SMART CITY

Abstract

Contents

Abstract

If you are desirous to give a web-based platform to your city to be showcased for all it has in terms of unique business, places, jobs, bus routes, emergency information, historical places, static and dynamic maps provided city over view, political and social information, things to do, secure registration of users, facility to provide communication between users.

.

Introduction

Vision

The main aim of this project services provided to the users who have registered in the site. The services regarding to city political, historical, conventional places, busroutes, business companies profile and jobs details.

Scope

The scope of the **Smart City** is as follows:

The system scope includes the following:

This project is aimed to automate the operations of **Smart City. The** system is an online application that can be accessed throughout the internet. This provides information to the customer. This information regarding to history, political information of the city. Business, political, news, papers and TV Channels information also.

Overview

Overall description consists of background of the entire specific requirement. It also gives explanation about actor and function which is used. It gives explanation about architecture diagram and it also gives what we are assumed and dependencies. It also supports specific requirement and also it support functional requirement, supplementary requirement other than actor which is used. It also gives index and appendices. It also gives explanation about any doubt and queries.

This system can be used as an application for the handicraft employees and customers of the Handicraft Company to manage the product information. Customer logging should be able to upload the information of the any required product. Customers/Company representatives logging in may also access/search any information of product related.

System Analysis

Existing System:

- The existing system is a manual system. Here the city information needs to save his information in the form of excel sheets or Disk Drives.
- There is no sharing is possible if the data is in the form of paper or Disk drives.
- The manual system gives us very less security for saving data; some data may be lost due to mismanagement.
- It's a limited system and fewer users friendly.
- Searching of particular information is very critical it takes lot of time.

Proposed System:

The development of this new system contains the following activities, which try to automate the entire process keeping in the view of database integration approach.

- User Friendliness is provided in the application with various controls provided by system Rich User Interface.
- The system makes the overall project management much easier and flexible.
- It can be accessed over the Intranet.
- The city information files can be stored in centralized database which can be maintained by the system.

Feasibility Study

Economic Feasibility

Economic feasibility attempts 2 weigh the costs of developing and implementing

a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system.

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision-making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

Operational Feasibility

Proposed project is beneficial only if it can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.

Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.

Has the user been involved in the planning and development of the project?

Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project.

Since the proposed system was to help reduce the hardships encountered. In

the existing manual system, the new system was considered to be operational feasible.

Technical Feasibility

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, .at this point in time, not too many detailed designs of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis.

Understand the different technologies involved in the proposed system before commencing the project we have to be very clear about what are the technologies that are to be required for the development of the new system. Find out whether the organization currently possesses the required technologies. Is the required technology available with the organization?

Software Requirement Specification

Software Requirements

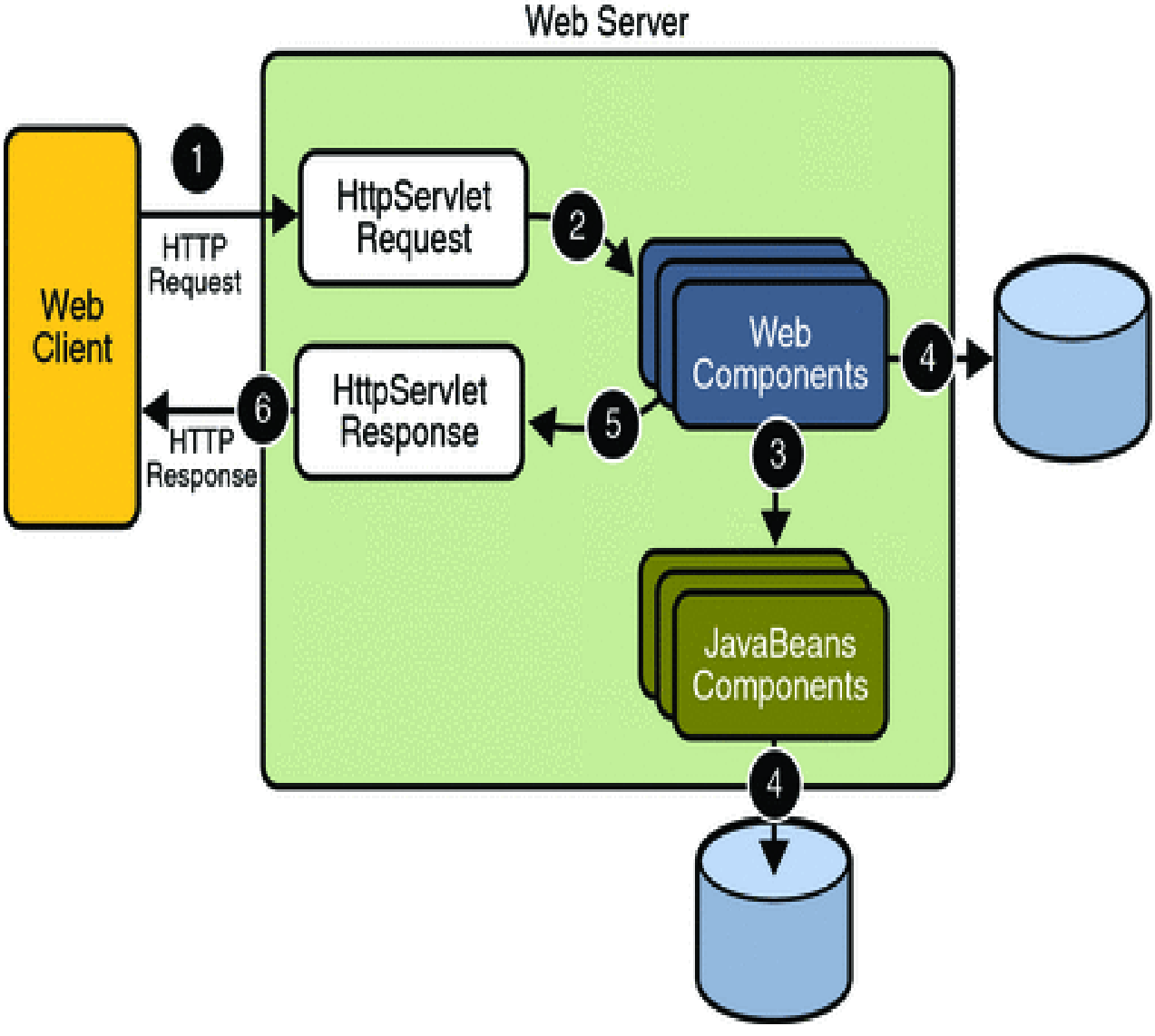
Operating System	:	Windows XP/2003 or Linux/Solaris
User Interface	:	HTML, CSS
Client-side Scripting	:	JavaScript
Programming Language	:	Java
Web Applications	:	JDBC, Servlets, JSP
IDE/Workbench	:	Eclipse with MyEclipse Plug-in
Database	:	Oracle/Access
Server Deployment	:	Apache Tomcat 5.x

Hardware Requirements

Processor	:	Pentium IV
Hard Disk	:	40GB
RAM	:	256MB

System Design

Architecture Diagram



Authentication

Functional Description

a. Login to the system through the first page of the application.
b. Change the password after login to the application.
c. See his/her details and change it.
d. Help from the system.

Functions

Maintenance

Modules Description:

User in System:

The system functionalities are full filled by following users
1. Administrator
2. General Users

Administrator Module:

- The details regarding complete history such as political and social is entered by admin and he want he can edit the details once entered.
- The details regarding business details such as top companies in the city and its information are entered and he want he can edit the details once entered
- The details regarding job details such as job title, vacancies and companies' profile is entered by admin and he can edit the details once entered

- The details regarding regarding emergency such as phone numbers of with respect to emergency is entered by admin and he can edit the details one he entered
- The details regarding regarding conventional places such as description an, location, address and image of the place entered by him and he can edit the details once entered
- The details regarding news papers, city channels such which papers are available in the city entered by admin and he can edit the details once he entered.
- The details regarding political leaders entered by admin and he can edit the details once he entered.

Users Module:

- Registration of the users
- Self management of the profile
- Searching city information about tourism, business, history etc
- He can perform communication between users.
- He can view the city information by using static and dynamic maps.
- He can know the information about city news papers and city channels by this system
- He can know the information about bus routes by this system

Reports Module:

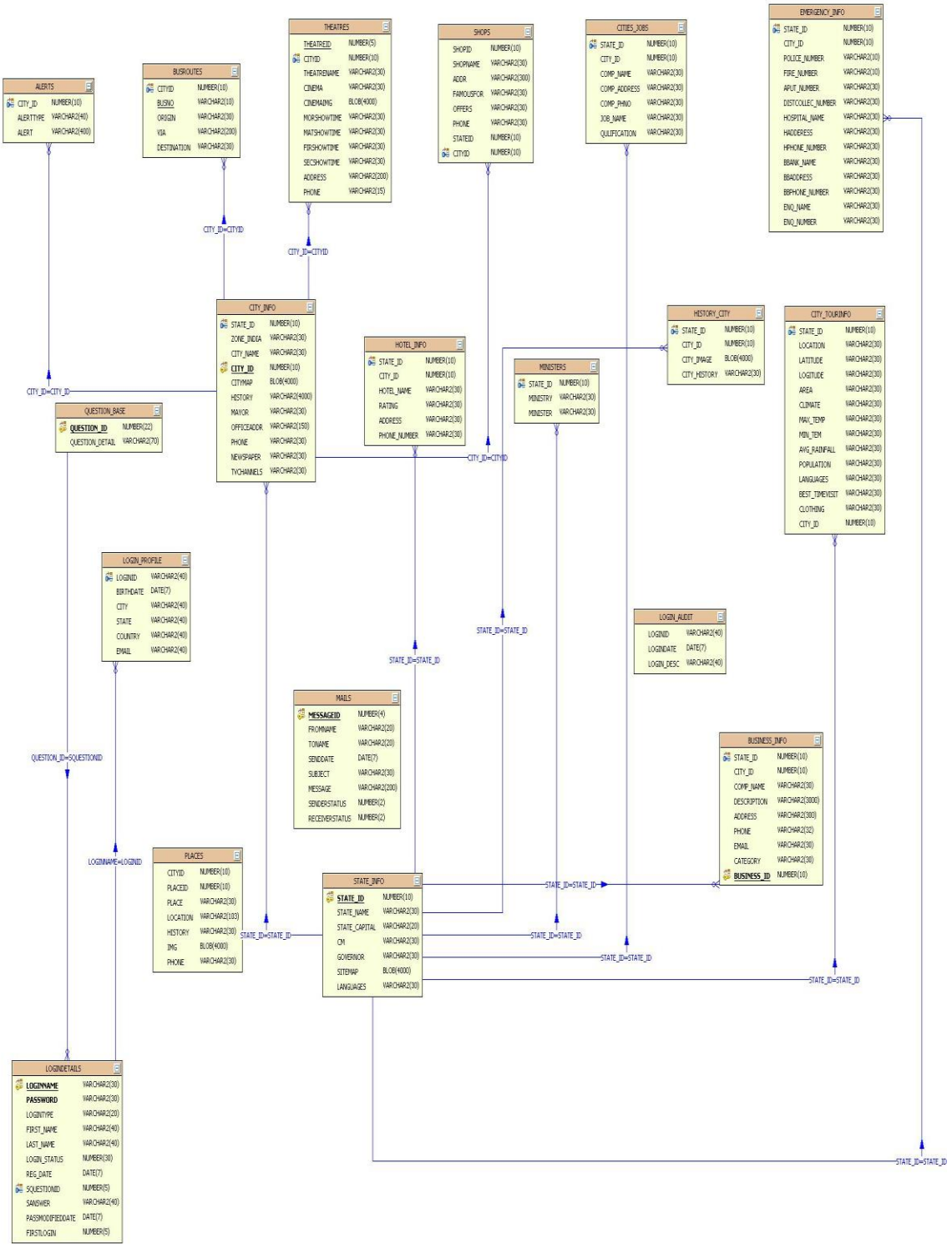
- Analyses reports business wise, tourists wise.
- Analysis reports area wise, sex wise, city health, business & agriculture growth etc.
- Emergency Report.
- City particulars Report

Mails

This module provides the mail system to every user in the system who are authenticated.

