prefname",MODE\_PRIVATE);

username = pref.getString("username","");//2nd arg is the default value

showDialog(1);

}

public void Logout()

{

Intent logoutIntent = new Intent(this,IMAS\_Home.class);

startActivity(logoutIntent);

}

/\* Changing The Password (Starts) \*/

protected Dialog onCreateDialog(int id)

{

switch(id)

{

case 1:

settingsDialog = new AlertDialog.Builder(HomePage.this);

settingsDialog.setTitle(" Change Password ");

settingsDialog.setIcon(R.drawable.setting1);

settingSLayout = new LinearLayout(HomePage.this);

settingSLayout.setOrientation(LinearLayout.VERTICAL);

TextView tv = new TextView(HomePage.this);

TextView tv1 = new TextView(HomePage.this);

TextView tv2 = new TextView(HomePage.this);

TextView tv3 = new TextView(HomePage.this);

TextView tv4 = new TextView(HomePage.this);

tv.setText("Username");

tv1.setText("Old Password");

tv2.setText("New Password");

tv3.setText("Confirm Password");

tv4.setText(validationMsg);

un = new EditText(HomePage.this);

un.setHint(username);

un.setText(username);

un.setEnabled(false);

old\_pwd = new EditText(HomePage.this);

old\_pwd.setHint("Enter Your Old Password.");

old\_pwd.setText(oldPwdValue);

new\_pwd = new EditText(HomePage.this);

new\_pwd.setHint("Enter Your New Password.");

new\_pwd.setText(newPwdValue);

confirm\_pwd = new EditText(HomePage.this);

confirm\_pwd.setHint("Confirm Your New Password.");

confirm\_pwd.setText(confirmPwdValue);

settingSLayout.addView(tv);

settingSLayout.addView(un);

settingSLayout.addView(tv1);

settingSLayout.addView(old\_pwd);

settingSLayout.addView(tv2);

settingSLayout.addView(new\_pwd);

settingSLayout.addView(tv3);

settingSLayout.addView(confirm\_pwd);

settingSLayout.addView(tv4);

settingsDialog.setView(settingSLayout);

settingsDialog.setPositiveButton("Change Password",new DialogInterface.OnClickListener()

{

public void onClick(DialogInterface dialog, int pos)

{

username = un.getText().toString();

oldPwd = old\_pwd.getText().toString().trim();

newPwd = new\_pwd.getText().toString().trim();

confirmPwd = confirm\_pwd.getText().toString();

if(oldPwd.equals("") && newPwd.equals("") && confirmPwd.equals(""))

{

validationMsg="Please,Enter The Mandatory Fields Value.";

showDialog(1);

Toast.makeText(getApplicationContext(),validationMsg,Toast.LENGTH\_LONG).show();

}

else if(oldPwd.equals(""))

{

oldPwdValue="";

newPwdValue=newPwd;

confirmPwdValue=confirmPwd;

validationMsg="Please,Enter The Old Password.";

old\_pwd.requestFocus();

showDialog(1);

Toast.makeText(getApplicationContext(),validationMsg,Toast.LENGTH\_LONG).show();

}

else if(newPwd.equals(""))

{

oldPwdValue=oldPwd;

newPwdValue="";

confirmPwdValue=confirmPwd;

validationMsg="New Password Can Not Be Blanked.";

new\_pwd.requestFocus();

showDialog(1);

Toast.makeText(getApplicationContext(),validationMsg,Toast.LENGTH\_LONG).show();

}

else if(confirmPwd.equals(""))

{

oldPwdValue=oldPwd;

newPwdValue=newPwd;

confirmPwdValue="";

validationMsg="Confirm Password Can Not Be Blanked.";

confirm\_pwd.requestFocus();

showDialog(1);

Toast.makeText(getApplicationContext(),validationMsg,Toast.LENGTH\_LONG).show();

}

else if(!newPwd.equals(confirmPwd))

{

oldPwdValue=oldPwd;

newPwdValue=newPwd;

confirmPwdValue=confirmPwd;

validationMsg="New Password and Confirm Password Is Not Same.";

showDialog(1);

Toast.makeText(getApplicationContext(),validationMsg,Toast.LENGTH\_LONG).show();

}

else

{

oldPwdValue="";

newPwdValue="";

confirmPwdValue="";

validationMsg="";

System.out.println("Old Password : " + oldPwd);

System.out.println("New Password : " + newPwd);

ArrayList<NameValuePair> changePassParameters = new ArrayList<NameValuePair>();

changePassParameters.add(new BasicNameValuePair("un",username));

changePassParameters.add(new BasicNameValuePair("old\_Pwd",oldPwd));

changePassParameters.add(new BasicNameValuePair("new\_Pwd",newPwd));

changePassParameters.add(new BasicNameValuePair("coinfirm\_Pwd",confirmPwd));

try

{

response = CustomHttpClient.executeHttpPost(Global.URL+"Res/JSP/User/JSP-For-Android/changePassword.jsp",changePassParameters);

//192.168.1.16 is the System IP address,to get type ipconfig in cmd prompt

res=response.toString();

res= res.replaceAll("\\s+","");

if(res.equals("true"))

{

/\* Displaying Confirmation Message \*/

AlertDialog.Builder alertBox =new AlertDialog.Builder(HomePage.this);

alertBox.setTitle("Confirmation Message");

alertBox.setIcon(R.drawable.ok\_icon);

alertBox.setMessage("Password Changed Sucessfully..");

alertBox.setNegativeButton("OK",new DialogInterface.OnClickListener()

{

public void onClick(DialogInterface dialog, int pos)

{

Toast.makeText(getApplicationContext(),"Password Changed Sucessfully...",Toast.LENGTH\_LONG).show();

Intent homeIntent = new Intent(HomePage.this,IMAS\_Home.class);

startActivity(homeIntent);

}

});

alertBox.show();

}

Else

{

/\* Displaying Confirmation Message \*/

AlertDialog.Builder alertBox =new AlertDialog.Builder(HomePage.this);

alertBox.setTitle("Authuntication Message");

alertBox.setIcon(R.drawable.cancel\_icon);

alertBox.setMessage("Sorry,Change Password Process Failed,Try Again..");

alertBox.setNegativeButton("OK",new DialogInterface.OnClickListener()

{

public void onClick(DialogInterface dialog, int pos)

{

Toast.makeText(getApplicationContext(),"Sorry,Change Password Process Failed,Try Again..",Toast.LENGTH\_LONG).show();

}

});

alertBox.show();

}

}

catch (Exception e)

{

System.out.println("\*\*\*\*\* Opps,Exception In HomePage-ChangePassword() \*\*\*\*\*");

e.printStackTrace();

}

}

}

});

settingsDialog.setNegativeButton("Cancel",new DialogInterface.OnClickListener()

{

public void onClick(DialogInterface dialog, int pos)