- He can view the inbox to know about the emails he got.
- He can view the sent mails to know the mails he sent.
- He can send the mail to another user.

E – R Diagrams



DATA FLOW DIAGRAMS

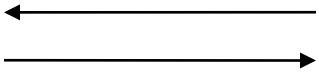
DATA FLOW DIAGRAMS:

A graphical tool used to describe and analyze the moment of data

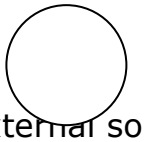
through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts. The Basic Notation used to create a DFD's are as follows:

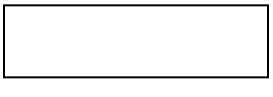
1. Dataflow: Data move in a specific direction from an origin to a destination.



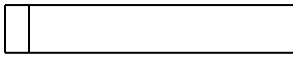
2. Process: People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.



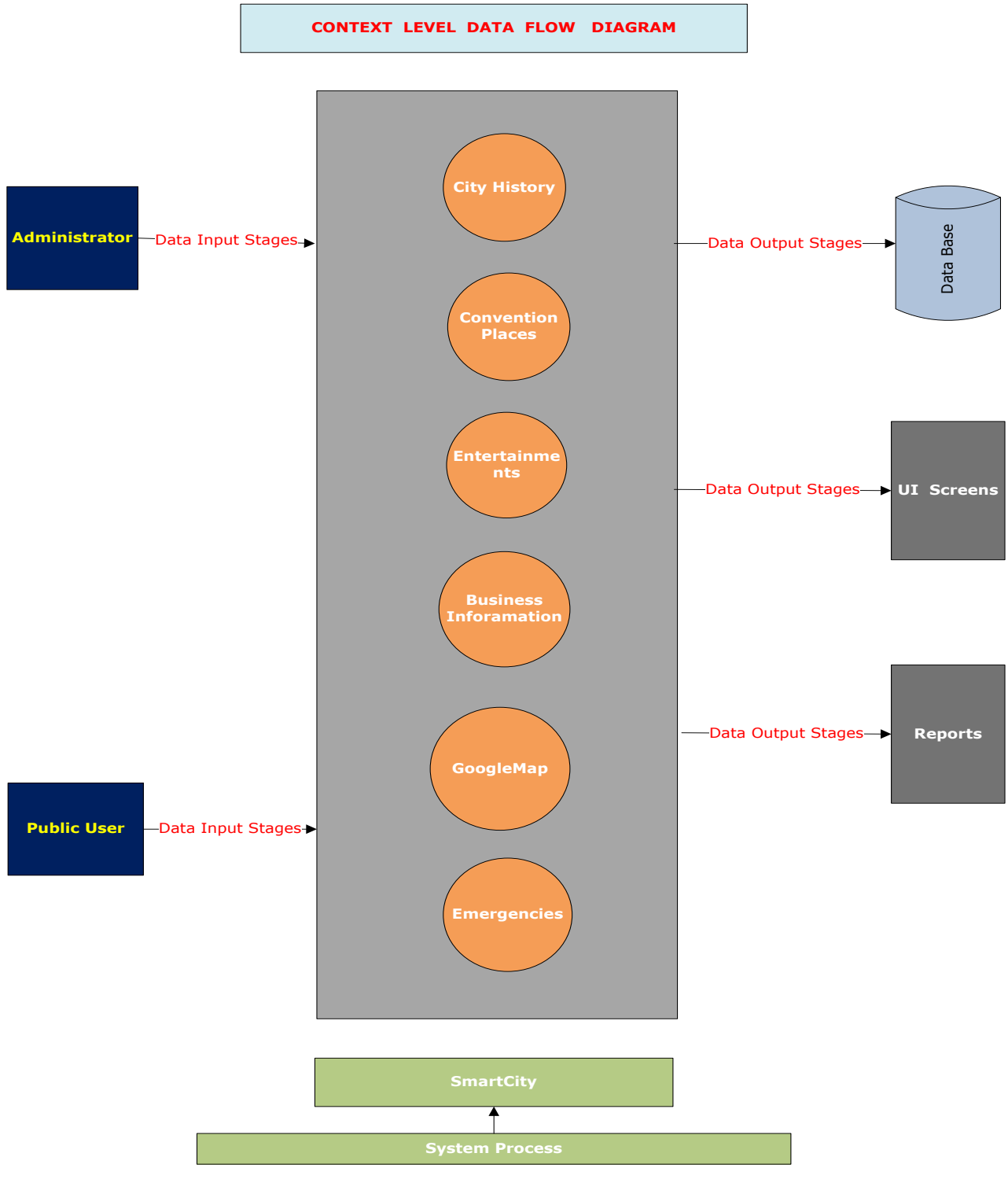
3. Source: External sources or destination of data, which may be People, programs, organizations or other entities.



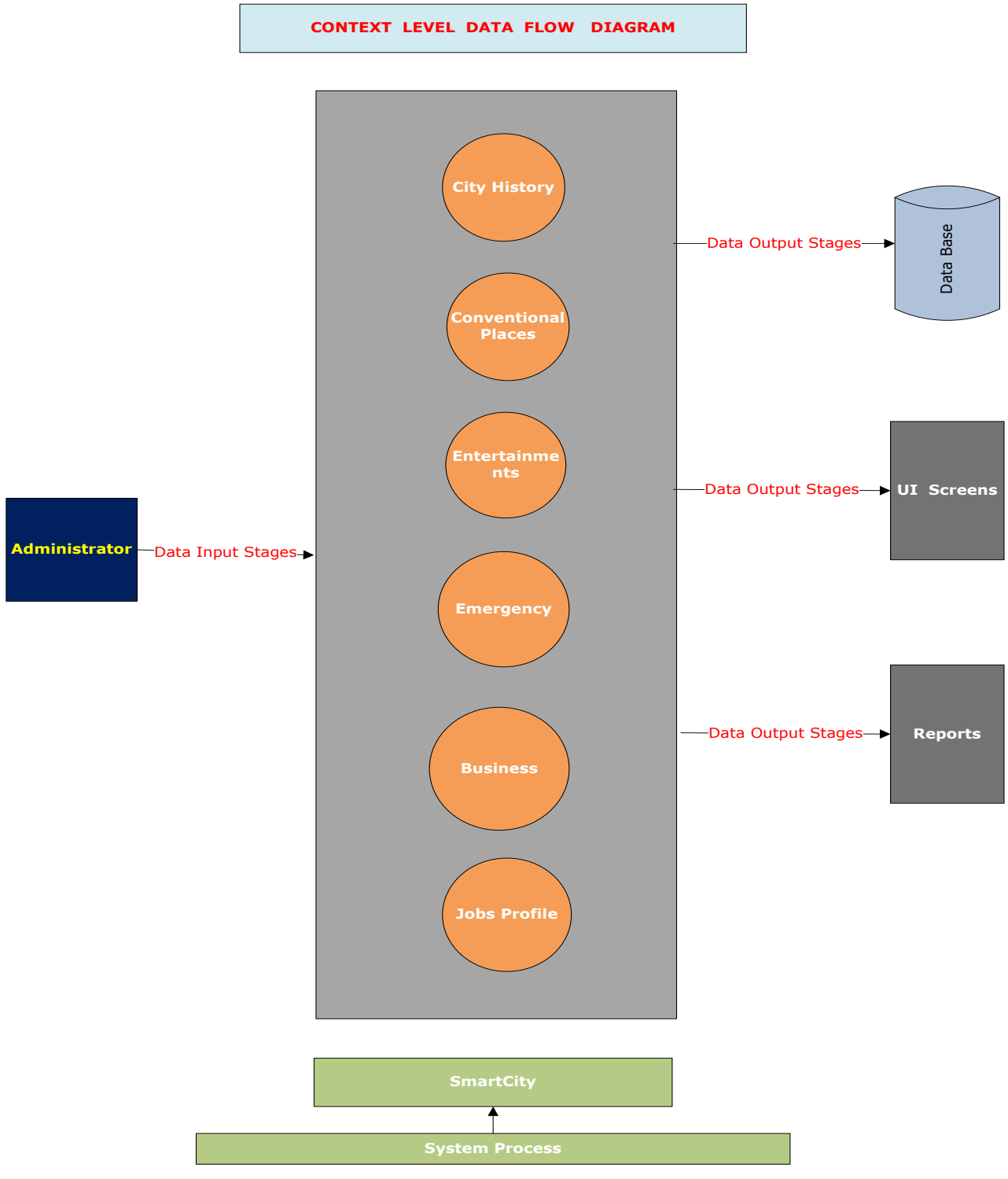
4. Data Store: Here data are stored or referenced by a process in the System.



DATA FLOW DIAGRAM:

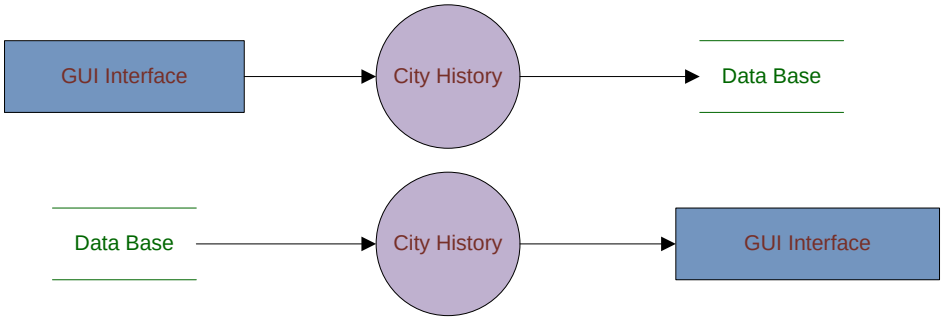

CONTEXT LEVEL DATA FLOW DIAGRAM
Administrator
Data Input Stages
City History
Convention Places
Entertainments
Business Inforamation
GoogleMap
Emergencies
Data Output Stages
Data Base
Data Output Stages
UI Screens
Data Output Stages
Reports
Public User
Data Input Stages
SmartCity
System Process

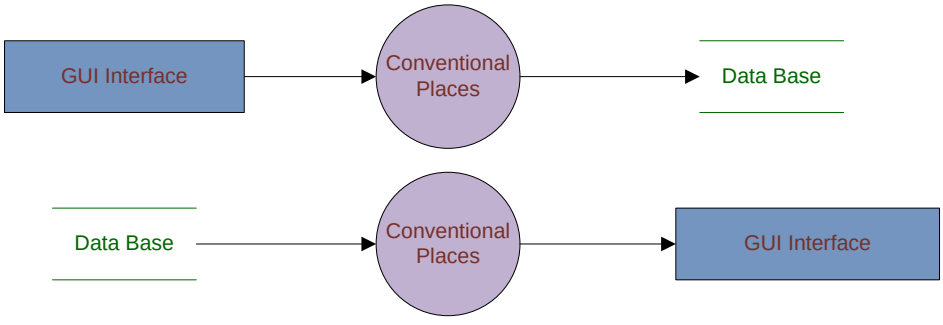
LEVEL-1:

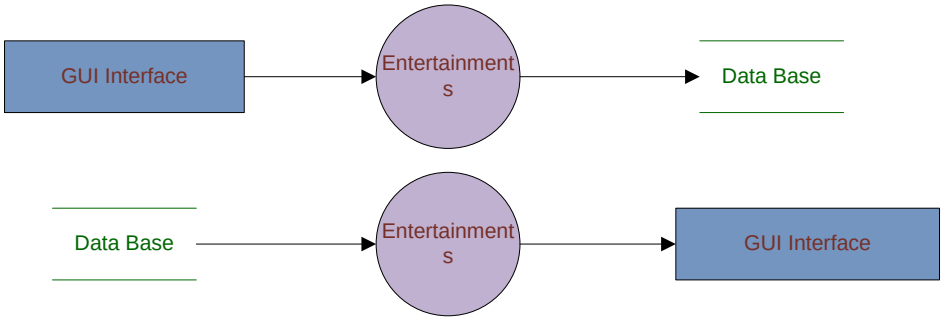


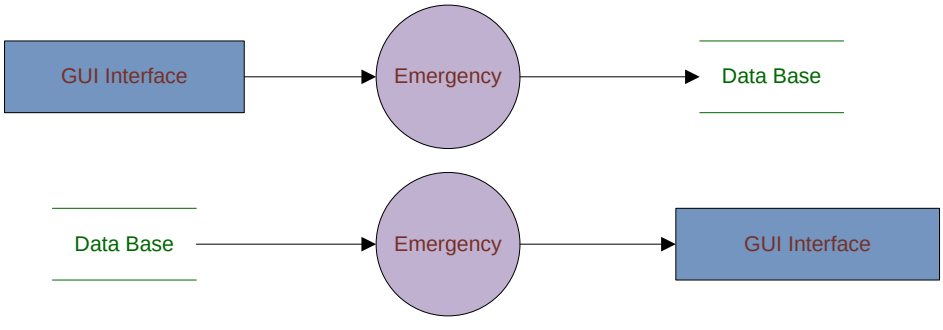
LEVEL-2:

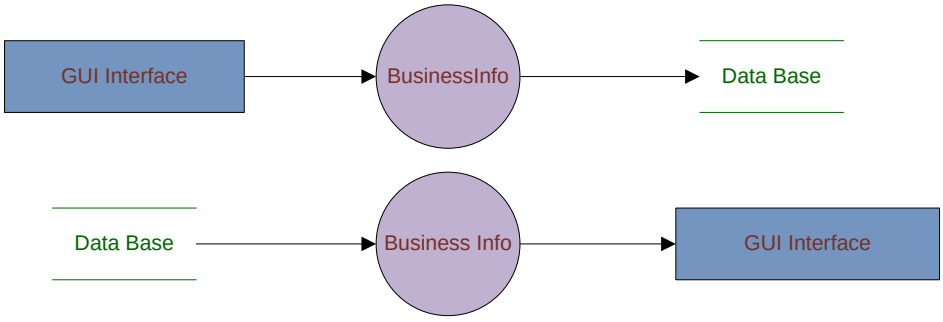
Administrator:

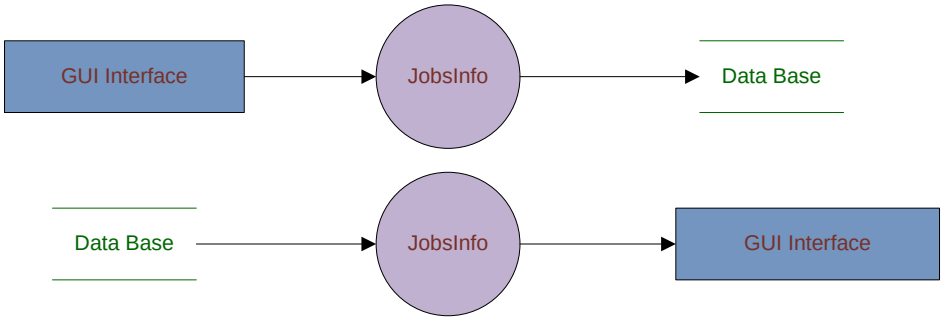

GUI Interface
City History
Data Base
Data Base
City History
GUI Interface


GUI Interface
Conventional Places
Data Base
Data Base
Conventional Places
GUI Interface


GUI Interface
Entertainments
Data Base
Data Base
Entertainments
GUI Interface

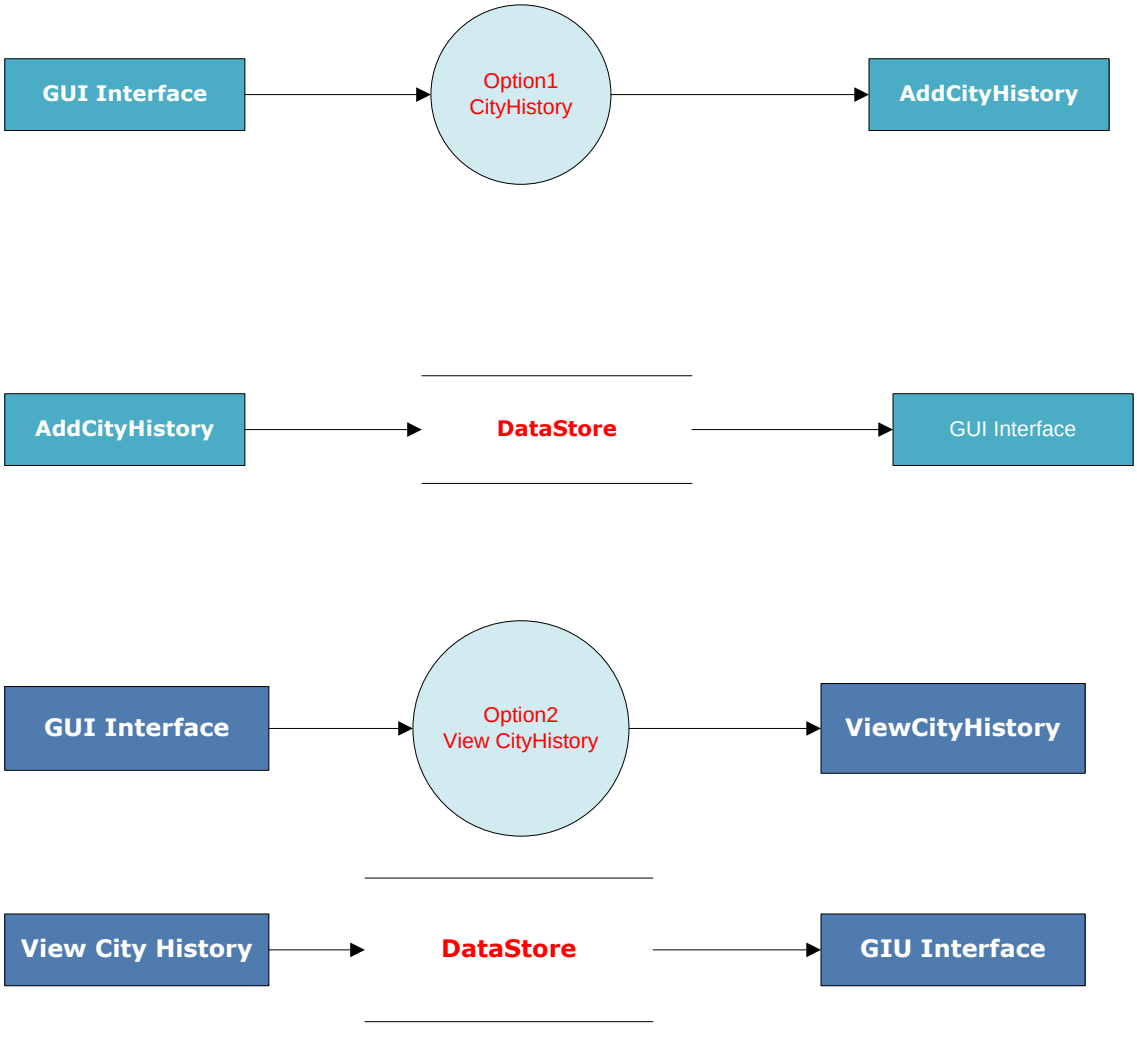

GUI Interface
Emergency
Data Base
Data Base
Emergency
GUI Interface

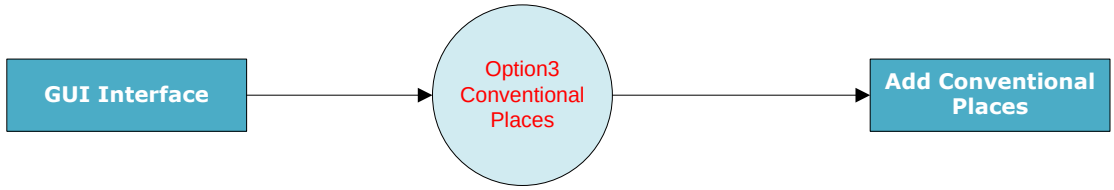

GUI Interface
BusinessInfo
Data Base
Data Base
Business Info
GUI Interface


GUI Interface
JobsInfo
Data Base
Data Base
JobsInfo
GUI Interface

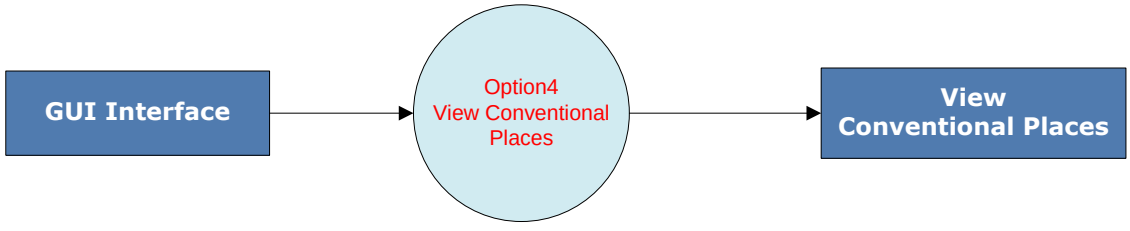
LEVEL-3:

ADMINISTRATOR:

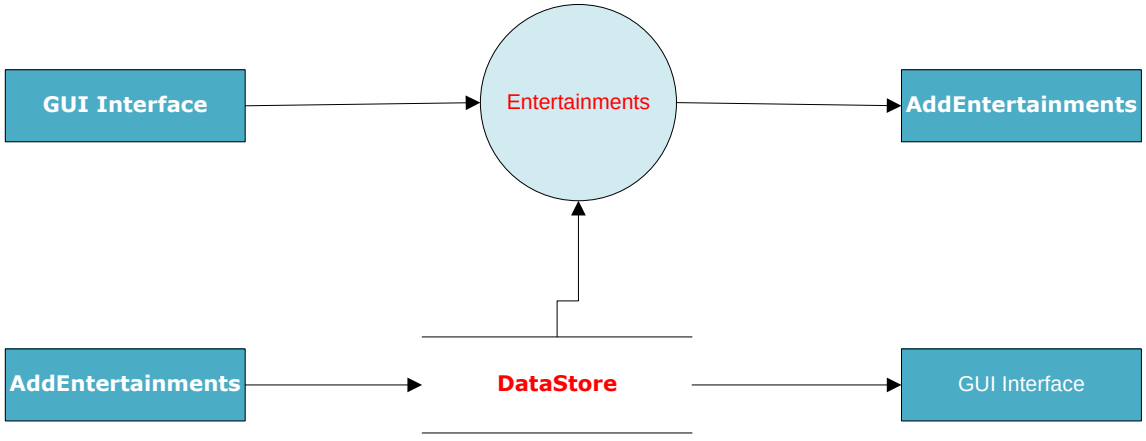


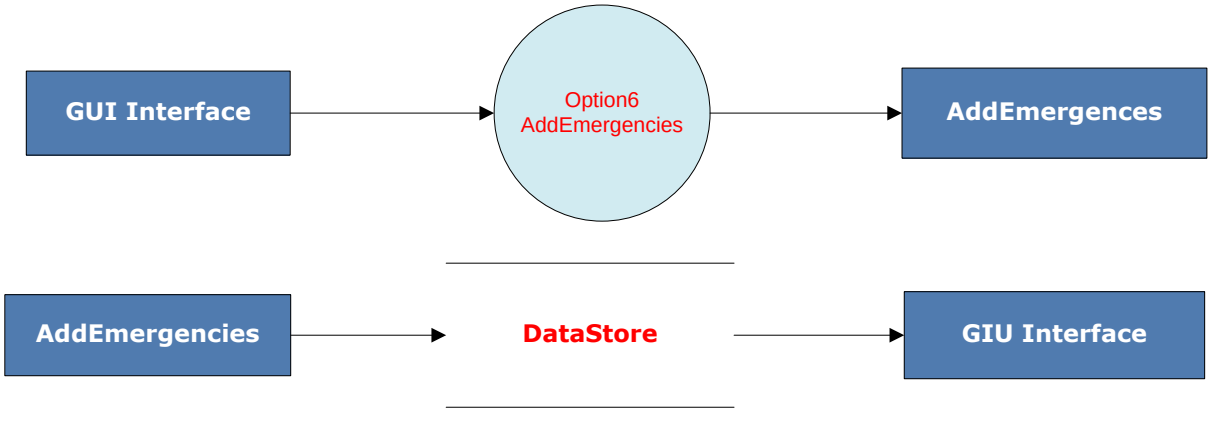

GUI Interface
Option3
Conventional
Places
Add Conventional
Places

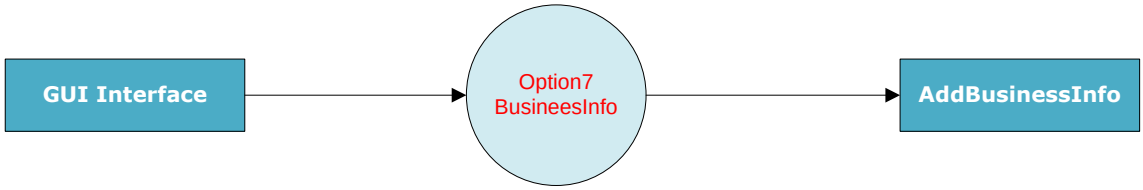

Add Conventional
Places
DataStore
GUI Interface


GUI Interface
Option4
View Conventional
Places
View
Conventional Places

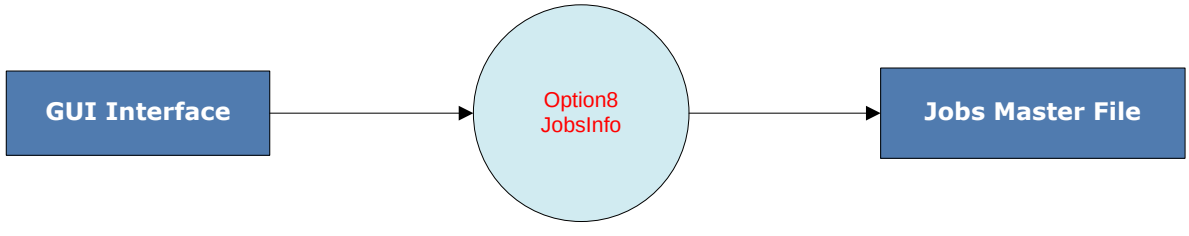

View Conventional
File
DataStore
GIU Interface


GUI Interface
Entertainments
AddEntertainments
AddEntertainments
DataStore
GUI Interface

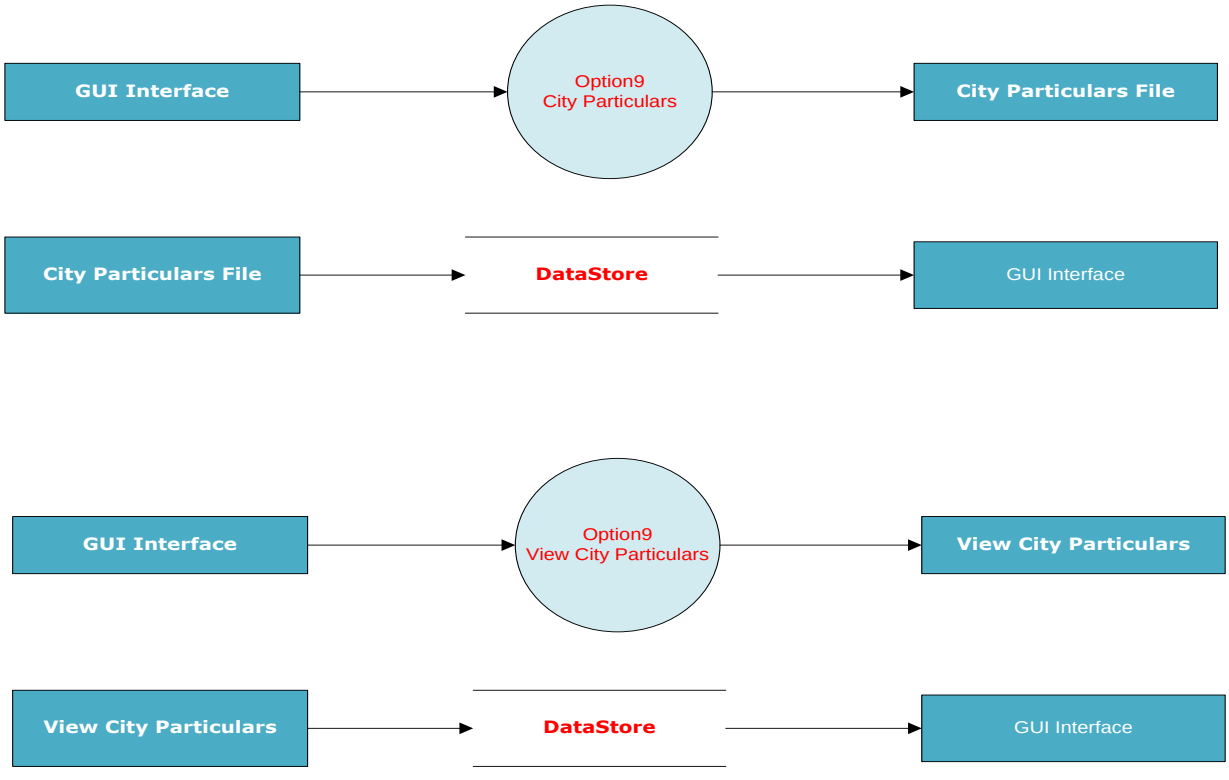

GUI Interface
Option6
AddEmergencies
AddEmergences
AddEmergencies
DataStore
GIU Interface


GUI Interface
Option7
BusineesInfo
AddBusinessInfo


Business Info
DataStore
GUI Interface


GUI Interface
Option8
JobsInfo
Jobs Master File

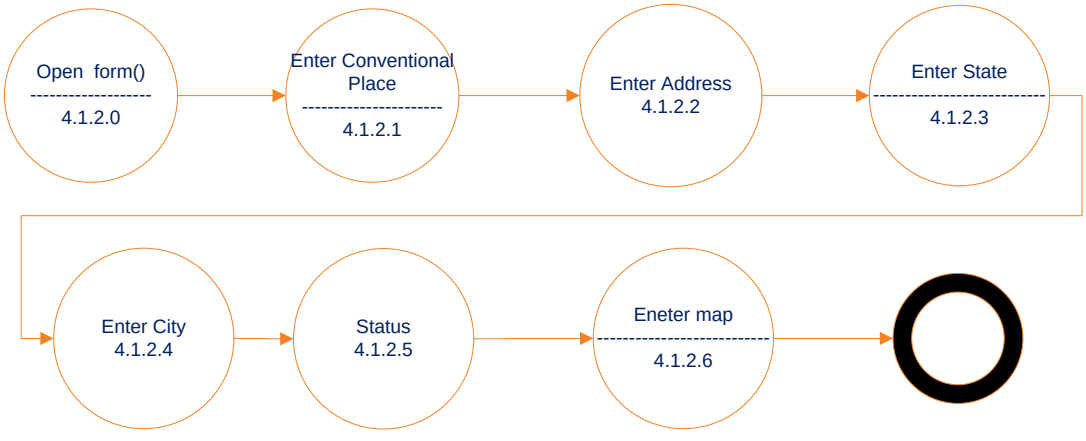

Jobs Master File
DataStore
GIU Interface



LEVEL-4:

Administrator:

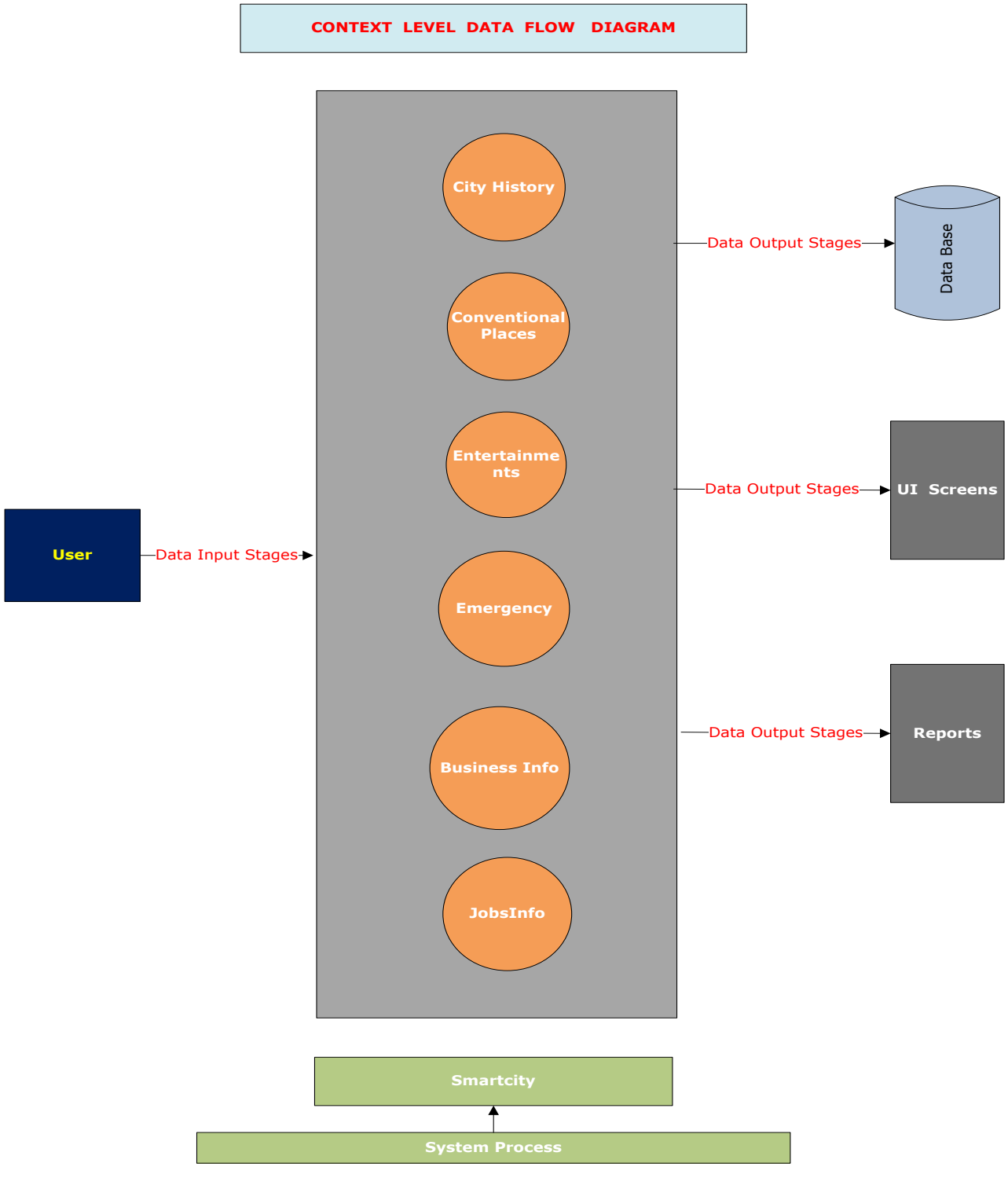
Level 4 Data Flow Diagram for Registration of Administrator