{

Toast.makeText(getApplicationContext(),"Change Password Process Canceled...",Toast.LENGTH\_LONG).show();

dialog.dismiss();

}

});

settingsDialog.show();

break;

default:

dialog = null;

break;

}

return dialog;

}//Change Password() Ends

}

**J2EE**

<%@page import=*"*com.dev.util.Utility*"*%>

<html>

<head>

<link href=*"*<%=request.getContextPath() %>*/Res/CSS/style.css"* rel=*"stylesheet"* type=*"text/css"* />

<link href=*"*<%=request.getContextPath() %>*/Res/CSS/message.css"* rel=*"stylesheet"* type=*"text/css"* />

<link rel=*"stylesheet"* href=*"*<%=request.getContextPath() %>*/Res/CSS/login.css"* type=*"text/css"*/>

<script type=*"text/javascript"* src=*"*<%=request.getContextPath() %>*/Res/JS/style.js"*></script>

<script src=*"*<%=request.getContextPath() %>*/Res/JS/jquery-1.4.2.min.js"* type=*"text/javascript"*></script>

<script type=*"text/javascript"*>

$(document).ready(**function**()

{

$(".TabMenu span:first").addClass("selector");

$(".TabMenu span").mouseover(**function**()

{

$(**this**).addClass("hovering");

});

$(".TabMenu span").mouseout(**function**()

{

$(**this**).removeClass("hovering");

});

$(".TabMenu span").click(**function**()

{

$(".selector").removeClass("selector");

$(**this**).addClass("selector");

**var** TabWidth = $(".TabContent:first").width();

**if**(parseInt($(".TabContent:first").css("margin-left")) > 0)

TabWidth += parseInt($(".TabContent:first").css("margin-left"));

**if**(parseInt($(".TabContent:first").css("margin-right")) >0)

TabWidth += parseInt($(".TabContent:first").css("margin-right"));

**var** newLeft = -1\* $("span").index(**this**) \* TabWidth;

$(".AllTabs").animate({

left: + newLeft + "px"},1000);

});

});

</script>

</head>

<body onload="startTimer()">

<img src=*"*<%=request.getContextPath() %>*/Res/Images/header.jpg"*></img>

<div class=*"TabMenu"* id=*"label"*>

<span id=*"AdminLogin"*>

<img src=*"*<%=request.getContextPath() %>*/Res/Images/user.png"* height=*"25"* width=*"25"* />Admin&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<br>

</span>

<span id=*"UserLogin"*>

<img src=*"*<%=request.getContextPath() %>*/Res/Images/user.png"* height=*"25"* width=*"25"* />User&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<br>

</span>

</div>

<br>

<div class=*"ContentFrame"*>

<div class=*"AllTabs"*>

<div class=*"TabContent"*>

<p><h2>Admin Login</h2>

<form method=*"post"* id=*"login"* action=*"*<%=request.getContextPath() %>*/AdminLogin"*>

Name&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input class=*"field"* name=*"name"* type=*"text"* required=*"yes"*/><br><br>

Password<input class=*"field"* name=*"pass"* type=*"password"* required=*"yes"*/><br><br>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<input class=*"button"* type=*"submit"*/>

</form>

</p>

</div>

<div class=*"TabContent"*>

<p>

<h2>Users Can Login</h2>

<h2>Through Android Mobile</h2>

<img alt=*""* src=*"*<%=request.getContextPath()%>*/Res/Images/mobile.jpg"*>

</p>

</div>

</div>

</div>

<%

**int** no=Utility.parse(request.getParameter("no"));

**if**(no==101)

{%>

<div class=*"error"* id=*"message"*>

<p>Opp's,Invalid userid / password..!</p>

</div>

<script type=*"text/javascript"*>

$(document).ready(**function**()

{

$('#AdminLogin').trigger('click');

});

</script>

<%

}

**if**(no==102)

{%>

<div class=*"error"* id=*"message"*>

<p>Opp's,Invalid userid / password..!</p>

</div>

<script type=*"text/javascript"*>

$(document).ready(**function**()

{

$('#UserLogin').trigger('click');

});

</script>

<%}

**if**(no==3)

{%>

<div class=*"error"* id=*"message"*>

<p>Opp's,Something went wrong ..!</p>

</div>

<%}

**if**(no==4)

{%>

<div class=*"success"* id=*"message"*>

<p>You have registered successfully!</p>

</div>

<script type=*"text/javascript"*>

$(document).ready(**function**()

{

$('#UserLogin').trigger('click');

});

</script>

<%}

**if**(no==2)

{%>

<div style="position:*absolute*;top:*170px*;left:*210*" class=*"c"*>

<%

**if**(Utility.parse(request.getParameter("no1"))==5)

{%>

<div class=*"error"* id=*"message"* style="position:*absolute*">

<p>Sorry this user id is already exists.....!</p>

</div>

<%}

**if**(Utility.parse(request.getParameter("no1"))==6)

{%>

<div class=*"error"* id=*"message"* style="position:*absolute*">

<p>Opp's Something Went Wrong.....!</p>

</div>

<%}

%>

<table id=*"login"*>

<h1 align=*"center"* id=*"label"*>User Registeration</h1>

<a href=*"*<%=request.getContextPath() %>*?no=0"*><img id=*"close"* src=*"*<%=request.getContextPath() %>*/Res/Images/close.png"* height=*"25"* width=*"25"* /></a>

<hr size=*"1"* color=*"#111"*></hr>

<form method=*"post"* action=*"*<%=request.getContextPath() %>*/Register"*>

<tr id=*"label"*>

<td>User-ID</td>

<td><input class=*"field"* name=*"id"* type=*"text"* required=*"yes"*/></td>

<td>Address</td>

<td><input class=*"field"* name=*"add"* type=*"text"* required=*"yes"*/></td>

</tr>

<tr>

<td>&nbsp;</td>

</tr>

<tr id=*"label"*>

<td>Password</td>

<td><input class=*"field"* name=*"pass"* type=*"password"* required=*"yes"*/></td>

<td>City</td>

<td><input class=*"field"* name=*"city"* type=*"text"* required=*"yes"*/></td>

</tr>

<tr>

<td>&nbsp;</td>

</tr>

<tr id=*"label"*>

<td>Name</td>

<td><input class=*"field"* name=*"name"* type=*"text"* required=*"yes"*/></td>

<td>Emai-ID</td>

<td><input class=*"field"* name=*"email"* type=*"text"* required=*"yes"*/></td>

</tr>

<tr>

<td>&nbsp;</td>

</tr>

<tr id=*"label"*>

<td>&nbsp;</td>

<td>

&nbsp;&nbsp;&nbsp;Male<input name=*"gender"* value=*"Male"* type=*"radio"* checked=*"true"*/>