Level 4 Data Flow Diagram for Add Conventional Places

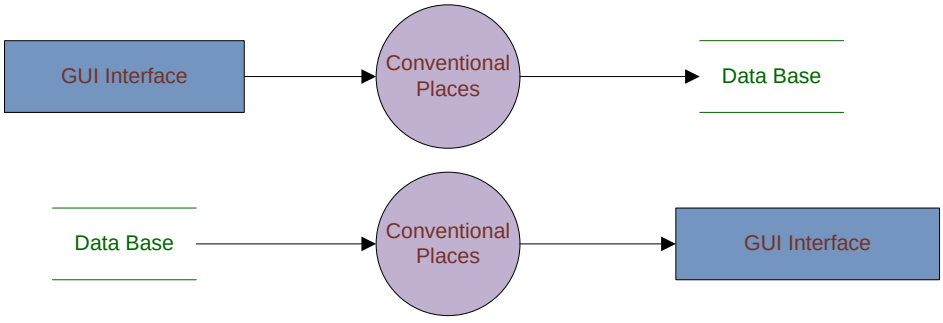
LEVEL-1:

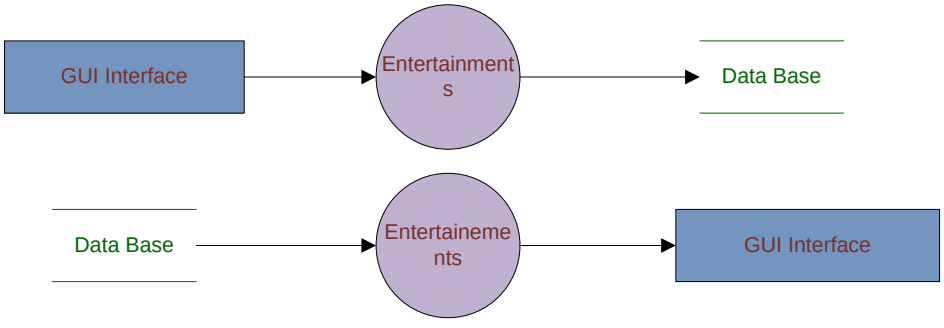
User:

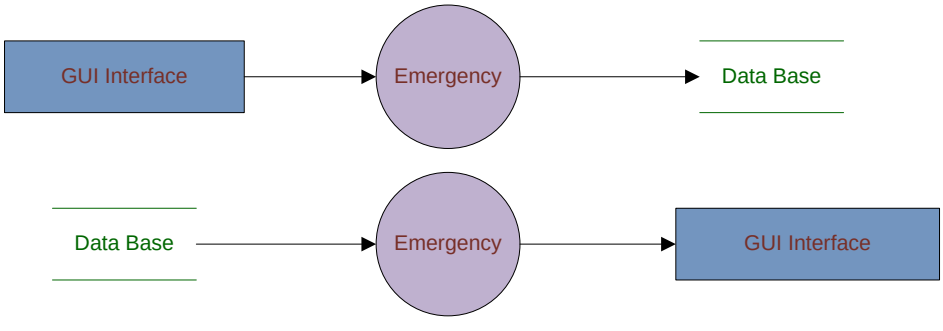

CONTEXT LEVEL DATA FLOW DIAGRAM
City History
Conventional Places
Entertainments
Emergency
Business Info
JobsInfo
Data Output Stages
Data Base
Data Output Stages
UI Screens
Data Output Stages
Reports
User
Data Input Stages
Smartcity
System Process

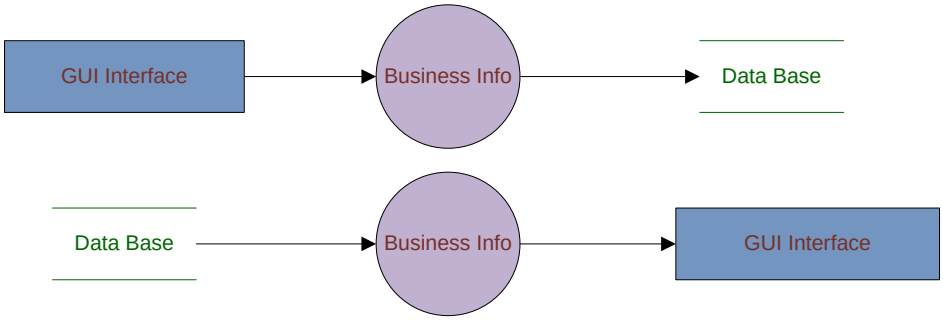
LEVEL-2:

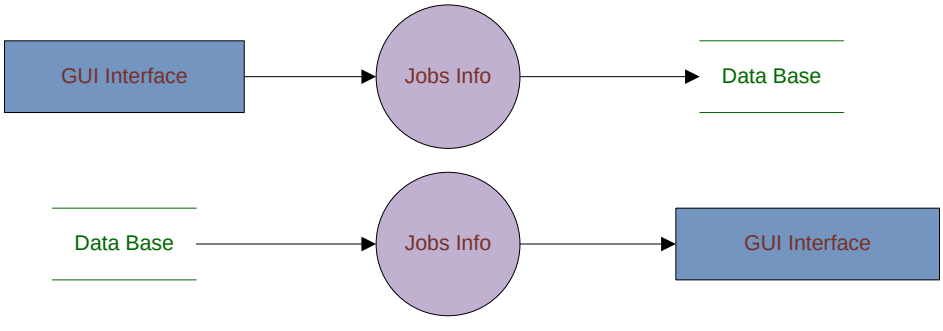
User:

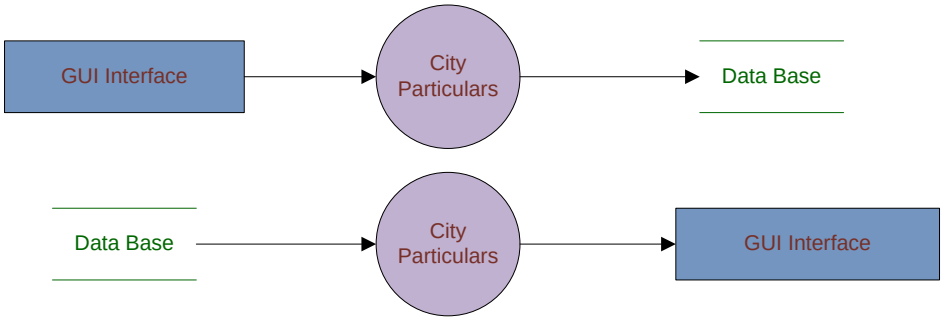

GUI Interface
Conventional Places
Data Base
Data Base
Conventional Places
GUI Interface


GUI Interface
Entertainments
Data Base
Data Base
Entertainements
GUI Interface


GUI Interface
Emergency
Data Base
Data Base
Emergency
GUI Interface

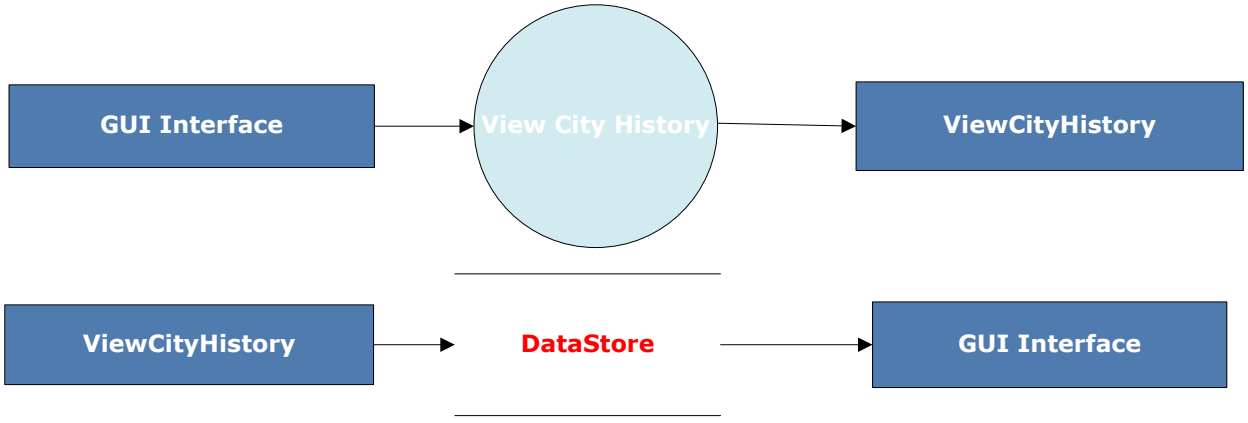

GUI Interface
Business Info
Data Base
Data Base
Business Info
GUI Interface

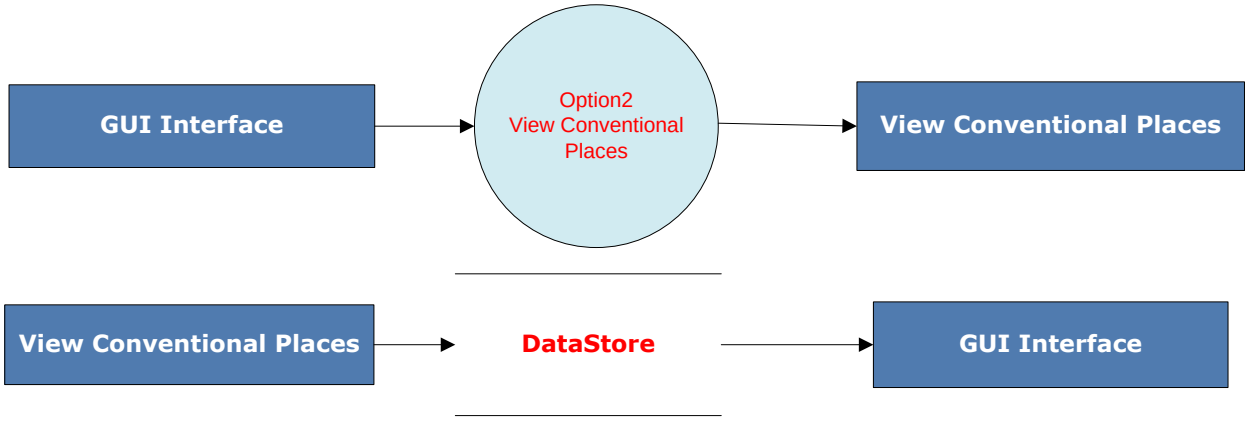

GUI Interface
Jobs Info
Data Base
Data Base
Jobs Info
GUI Interface

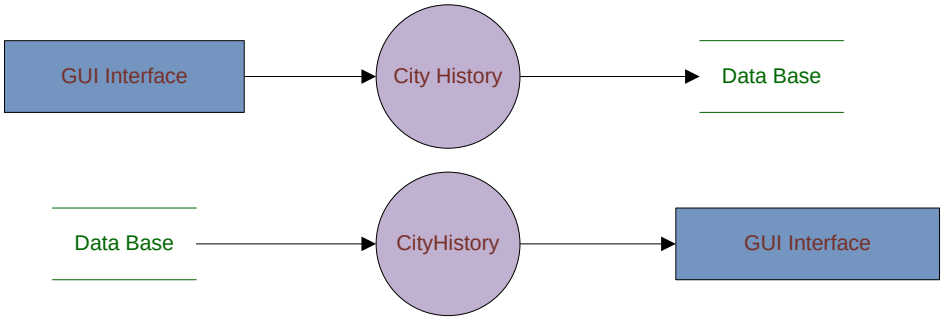

GUI Interface
City Particulars
Data Base
Data Base
City Particulars
GUI Interface