&nbsp;&nbsp;&nbsp;Female<input name=*"gender"* value=*"Female"* type=*"radio"*/>

</td>

<td>Contact No</td>

<td><input class=*"field"* name=*"cno"* type=*"text"* required=*"yes"*/></td>

</tr>

<tr>

<td>&nbsp;</td>

</tr>

<tr>

<td>&nbsp;</td>

<td><input class=*"button"* type=*"submit"* value=*"Register"*/></td>

</tr>

</form>

</table>

</div>

<%

}

else

{

%>

<div style="position:*absolute*;top:*170px*;left:*210*">

<p>

</p>

</div>

<%

}

%>

<div style="background-color:*#sadfas*;position:*absolute*;top:*650px*" >

<p><b>&copy; 2012 Celestial V Solutions</b></p>

</div>

</body>

</html>

**8.2 Snapshots**

**LOGIN**

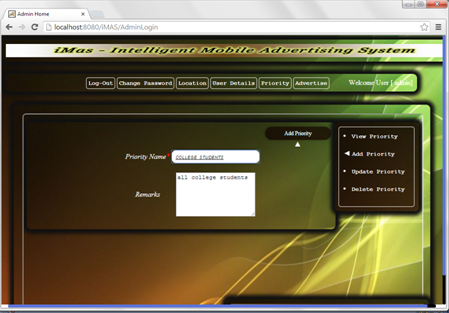


**ADS**

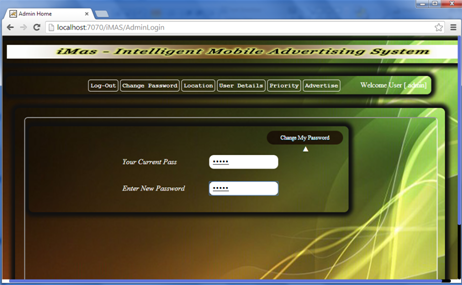


**LOCATION**

**PRIORITY**



**CHANGE PASSWORD**



**USER REGISTRATION**



**LOGIN**



**LOCATION BASED ADS**



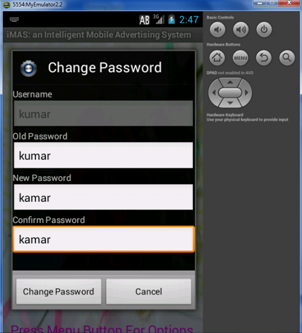
**VIEW CURRENT LOCATION ADS**

****

**VIEW ALL ADS**



**CHANGE PASSWORD**



**Chapter 9**

**BIBLIOGRAPHY**

5. K. Yang, S. Ou, and H.H. Chen, “On Effective Offloading Services for Resource-Constrained Mobile Devices Run-ning Heavier Mobile Internet Applications,” IEEE Comm. Magazine, vol. 46, no. 1, 2008, pp. 56-63.

6. C. Wang and Z. Li, “Parametric Analysis for Adaptive Com-putation Offloading,” ACM SIGPLAN Notices, vol. 39, no. 6, 2004, pp. 119-130.

7. R. Wolski et al., “Using Bandwidth Data to Make Computa-tion Offloading Decisions,” Proc. IEEE Int’l Symp. Parallel and Distributed Processing (IPDPS 08), 2008, pp. 1-8.

8. M. Newborn, “A Parallel Search Chess Program,” Proc. ACM Ann. Conf. Range of Computing: Mid-80’s Perspective, ACM Press, 1985, pp. 272-277.

9. J. Kincaid, “Google Privacy Blunder Shares Your Docs without Permission,” TechCrunch, 7 Mar. 2009; http:// techcrunch.com/2009/03/07/huge-google-privacy-blunder-sharesyour-docs-without-permission.

**Chapter 10**

**An Overview**

**Html**

HTML, an initialism of Hypertext Markup Language, is the predominant markup language for web pages. It provides a means to describe the structure of text-based information in a document — by denoting certain text as headings, paragraphs, lists, and so on — and to supplement that text with interactive forms, embedded images, and other objects. HTML is written in the form of labels (known as tags), surrounded by angle brackets. HTML can also describe, to some degree, the appearance and semantics of a document, and can include embedded scripting language code which can affect the behavior of web browsers and other HTML processors.

HTML is also often used to refer to content of the MIME type text/html or even more broadly as a generic term for HTML whether in its XML-descended form (such as XHTML 1.0 and later) or its form descended directly from SGML

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

**Basic HTML Tags:**

<! -- --> specifies comments

<A>……….</A> Creates hypertext links

<B>……….</B> Formats text as bold

<BIG>……….</BIG> Formats text in large font. <BODY>…</BODY> Contains all tags and text in the HTML document

<CENTER>...</CENTER> Creates text

<DD>…</DD> Definition of a term

<DL>...</DL> Creates definition list

<FONT>…</FONT> Formats text with a particular font

<FORM>...</FORM> Encloses a fill-out form

<FRAME>...</FRAME> Defines a particular frame in a set of frames

<H#>…</H#> Creates heading of different levels(1 – 6)

<HEAD>...</HEAD> Contains tags that specify information about a document

<HR>...</HR> Creates a horizontal rule

<HTML>…</HTML> Contains all other HTML tags

<META>...</META> Provides meta-information about a document

<SCRIPT>…</SCRIPT> Contains client-side or server-side script

<TABLE>…</TABLE> Creates a table

<TD>…</TD> Indicates table data in a table

<TR>…</TR> Designates a table row

<TH>…</TH> Creates a heading in a table

**Attributes**

The attributes of an element are name-value pairs, separated by "=", and written within the start label of an element, after the element's name. The value should be enclosed in single or double quotes, although values consisting of certain characters can be left unquoted in HTML (but not XHTML).Leaving attribute values unquoted is considered unsafe.

Most elements take any of several common attributes: id, class, style and title. Most also take language-related attributes: lang and dir.

The id attribute provides a document-wide unique identifier for an element. This can be used by stylesheets to provide presentational properties, by browsers to focus attention on the specific element or by scripts to alter the contents or presentation of an element. The class attribute provides a way of classifying similar elements for presentation purposes. For example, an HTML document (or a set of documents) may use the designation class="notation" to indicate that all elements with this class value are all subordinate to the main text of the document (or documents). Such notation classes of elements might be gathered together and presented as footnotes on a page, rather than appearing in the place where they appear in the source HTML.

An author may use the style non-attributal codes presentational properties to a particular element. It is considered better practice to use an element’s son- id page and select the element with a style sheet, though sometimes this can be too cumbersome for a simple ad hoc application of styled properties. The title is used to attach sub textual explanation to an element. In most browsers this title attribute is displayed as what is often referred to as a tool tip. The generic inline span element can be used to demonstrate these various non-attributes.

The preceding displays as HTML (pointing the cursor at the abbreviation should display the title text in most browsers).

**Advantages**

* A HTML document is small and hence easy to send over the net.
* It is small because it does not include formatted information.
* HTML is platform independent.
* HTML tags are not case-sensitive.

**JavaScript**

JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then update the browser’s display accordingly

Even though JavaScript supports both client and server Web programming, we prefer JavaScript at Client side programming since most of the browsers supports it. JavaScript is almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags

*<Script> ………. </Script>*

*<Script Language = “JavaScript”>*

*JavaScript statements*

*</Script>*

**Here are a few things we can do with JavaScript**

* Validate the contents of a form and make calculations.
* Add scrolling or changing messages to the Browser’s status line.
* Animate images or rotate images that change when we move the mouse over them.
* Detect the browser in use and display different content for different browsers.
* Detect installed plug-ins and notify the user if a plug-in is required.
* We can do much more with JavaScript, including creating entire application.

**JavaScript and Java are entirely different languages. A few of the most glaring differences are:**

* Java applets are generally displayed in a box within the web document; JavaScript can affect any part of the Web document itself.
* While JavaScript is best suited to simple applications and adding interactive features to Web pages; Java can be used for incredibly complex applications.

There are many other differences but the important thing to remember is that JavaScript and Java are separate languages. They are both useful for different things; in fact they can be used together to combine their advantages.

* JavaScript can be used for Sever-side and Client-side scripting.
* It is more flexible than VBScript.
* JavaScript is the default scripting languages at Client-side since all the browsers supports it.

**Java Technology**

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.

#### 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 serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

#### 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**

Every time you that you download a “normal” program you are risking a viral infection. Prior to java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a “firewall” between a network application and your computer.

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-fly compilation 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

# *Java Source*

## Java byte code

# *Java VM*

***Java***

***.Class***

#### 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.

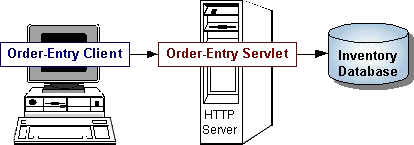
# Introduction to Servlets

Servlets provide a Java(TM)-based solution used to address the problems currently associated with doing server-side programming, including inextensible scripting solutions, platform-specific APIs, and incomplete interfaces.

Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side -- object byte codes that can be dynamically loaded off the net. They differ from applets in that they are faceless objects (without graphics or a GUI component). They serve as platform-independent, dynamically loadable, plug gable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

**What is a Servlet?**

Servlets are modules that extend request/response-oriented servers, such as Java-enabled web servers. For example, a servlet might be responsible for taking data in an HTML order-entry form and applying the business logic used to update a company’s order database.



Servlets are to servers what applets are to browsers. Unlike applets, however, Servlets have no graphical user interface. Servlets can be embedded in many different servers because the servlet API, which you use to write Servlets, assumes nothing about the server's environment or protocol. Servlets have become most widely used within HTTP servers; many web servers support the Servlet API.

**Use Servlets instead of CGI Scripts**

* Servlets are an effective replacement for CGI scripts. They provide a way to generate dynamic documents that is both easier to write and faster to run. Servlets also address the problem of doing server-side programming with platform-specific APIs: they are developed with the Java Servlet API, a standard Java extension.
* So use Servlets to handle HTTP client requests. For example, have Servlets process data posted over HTTPS using an HTML form, including purchase order or credit card data. A servlet like this could be part of an order-entry and processing system, working with product and inventory databases, and perhaps an on-line payment system.

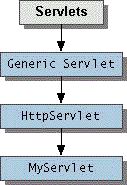
**Other Uses for Servlets**

**Here are a few more of the many applications for Servlets**

* Allowing collaboration between people. A Servlet can handle multiple requests concurrently, and can synchronize requests. This allows Servlets to support systems such as on-line conferencing.
* Forwarding requests. Servlets can forward requests to other servers and Servlets. Thus Servlets can be used to balance load among several servers that mirror the same content, and to partition a single logical service over several servers, according to task type or organizational boundaries.
* Architecture of the Servlet Package
* The javax.servlet package provides interfaces and classes for writing Servlets. The architecture of the package is described below.

**The Servlet Interface**

* The central abstraction in the Servlet API is the Servlet interface. All Servlets implement this interface, either directly or, more commonly, by extending a class that implements it such as HttpServlet.



* The Servlet interface declares, but does not implement, methods that manage the servlet and its communications with clients. Servlet writers provide some or all of these methods when developing a servlet.

**Client Interaction**

* When a servlet accepts a call from a client, it receives two objects:
* A ServletRequest, which encapsulates the communication from the client to the server.
* A ServletResponse, which encapsulates the communication from the servlet back to the client.
* ServletRequest and ServletResponse are interfaces defined by the javax.servlet package.

#### The ServletRequest Interface

* The ServletRequest interface allows the servlet access to: Information such as the names of the parameters passed in by the client, the protocol (scheme) being used by the client, and the names of the remote host that made the request and the server that receive the input stream, ServletInputStream. Servlets use the input stream to get data from clients that use application protocols such as the HTTP POST and PUT methods.

Interfaces that extend ServletRequest interface allow the servlet to retrieve more protocol-specific data. For example, the HttpServletRequest interface contains methods for accessing HTTP-specific header information.

#### The ServletResponse Interface

The ServletResponse interface gives the servlet methods for replying to the client. It

* Allows the servlet to set the content length and MIME type of the reply.
* Provides an output stream, ServletOutputStream, and a Writer through which the servlet can send the reply data.

Interfaces that extend the ServletResponse interface give the servlet more protocol-specific capabilities. For example, the HttpServletResponse interface contains methods that allow the servlet to manipulate HTTP-specific header information.

### Additional Capabilities of HTTP Servlets

The classes and interfaces described above make up a basic Servlet. HTTP Servlets have some additional objects that provide session-tracking capabilities. The servlet writer can use these APIs to maintain state between the servlet and the client that persists across multiple connections during some time period. HTTP Servlets also have objects that provide cookies. The servlet writer uses the cookie API to save data with the client and to retrieve this data.

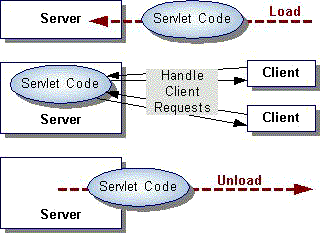
The classes mentioned in the Architecture of the Servlet Package section are shown in the example in bold:

* SimpleServlet extends the HttpServlet class, which implements the Servlet interface.
* SimpleServlet overrides the doGet method in the HttpServlet class. The doGet method is called when a client makes a GET request (the default HTTP request method) and results in the simple HTML page being returned to the client.
* Within the doGet method,   
  An HttpServletRequest object represents the user’s request.
  + An HttpServletResponse object represents the response to the user.
  + Because text data is returned to the client, the reply is sent using the Writer object obtained from the HttpServletResponse object.

##### Servlet Lifecycle

***Each servlet has the same life cycle***

* A server loads and initializes the servlet
* The servlet handles zero or more client requests
* The server removes the servlet



**Initializing a Servlet**

When a server loads a servlet, the server runs the servlet's init method. Initialization completes before client requests are handled and before the servlet is destroyed.

Even though most Servlets are run in multi-threaded servers, Servlets have no concurrency issues during servlet initialization. The server calls the init method once, when the server loads the servlet, and will not call the init method again unless the server is reloading the servlet. The server cannot reload a servlet until after the server has destroyed the servlet by calling the destroy method.

**The init Method:**

The init method provided by the HttpServlet class initializes the servlet and logs the initialization. To do initialization specific to your servlet, override the init () method following these rules:

If an initialization error occurs that renders the servlet incapable of handling client requests, throw an Unavailable Exception.

**Initialization Parameters:**

The second version of the init method calls the getInitParameter method. This method takes the parameter name as an argument and returns a String representation of the parameter's value.

The specification of initialization parameters is server-specific. In the Java Web Server, the parameters are specified with a servlet is added then configured in the Administration Tool. For an explanation of the Administration screen where this setup is performed, see the Administration Tool: Adding Servlets online help document.

In some cases, if we need to get the parameter names, we can use the getParameterNames method.

**Destroying a Servlet**:

Servlets run until the server is destroys them, for example at the request of a system administrator. When a server destroys a servlet, the server runs the servlet's destroy method. The method is run once; the server will not run that servlet again until after the server reloads and reinitializes the servlet.

When the destroy method runs, another thread might be running a service request. The Handling Service Threads at Servlet Termination section shows you how to provide a clean shutdown when there could be long-running threads still running service requests.

**Using the Destroy Method:**

The destroy method provided by the HttpServlet class destroys the servlet and logs the destruction. To destroy any resources specific to your servlet, override the destroy method. The destroy method should undo any initialization work and synchronize persistent state with the current in-memory state.

A server calls the destroy method after all service calls have been completed, or a server-specific number of seconds have passed, whichever comes first. If your servlet handles any long-running operations, service methods might still be running when the server calls the destroy method. You are responsible for making sure those threads complete. The next section shows you how.

The destroy method shown above expects all client interactions to be completed when the destroy method is called, because the servlet has no long-running operations.

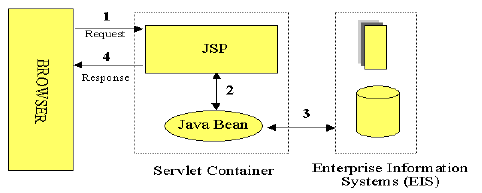
##### 

**Java Server Pages**

Java Server Pages technology lets you put snippets of servlet code directly into a text-based document. A JSP page is a text-based document that contains two types of text: static template data, which can be expressed in any text-based format such as HTML, WML, and XML, and JSP elements, which determine how the page constructs dynamic content.

Java Server Page™ (JSP): An extensible Web technology that uses template data, custom elements, scripting languages, and server-side Java objects to return dynamic content to a client. Typically the template data is HTML or XML elements, and in many cases the client is a Web browser.

According to JSP model1 we can develop the application as,



According to above model the presentation logic has to be implemented in JSP page and the business logic has to be implemented as part of Java bean This model help us in separating the presentation and business logic. For large-scale projects instead of using model1 it is better to use model2 (MVC). Struts framework is based on model 2.

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 for the dynamic parts in special tags, most of which start with "<%" and end with "%>". For example, here is a section of a JSP page that results in something like "Thanks for ordering Core Web Programming

For URL of

*http://host/OrderConfirmation.jsp?title=Core+Web+Programming:*

*Thanks for ordering*

*<I><%= request.getParameter("title") %></I>*

You normally give your file a .jsp extension, and typically install it in any place you could place a normal Web page. Although what you write often looks more like a regular HTML file than a servlet, behind the scenes, the JSP page just gets converted to a normal servlet, with the static HTML simply being printed to the output stream associated with the servlet's service method. This is normally done the first time the page is requested, and developers can simply request the page themselves when first installing it if they want to be sure that the first real user doesn't get a momentary delay when the JSP page is translated to a servlet and the servlet is compiled and loaded. Note also that many Web servers let you define aliases that so that a URL that appears to reference an HTML file really points to a servlet or JSP page.

Aside from the regular HTML, 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. To simplify the scripting elements, you have access to a number of predefined variables such as request in the snippet above.

**J2EE Platform Overview**

The J2EE platform is designed to provide server-side and client-side support for developing distributed, multi-tier applications. Such applications are typically configured as a client tier to provide the user interface, one or more middle-tier modules that provide client services and business logic for an application, and back-end enterprise information systems providing data management.



**Multitier Model**

The J2EE platform provides a multi-tier distributed application model. This means that the various parts of an application can run on different devices. The J2EE architecture defines a client tier, a middle tier (consisting of one or more sub-tier), and a back-end tier. The client tier supports a variety of client types, both outside and inside of corporate firewalls. The middle tier supports client services through Web containers in the Web tier and supports business logic component services through JavaBeans TM. On the back end, the enterprise information systems in the tier are accessible by way of standard APIs.

**Container-Based Component Management**

Central to the J2EE component-based development model is the notion of containers. Containers are standardized runtime environments that provide specific services to components. Components can expect these services to be available on any J2EE platform from any vendor. For example, all J2EE Web containers provide runtime support for responding to client requests, performing request-time processing (such as invoking JSP pages or servlet behavior), and returning results to the client. In addition, they provide APIs to support user session management. All WEB containers provide automated support for transaction and life cycle management of WEB components, as well as bean lookup and other services. Containers also provide standardized access to enterprise information systems; for example, providing access to relational data through the JDBC API. In addition, containers provide a mechanism for selecting application behaviors at assembly or deployment time. Through the use of deployment descriptors (XML files that specify component and container behavior), components can be configured to a specific container’s environment when deployed, rather than in component code. Features that can be configured at deployment time include security checks, transaction control, and other management responsibilities.

While the J2EE specification defines the component containers that a platform implementation must support, it doesn’t specify or restrict the containers’ configurations. Thus, both container types can run on a single platform, Web containers can live on one platform and WEB containers on another or a J2EE platform can be made up of multiple containers on multiple platforms.

**Support for Client Components**

The J2EE client tier provides support for a variety of client types, both within the enterprise firewall and outside. Clients can be offered through Web browsers by using plain HTML pages, HTML generated dynamically by Java Server PagesTM.

**Support for Business Logic Components**

While simple J2EE applications may be built largely in the client tier, business logic is often implemented on the J2EE platform in the middle tier as Java Beans components (also known as enterprise beans). Enterprise beans allow the component or application developer to concentrate on the business logic while the complexities of delivering a reliable, scalable service are handled by the WEB container.

In many ways, the J2EE platform and Java Beans architecture have complementary goals. The Java Beans component model is the backbone of industrial-strength application architectures in the J2EE programming model. The J2EE platform complements the specification by:

* Fully specifying the APIs that an enterprise bean developer can use to implement enterprise beans
* Defining the larger, distributed programming environment in which enterprise beans are used as business logic components

**J2EE Platform Benefits**

With features designed to expedite the process of developing distributed applications, the J2EE platform offers several benefits:

* Simplified architecture and development
* Freedom of choice in servers, tools, and components
* Integration with existing information systems
* Scalability to meet demand variations
* Flexible security model

**Simplified Architecture and Development**

The J2EE platform supports a simplified, component-based development model. Because it is based on the Java programming language and the Java 2 Platform, Standard Edition (J2SETM platform), this model offers “Write-Once-Run-Anywhere” portability, supported by any server product that conforms to the J2EE standard.

The component-based J2EE development model can enhance application development productivity in a number of ways:

* Maps easily to application functionality—Component-based application models map easily and flexibly to the functionality desired from an application. As the examples presented throughout this book illustrate, the J2EE platform provides a variety of ways to configure the architecture of an application, depending on such things as client types required, level of access required to data sources, and other considerations. Component-based design also simplifies application maintenance, since components can be updated and replaced independently—new functionality can be shimmed into existing applications simply by updating selected components.
* Enables assembly- and deploy-time behaviors—Because of the high level of service standardization, much of the code of a J2EE application can be generated automatically by tools, with minimal developer intervention. In addition, components can expect standard services to be available in the runtime environment and can dynamically connect to other components by means of consistent interfaces. As a result, many application behaviors can be configured at application assembly or deployment time, without recoding. Component developers can communicate requirements to application deployers through specific deployment descriptors and settings. Tools can automate this process to further expedite development.
* Supports division of labor—Components help divide the labor of application development among specific skill sets, enabling each member of a development team to focus on his or her ability. Web page authors can create JSP templates, Java programming language coders can implement application behavior, domain experts can develop business logic, and application developers and integrators can assemble and deploy applications. This division of labor also expedites application maintenance. For example, the user interface is the most dynamic part of many applications, particularly on the Web. With the J2EE platform, Web page authors can tweak the look and feel of JSP pages without programmer intervention. The J2EE specifications define a number of roles, including application component provider, application assembler, and application deployer. On some development teams, one or two people may perform all these roles, while on others.

**Integrating Existing Enterprise Information Systems**

The J2EE platform, together with the J2SE platform, includes a number of industries standard APIs for accessing existing enterprise information systems. Basic access to these systems is provided by the following APIs:

The J2EE Connector architecture is the infrastructure for interacting with a variety of Enterprise Information System types, including ERP, CRM, and other legacy systems.

The JDBCTM API is used for accessing relational data from the Java programming language.

The Java Transaction API (JTA) is the API for managing and coordinating transactions across heterogeneous enterprise information systems.

The Java Naming and Directory Interface TM (JNDI) is the API for accessing information in enterprise name and directory services.

The Java Message Service (JMS) is the API for sending and receiving messages via enterprise messaging systems such as IBM MQ Series and TIBCO Rendezvous. In the J2EE platform version 1.3, message-driven beans provide a component-based approach to encapsulating messaging functionality.

Java APIs for XML provide support for integration with legacy systems and applications, and for implementing Web services in the J2EE platform. In addition, specialized access to enterprise resource planning and mainframe systems such as IBM’s CICS and IMS is provided through the J2EE Connector architecture. Since each of these systems is highly complex and specialized, they require unique tools and support to ensure utmost simplicity to application developers.

**Choice of Servers, Tools, and Components**

The J2EE standard and J2EE brand have created a huge marketplace for servers, tools, and components. The J2EE brand on a server product ensures the consistent level of service that is fundamental to the goals of the J2EE platform. At the same time, J2EE standards ensure a lively marketplace for tools and components. Based on past experience and industry momentum, all leading enterprise software vendors are expected to provide the marketplace for J2EE 1.3 products. The standardization and branding of the J2EE platform provides many benefits, including:

* **A range of server choices** - Application development organizations can expect J2EE branded platforms from a variety of vendors, providing a range of choices in hardware platforms, operating systems, and server configurations. This ensures that businesses get a choice of servers appropriate to their needs.
* **Designed for tool support -** Both enterprise beans and JSP page components are designed to be manipulated by graphical development tools and to allow automating many of the application development tasks traditionally requiring the ability to write and debug code. Both J2EE server providers and third-party tool developers have developed tools that conform to J2EE standards and support various application development tasks and styles. Application developers have a choice of tools to manipulate and assemble components, and individual team members may choose tools that best suit their specific requirements.
* **A marketplace for components** - Component-based design ensures that many types of behavior can be standardized, packaged, and reused by any J2EE application. Component vendors will provide a variety of off-the-shelf component solutions, including accounting beans, user interface templates, and even vertical market functionality of interest in specific industries. Application architects get a choice of standardized components to handle common or specialized tasks. The J2EE standard and associated branding programs ensure that solutions are compatible. By setting the stage for freedom of choice, the J2EE platform makes it possible to develop with confidence that the value of your investment will be protected.

**Scales Easily**

J2EE containers provide a mechanism that supports simplified scaling of distributed applications, with no application development effort. Because J2EE containers provide components with transaction support, database connections, life cycle management, and other features that influence performance, they can be designed to provide scalability in these areas. For example, containers may pool database connections, providing clients with quick, efficient access to data. Because containers may run on multiple systems, Web containers can automatically balance load in response to fluctuating demand.

**Simplified, Unified Security Model**

The J2EE security model is designed to support single sign on access to application services. Component developers can specify the security requirements of a component at the method level to ensure that only users with appropriate permissions can access specific data operations. While both Java Beans technology and Java Servlet APIs provide programmatic security control, the basic role-based security mechanism (where groups of users share specific permissions) is specified entirely at application deployment time. This provides both greater flexibility and better security control.

**J2EE Application Scenarios**

The J2EE specifications encourage architectural diversity. The J2EE specifications and technologies make few assumptions about the details of API implementations. The application-level decisions and choices are ultimately a trade-off between functional richness and complexity. The J2EE programming model is flexible enough for applications that support a variety of client types, with both the Web container and WEB container as optional.



The following enterprise requirements heavily influenced the choices made in developing the sample application:

* The need to make rapid and frequent changes to the “*look and feel*” of the application.
* The need to partition the application along the lines of presentation and business logic so as to increase modularity
* The need to simplify the process of assigning suitably trained human resources to accomplish the development task such that work can proceed along relatively independent but cooperating tracks

The need to have developers familiar with back-office applications unburdened from GUI and graphic design work, for which they may not be ideally qualified.

The need to have the necessary vocabulary to communicate the business logic to teams concerned with human factors and the aesthetics of the application.

The ability to assemble back-office applications using components from a variety of sources, including off-the-shelf business logic components .

The ability to deploy transactional components across multiple hardware and software platforms independently of the underlying database technology.

The ability to externalize internal data without having to make many assumptions about the consumer of the data and to accomplish this in a loosely coupled manner Clearly, relaxing any or all of these requirements would influence some of the application-level decisions and choices that a designer would make. Although it is reasonable to speak of “throw-away” presentation logic (that is, applications with a look and feel that ages rapidly), there is still significant inertia associated with business logic. This is even truer in the case of database schemas and data in general. It is fair to say that as one moves further away from EIS resources, the volatility of the application code increases dramatically; that is, the code’s “shelf life” drops significantly.

**Multitier Application Scenario**

JSP pages, supported by Servlets, generate dynamic Web content for delivery to the client. The Web container hosts application components that use EIS resources to service requests from Web-tier components. This architecture decouples data access from the application’s user interface. The architecture is also implicitly scalable. Application back-office functionality is relatively isolated from the end-user look and feel.



It is worth noting that XML plays an integral role in this scenario. The ability to both produce and consume XML data messages in the Web container is an extremely flexible way to embrace a diverse set of client types. These platforms range from general purpose XML-enabled browsers to specialized XML rendering engines targeting vertical solutions. XML data messages typically use HTTP as their transport protocol. Java and XML are complementary technologies: The Java language offers portable code, XML provides portable data. In the Web tier, the question of whether to use JSP pages or Servlets comes up repeatedly. JSP technology is intended for application user interface components, while Java Servlets are preferred for request processing and application control logic. Servlets and JSP pages work together to provide dynamic content from the Web tier.

**Stand-Alone Client Scenario**

****

**The stand-alone client may be one of three types:**

WEB clients interacting directly with enterprise beans hosted in an WEB container within an WEB server, as shown in Figure 1.5. This scenario uses RMI-IIOP, and the WEB server accesses EIS resources using JDBC and the J2EE Connector architecture.

* Stand-alone clients, implemented in the Java language or another programming language, consuming dynamic Web content (usually XML data messages). In this scenario, the Web container essentially handles XML transformations and provides Web connectivity to clients. Presentation logic occurs in the client tier. The Web tier handles business logic and may directly access EIS resources. Ideally, business logic is implemented as enterprise beans to take advantage of the rich enterprise beans component model.
* Stand-alone Java application clients accessing enterprise information system resources directly using JDBC or Connectors. In this scenario, presentation and business logic are co-located on the client platform and may in fact be tightly integrated into a single application. This scenario is classic two-tier client-server architecture, with its associated distribution, maintenance, and scalability issues.

**Web-Centric Application Scenario**

****

There are a number of scenarios in which the use of enterprise beans in an application would be considered overkill: *sort of like using a sledgehammer to crack a nut*. The J2EE specification doesn’t mandate a specific application configuration, nor could it realistically do so. The J2EE platform is flexible enough to support the application configuration most appropriate to a specific application design requirement.

The Web container hosts both presentation and business logic, and it is assumed that JDBC and the J2EE Connector architecture are used to access EIS resources.



In many cases, J2EE platform providers may co-locate their Web and WEB containers, running them within the same Java Virtual Machine (JVM). J2EE applications deployed on such an implementation are still considered Multitier applications, because of the division of responsibilities that the separate technologies imply.

**Business-to-Business Scenario**

This scenario focuses on peer level interactions between both Web and WEB containers. The J2EE programming model promotes the use of XML data messaging over HTTP as the primary means of establishing loosely coupled communications between Web containers. This is a natural fit for the development and deployment of Web-based commerce solutions.

The peer-level communications between WEB containers is currently a more tightly coupled solution most suitable for intranet environments. With support for JMS and message-driven beans, the J2EE 1.3 platform makes developing loosely coupled intranet solutions increasingly practical.

Future releases of the J2EE platform will provide additional functionality in the form of Java APIs for XML, which enable more complete support for loosely coupled applications through XML-based Web services.

**Communication Technologies**

Communication technologies provide mechanisms for communication between clients and servers and between collaborating objects hosted by different servers. The J2EE specification requires support for the following types of communication technologies:

**Internet protocols**

* Remote method invocation protocols
* Object Management Group protocols
* Messaging technologies
* Data formats

Internet protocols define the standards by which the different pieces of the J2EE platform communicate with each other and with remote entities. The J2EE platform supports the following Internet protocols:

* TCP/IP—Transport Control Protocol over Internet Protocol. These two protocols provide for the reliable delivery of streams of data from one host to another. Internet Protocol (IP), the basic protocol of the Internet, enables the unreliable delivery of individual packets from one host to another. IP makes no guarantees as to whether the packet will be delivered, how long it will take, or if multiple packets will arrive in the order they were sent. The Transport Control Protocol (TCP) adds the notions of connection and reliability.
* HTTP 1.0—Hypertext Transfer Protocol. The Internet protocol used to fetch hypertext objects from remote hosts. HTTP messages consist of requests from client to server and responses from server to client.
* SSL 3.0—Secure Socket Layer. A security protocol that provides privacy over the Internet. The protocol allows client-server applications to communicate in a way that cannot be eavesdropped or tampered with. Servers are always authenticated and clients are optionally authenticated.

**Remote Method Invocation Protocols**

Remote Method Invocation (RMI) is a set of APIs that allow developers to build distributed applications in the Java programming language. RMI uses Java language interfaces to define remote objects and a combination of Java serialization technology and the Java Remote Method Protocol (JRMP) to turn local method invocations into remote method invocations. The J2EE platform supports the JRMP protocol, the transport mechanism for communication between objects in the Java language in different address spaces.

**Object Management Group Protocols**

Object Management Group (OMG) protocols allow objects hosted by the J2EE platform to access remote objects developed using the OMG’s Common Object Request Broker Architecture (CORBA) technologies and vice versa. CORBA objects are defined using the Interface Definition Language (IDL). An application component provider defines the interface of a remote object in IDL and then uses an IDL compiler to generate client and server stubs that connect object implementations to an Object Request Broker (ORB), a library that enables CORBA objects to locate and communicate with one another. ORBs communicate with each other using the Internet Inter-ORB Protocol (IIOP). The OMG technologies required by the J2EE platform are Java IDL and RMI-IIOP.

**Java IDL**

Java IDL allows Java clients to invoke operations on CORBA objects that have been defined using IDL and implemented in any language with a CORBA mapping. Java IDL is part of the J2SE platform. It consists of a CORBA API and ORB. An application component provider uses the idlj IDL compiler to generate a Java client stub for a CORBA object defined in IDL. The Java client is linked with the stub and uses the CORBA API to access the CORBA object.

**RMI-IIOP**

RMI-IIOP is an implementation of the RMI API over IIOP. RMI-IIOP allows application component providers to write remote interfaces in the Java programming language. The remote interface can be converted to IDL and implemented in any other language that is supported by an OMG mapping and an ORB for that language. Clients and servers can be written in any language using IDL derived from the RMI interfaces. When remote interfaces are defined as Java RMI interfaces, RMI over IIOP provides interoperability with CORBA objects implemented in any language.

**RMI-IIOP contains:**

* The rmic compiler, which generates: - Client and server stubs that work with any ORB. An IDL file compatible with the RMI interface. To create a C++ server object, an application component provider would use an IDL compiler to produce the server stub and skeleton for the server object.
* A CORBA API and ORB. Application clients must use RMI-IIOP to communicate with enterprise beans.

# MySql

**Introduction**

***MySql*** is a relational database management system, which organizes data in the form of tables. MySQL is one of many databases servers based on RDBMS model, which manages a seer of data that attends three specific things-data structures, data integrity and data manipulation. With MySQL cooperative server technology we can realize the benefits of open, relational systems for all the applications. MySQL makes efficient use of all systems resources, on all hardware architecture; to deliver unmatched performance, price performance and scaleability.Any DBMS to be called as RDBMS has to satisfy Dr.E.F.Codd’s rules.

**Distinct Features of MySql:**

* **MYSQL IS PORTABLE:**

The MySQL RDBMS is available on wide range of platforms ranging from PCs to super computers and as a multi user loadable module for Novel NetWare, if you develop application on system you can run the same application on other systems without any modifications.

* **MYSQL IS COMPATIBLE:**

MySQL commands can be used for communicating with IBM DB2 mainframe RDBMS that is different from MySQL , that is MySQL compatible with DB2 .MySQL RDBMS is a high performance fault tolerant DBMS , which is specially designed for online transaction processing and for handling large database applications.

* **MULTITHREADED SERVER ARCHITECTURE:**

MySQL adaptable multithreaded server architecture delivers scalable high performance for very large number of users on all hardware architecture including symmetric multiprocessors (sumps) and loosely coupled multiprocessors. Performance is achieved by eliminating CPU, I/O, memory and operating system bottlenecks and by optimizing the MySQL DBMS server code to eliminate all internal bottlenecks.

**FEATURES OF MYSQL**

***Most popular RDBMS in the market because of its ease of use***

* Client/server architecture.
* Data independence.
* Ensuring data integrity and data security.
* Managing data concurrency.
* Parallel processing support for speed up data entry and online transaction processing used for applications.
* DB procedures, functions and packages.

**Dr.E.F.OCDD’s RULES**

These rules are used for valuating a product to be called as relational database management systems. Out of 12 rules, a RDBMS product should satisfy at least 8 rules +rule called rule 0 that must be satisfied.

**RULE 0: FOUNDATION RULE:**

For any system that is to be advertised as ,or claimed to be relational DBMS. That system should manage database with in self, with out using an external language.

**RULE 1.INFORMATION RULE**

All information in relational database is represented at logical level in only one way as values in tables.

**RULE 2.GUARANTEED ACCESS:**

Each and every data in a relational database is guaranteed to be logically accessibility by using to a combination of table name, primary key value and column name

**RULE 3. SYSTEMATIC TREATMENT OF NULL VALUES**

Null values are supported for representing missing information and inapplicable information. They must be handled in systematic way, independent of data types.

**RULE 4. DYNAMIC ONLINE CATALOG BASED RELATION MODEL:**

The database description is represented at the logical level in the same way as ordinary data so that authorized users can apply the same relational language to its interrogation as they do to the regular data.

**RULE 5: COMPRHENSIVE DATA SUB LANGUAGE**

A relational system may support several languages and various models of terminal use. However there must be one language whose statement can express all of the following:

*Data Definitions, View Definitions, Data Manipulations, Integrity, Constraints, Authorization and transaction boundaries.*

**RULE 6: VIEW UPDATING**

Any view that is theoretically that updatable if changes can be made to the tables that effect the desired changes in the view.

**RULE 7.HIGH LEVEL UPDATE, INSERT and DELETE**

The capability of handling a base relational or derived relational as a single operand applies not only retrieval of data also to its insertion, updating, and deletion.

**RULE 8.PHYSICAL DATA INDEPENDENCE**

Application program and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access method.

**RULE 9.LOGICAL DATA INDEPENDENCE**

Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage repres3entation or access methods.

**RULE 10: INTEGRITY INDEPENDENCE:**

Integrity constraints specific to particular database must be definable in the relational data stored in the catalog, not in application program.

**RULE 11: DISTRIBUTED INDEPENDENCE:**

Weather or not a system support data base distribution, it must have a data sub-language that can support distributed databases without changing the application program.

**RULE 12: NON SUB-VERSION:**

If a relational system has low level language, that low language cannot use to subversion or by pass the integrity rules and constraints expressed in the higher level relational language.

**MYSQL SUPPORTS THE FOLLOWING CODD’S RULES:**

**Rule 1:** Information Rule (Representation of information)-YES.

**Rule 2:** Guaranteed Access-YES.

**Rule 3:** Systematic treatment of Null values-YES.

**Rule 4:** Dynamic on-line catalog-based Relational Model-YES.

**Rule 5:** Comprehensive data sub language-YES.

**Rule 6:** View Updating-PARTIAL.

**Rule 7:** High-level Update, Insert and Delete-YES.

**Rule 8:** Physical data Independence-PARTIAL.

**Rule 9:** Logical data Independence-PARTIAL.

**Rule 10:** Integrity Independence-PARTIAL.

**Rule 11:** Distributed Independence-YES.

**Rule 12:** Non-subversion-YES.

**APPENDIX-4**

**Execution steps**

* Copy web application directory and paste it to **C:\ProgramFiles\Apache Software Foundation\Tomcat**

**5.0\webapps** in your local machine.

* Export the **DB\_BackUp**\**dump.sql** into my-SQL database.
* Please copy the Supporting\_Files from Supporting\_Files folder and paste it in (C:\WINDOWS) Directory.

**Note :**

* Before run this Project, Please copy the Images folder from this project and paste it in (C :) Directory.
* To start your tomcat server go to start🡪All Programs🡪Apache Tamcat 5.0🡪Configure Tomcat, and the click start button in Configure Tomcat window.
* To check whether tomcat in started or not. Just open your browser and type http://localhost:8080, Tomcat home page will be open.
* For the best view use IE-6.0 browser.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## *Login Details*

**Mod ID and Password for Moderator Login :**

User Id :admin

# Password : admin

**User ID and Password for Member Login :**

User Id : user

# Password : user

**\*** There is no user id and password for end users. Just type

**http://localhost:8080** in your

browser. Then end user page will open.