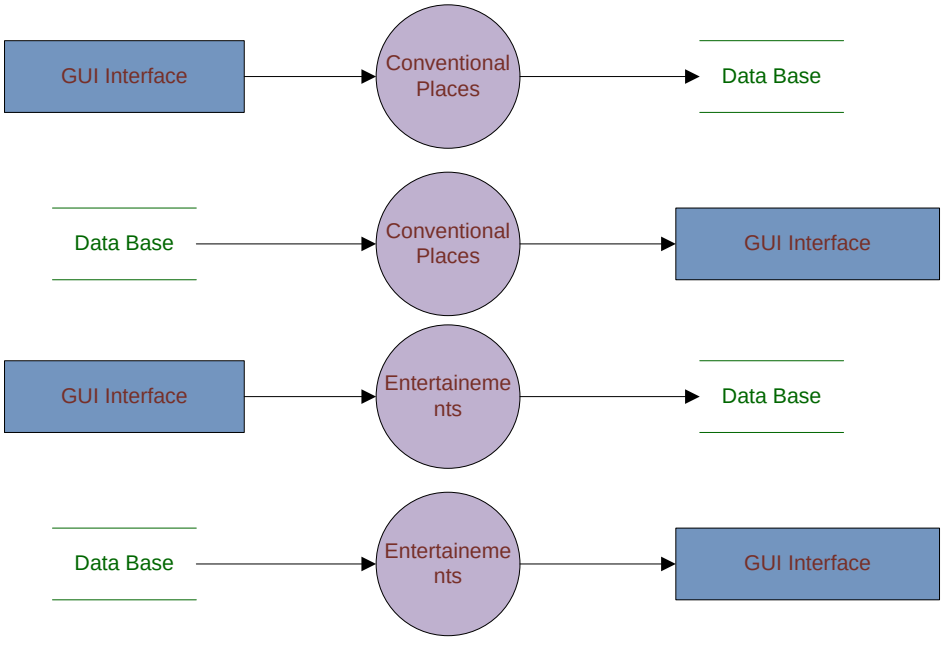
LEVEL-3:

User:


GUI Interface
View City History
ViewCityHistory
ViewCityHistory
DataStore
GUI Interface

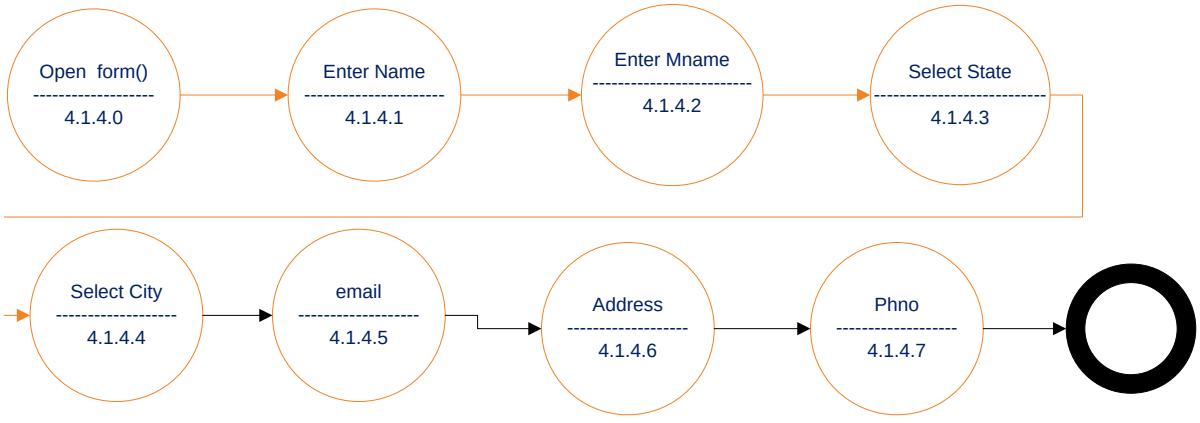

GUI Interface
Option2
View Conventional
Places
View Conventional Places
View Conventional Places
DataStore
GUI Interface


GUI Interface
City History
Data Base
Data Base
CityHistory
GUI Interface


GUI Interface
Conventional Places
Data Base
Data Base
Conventional Places
GUI Interface
GUI Interface
Entertainements
Data Base
Data Base
Entertainements
GUI Interface

LEVEL-4:

User:



Level 4 Data Flow Diagram for Registration

UML Diagrams

Unified Modeling Language:

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows.

- User Model View
    i. This view represents the system from the user's perspective.
    ii. The analysis representation describes a usage scenario from the end-user's perspective.

- Structural model view
    i. In this model the data and functionality are arrived from inside the system.
    ii. This model view models the static structures.

- Behavioral Model View
  It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

- Implementation Model View
  In this the structural and behavioral as parts of the system are represented as they are to be built.

- Environmental Model View
  In these the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are:

- ✓ UML Analysis modeling, this focuses on the user model and structural

model views of the system.

- ✓ UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

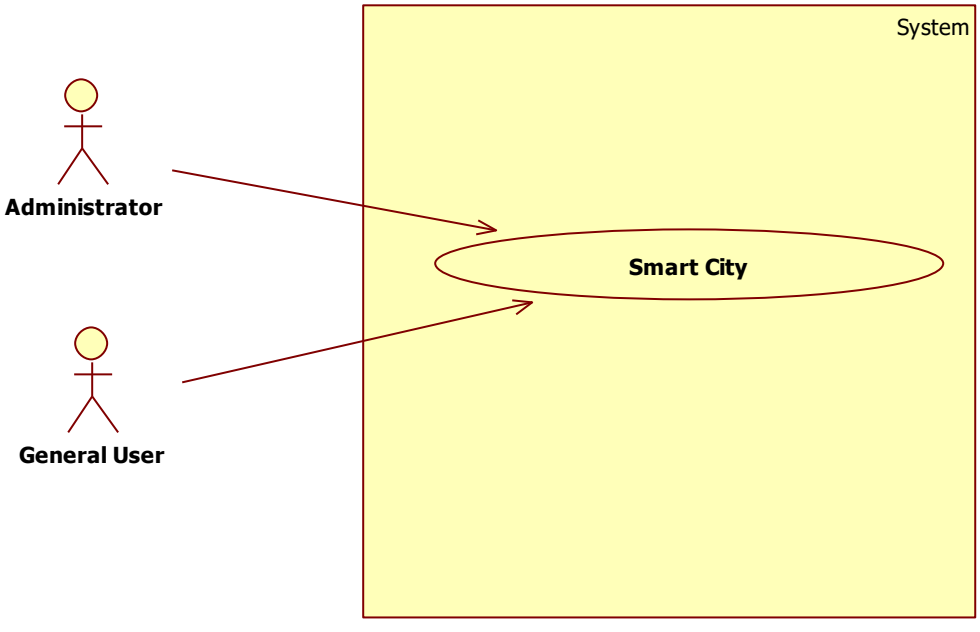
Use case Diagrams represent the functionality of the system from a user's point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

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

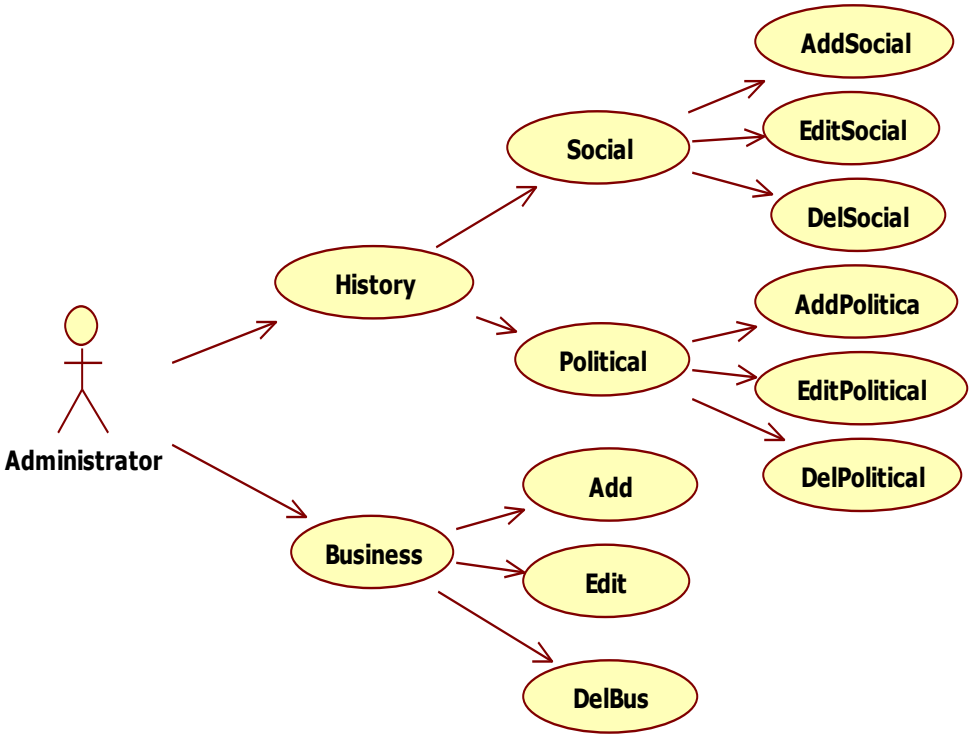
Class Diagram

Use-case Diagram

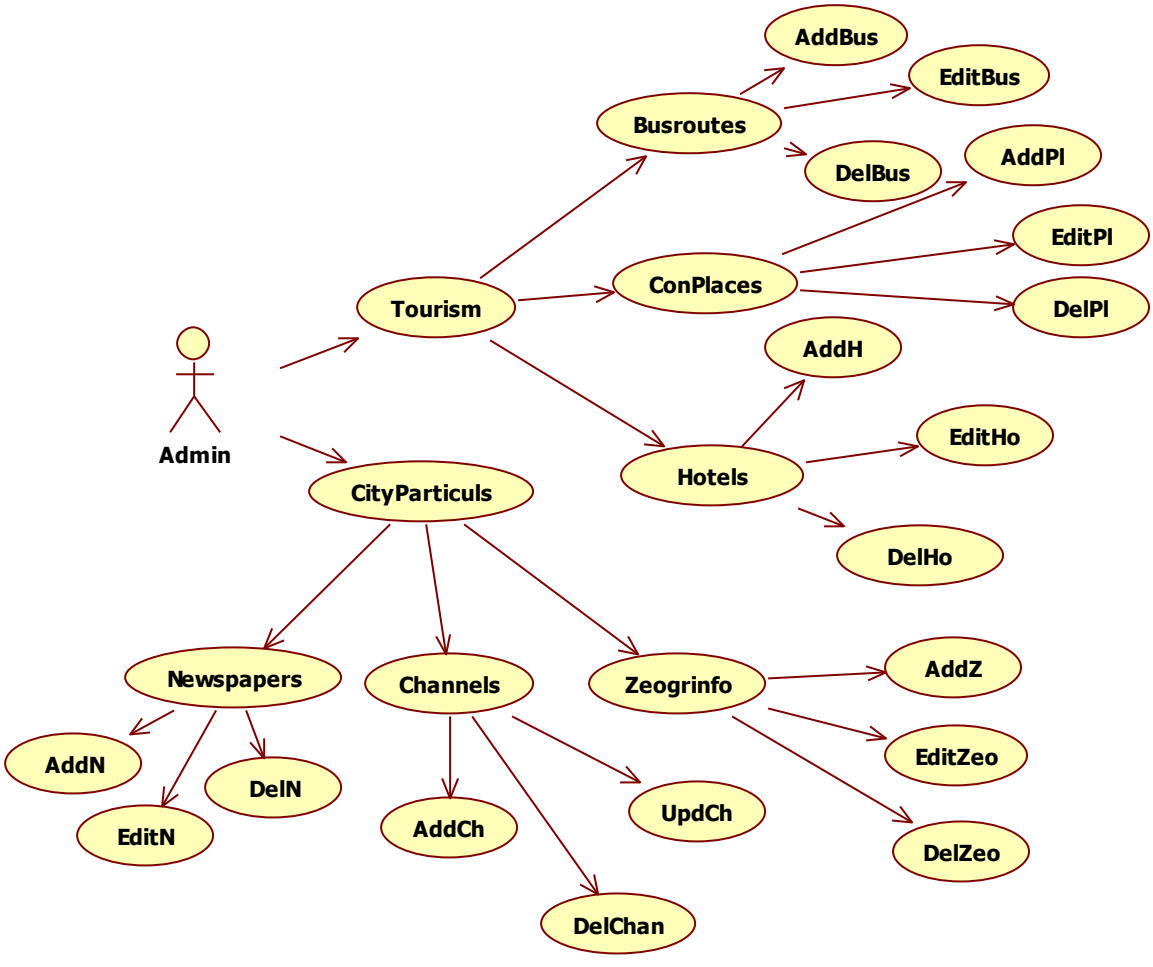
System Use Case Diagram:



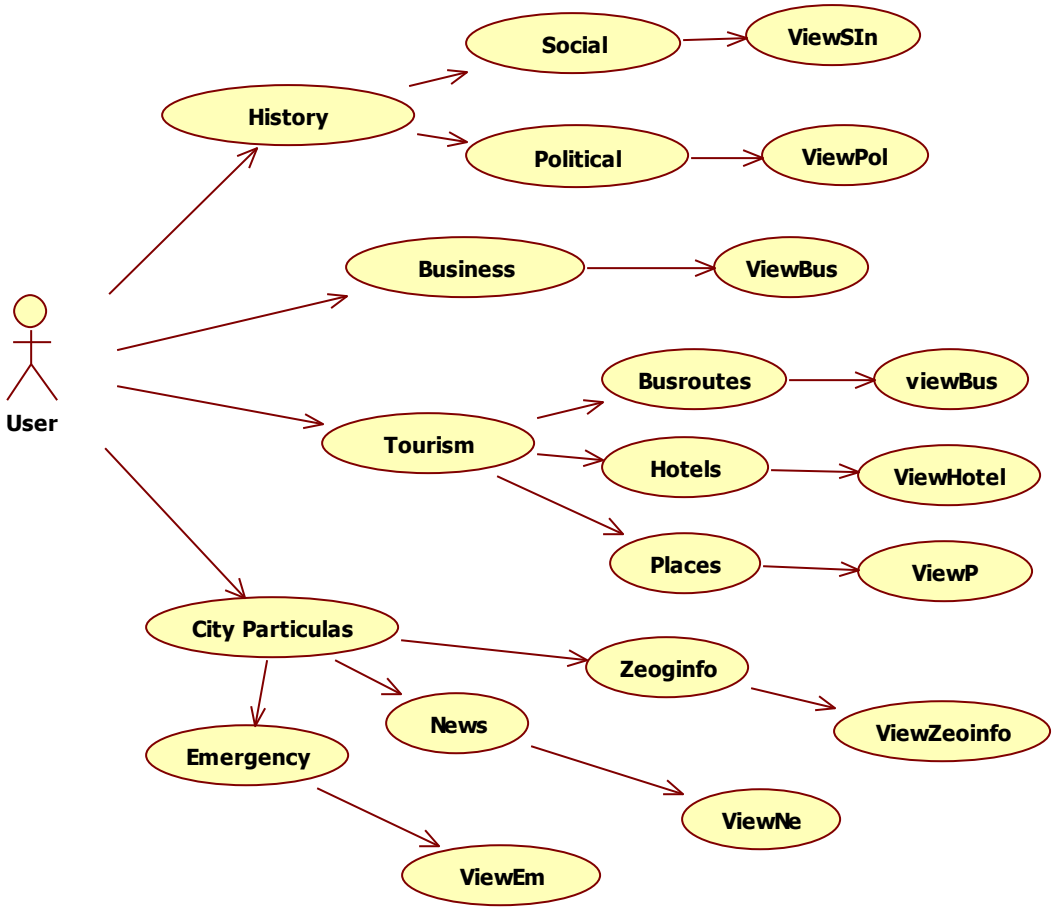
Administrator Use case Diagram1:



Administrator Use Case Diagram2:

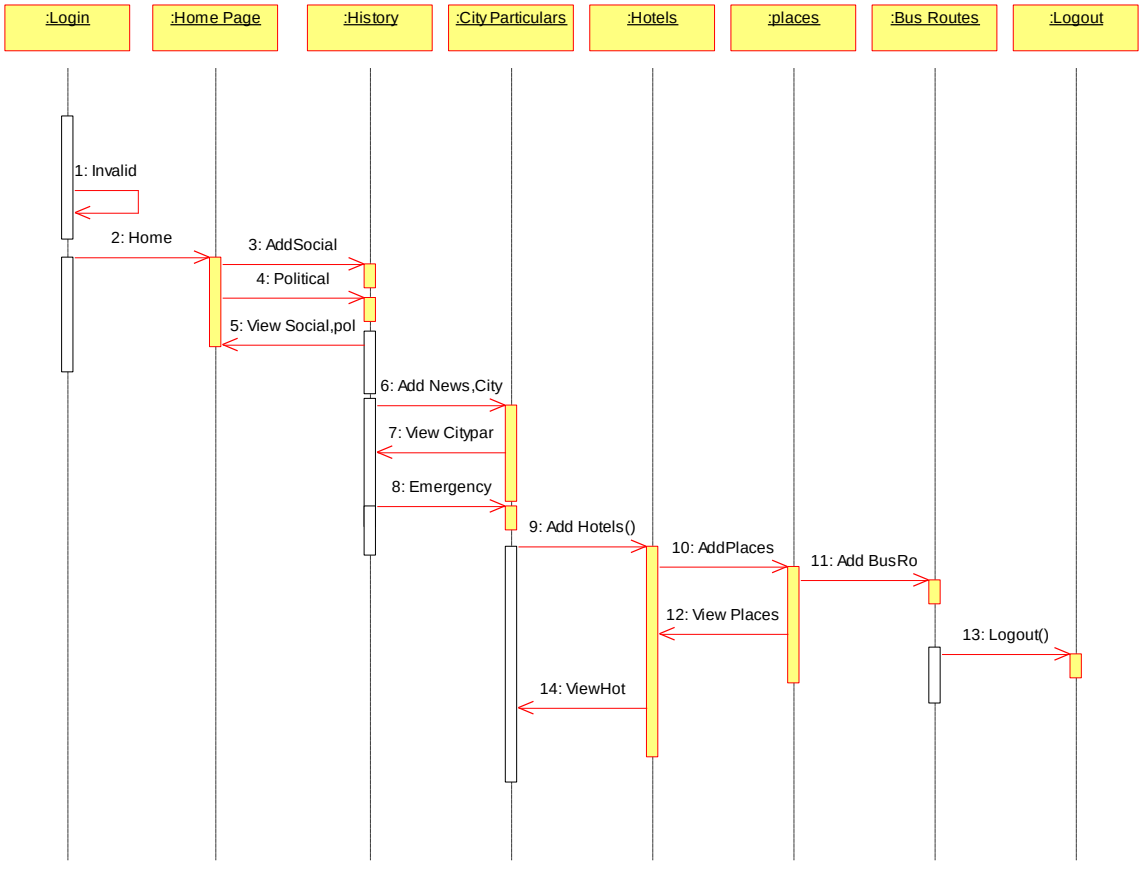

AddBus
EditBus
Busroutes
DelBus
AddPl
EditPl
ConPlaces
DelPl
Tourism
AddH
Admin
EditHo
Hotels
CityParticuls
DelHo
Newspapers
Channels
Zeogrinfo
AddZ
AddN
EditZeo
DelN
EditN
AddCh
UpdCh
DelZeo
DelChan

User Use case Diagram:

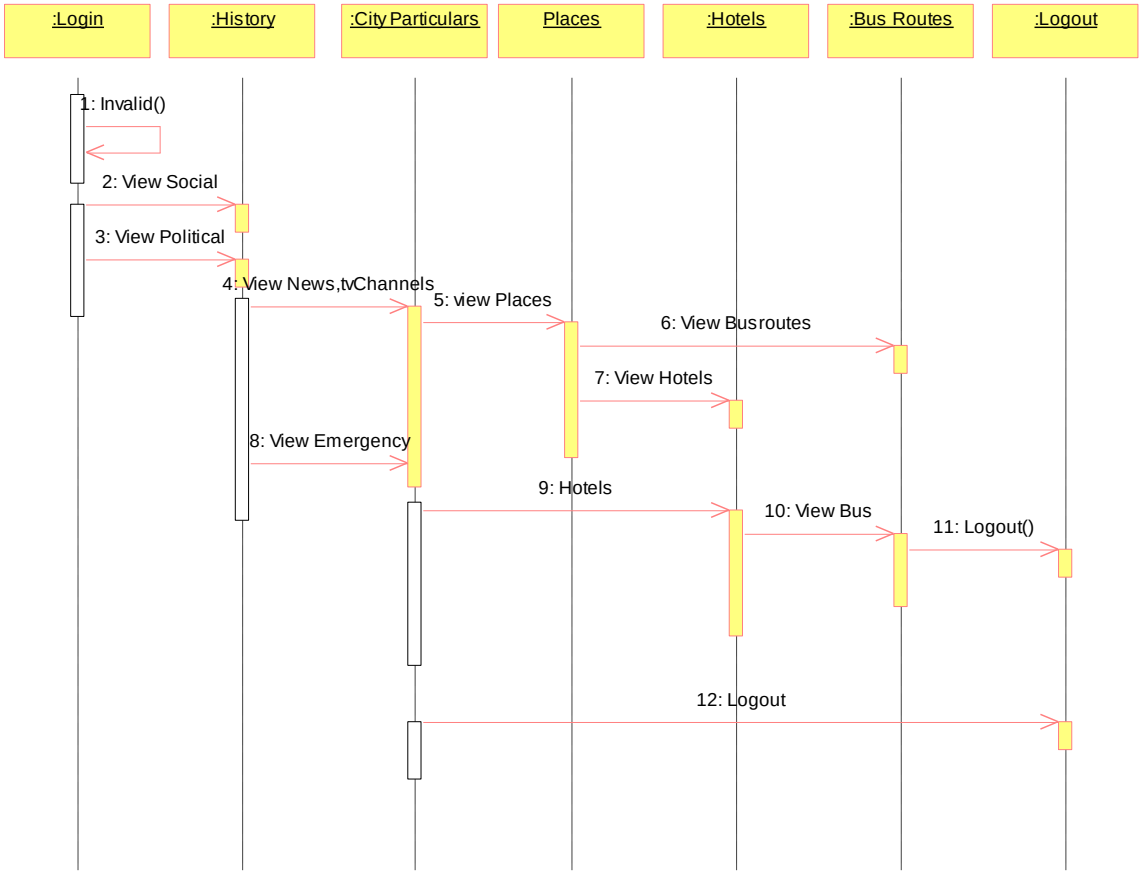


Sequence Diagram

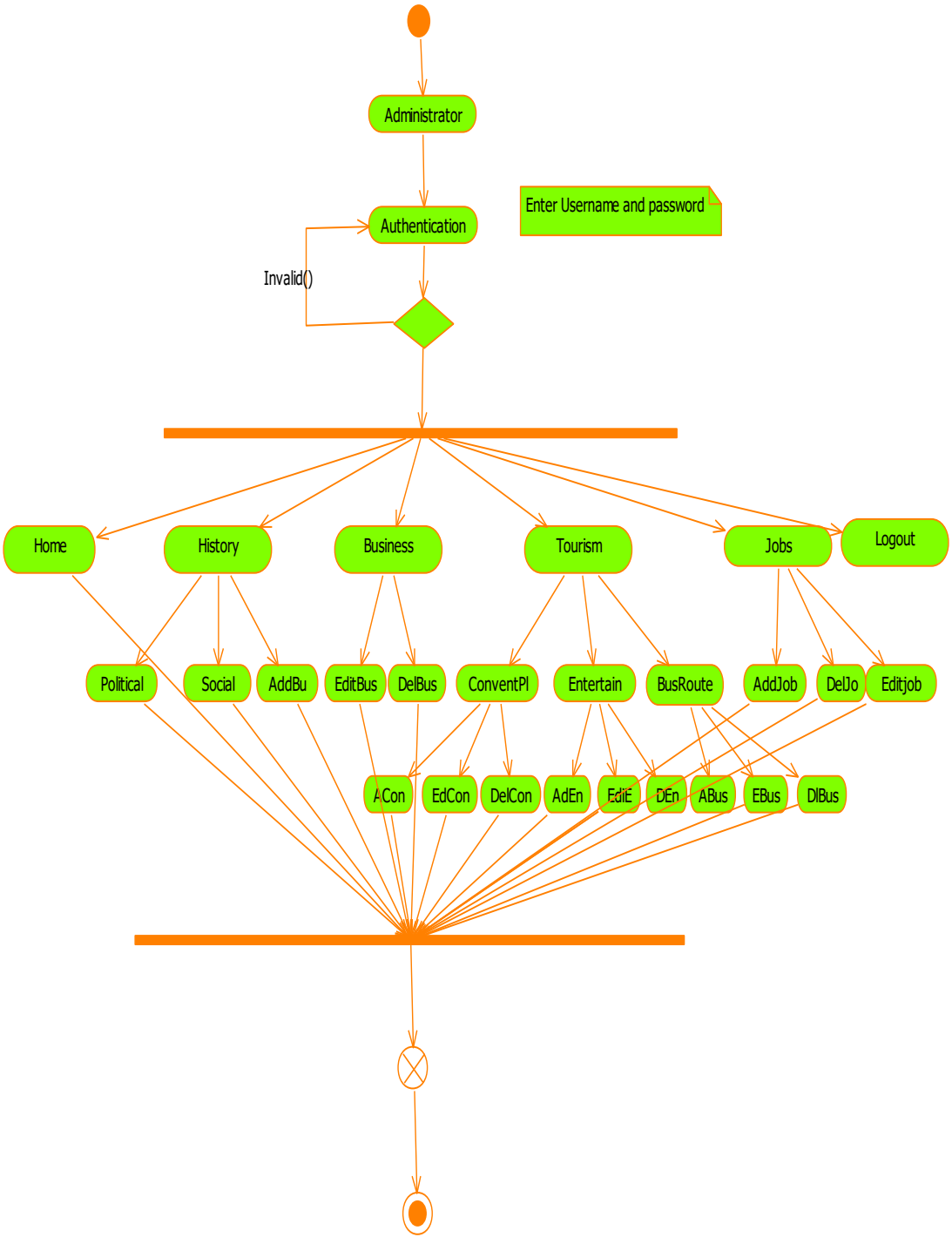
Administrator Sequence Diagram:



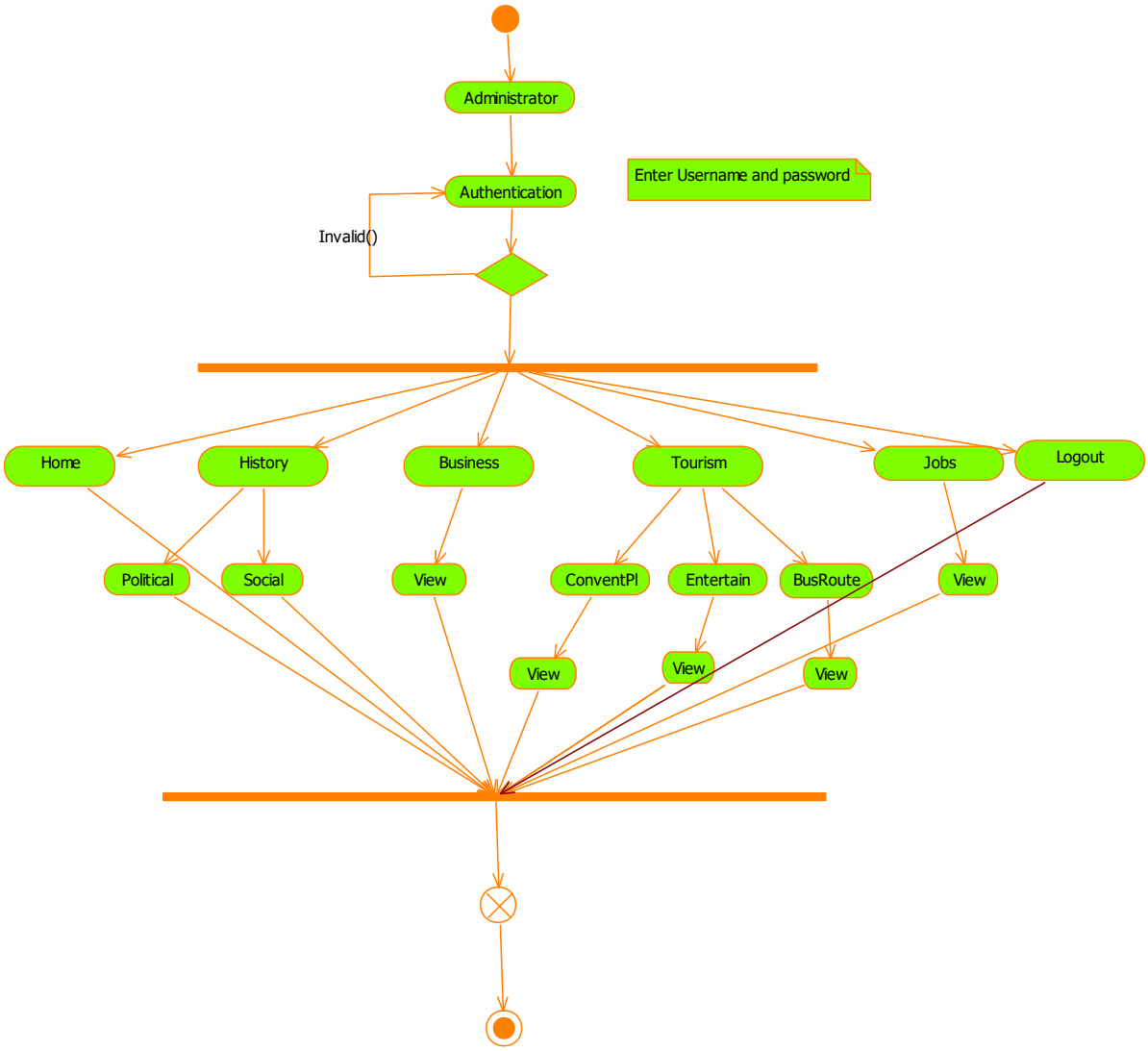
General user sequence Diagram:



Activity Diagrams:


Administrator
Authentication
Enter Username and password
Invalid()
Home
History
Business
Tourism
Jobs
Logout
Political
Social
AddBu
EditBus
DelBus
ConventPl
Entertain
BusRoute
AddJob
DelJo
Editjob
ACon
EdCon
DelCon
AdEn
EditE
DEn
ABus
EBus
DlBus

Employee:


Administrator
Authentication
Enter Username and password
Invalid()
Home
History
Business
Tourism
Jobs
Logout
Political
Social
View
ConventPl
Entertain
BusRoute
View
View
View
View

AVC ALERTS

Column Name	Data Type	Size	Accept Null Value	Unique
CITY_ID	NUMBER	10	YES	NO
ALERTTYPE	VARCHAR2	40	YES	NO
ALERT	VARCHAR2	200	YES	NO

BUSINESS_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
CITY_ID	NUMBER	10	YES	NO
COMP_NAME	VARCHAR2	20	YES	NO
DESCRIPTION	VARCHAR2	2000	YES	NO
ADDRESS	VARCHAR2	100	YES	NO
PHONE	VARCHAR2	10	YES	NO
EMAIL	VARCHAR2	30	YES	NO
CATEGORY	VARCHAR2	20	YES	NO
BUSINESS_ID	NUMBER	10	YES	NO

BUSROUTES

Column Name	Data Type	Size	Accept Null Value	Unique
CITYID	NUMBER	10	YES	NO
BUSNO	VARCHAR2	10	YES	YES
ORIGIN	VARCHAR2	30	YES	NO
VIA	VARCHAR2	200	YES	NO
DESTINATION	VARCHAR2	30	YES	NO

CITIES_JOBS

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
CITY_ID	NUMBER	10	YES	NO
COMP_NAME	VARCHAR2	10	YES	NO
COMP_ADDRESS	VARCHAR2	100	YES	NO
COMP_PHNO	VARCHAR2	10	YES	NO
JOB_NAME	VARCHAR2	20	YES	NO
QUALIFICATION	VARCHAR2	20	YES	NO

CITY_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	12	YES	NO
ZONE_INDIA	VARCHAR2	20	YES	NO
CITY_NAME	VARCHAR2	20	YES	NO
CITY_ID	NUMBER	10	NO	YES
CITYMAP	BLOB	4000	YES	NO
HISTORY	VARCHAR2	2000	YES	NO
MAYOR	VARCHAR2	30	YES	NO
OFFICEADDR	VARCHAR2	150	YES	NO
PHONE	VARCHAR2	15	YES	NO
NEWSPAPER	VARCHAR2	50	YES	NO
TVCHANNELS	VARCHAR2	20	YES	NO

CITY_TOURINFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
LOCATION	VARCHAR2	20	YES	NO
LATITUDE	VARCHAR2	20	YES	NO
LOGITUDE	VARCHAR2	20	YES	NO
AREA	VARCHAR2	20	YES	NO
CLIMATE	VARCHAR2	20	YES	NO
MAX_TEMP	VARCHAR2	20	YES	NO
MIN_TEM	VARCHAR2	20	YES	NO
AVG_RAINFALL	VARCHAR2	20	YES	NO
POPULATION	VARCHAR2	20	YES	NO
LANGUAGES	VARCHAR2	30	YES	NO
BEST_TIMEVISIT	VARCHAR2	20	YES	NO
CLOTHING	VARCHAR2	30	YES	NO
CITY_ID	NUMBER	12	YES	NO

EMERGENCY_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
CITY_ID	NUMBER	10	YES	NO
POLICE_NUMBER	VARCHAR2	20	YES	NO
FIRE_NUBMER	VARCHAR2	20	YES	NO
APUT_NUMBER	VARCHAR2	20	YES	NO
DISTCOLLEC_NUMBER	VARCHAR2	20	YES	NO
HOSPITAL_NAME	VARCHAR2	20	YES	NO
HADDRESS	VARCHAR2	100	YES	NO
HPHONE_NUMBER	VARCHAR2	20	YES	NO
BBANK_NAME	VARCHAR2	20	YES	NO
BBADDRESS	VARCHAR2	20	YES	NO
BBPHONE_NUMBER	VARCHAR2	20	YES	NO
ENQ_NAME	VARCHAR2	20	YES	NO
ENQ_NUMBER	VARCHAR2	20	YES	NO

HISTORY_CITY

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	12	YES	NO
CITY_ID	NUMBER	10	YES	NO
CITY_IMAGE	BLOB	4000	YES	NO
CITY_HISTORY	VARCHAR2	2000	YES	NO

HOTEL_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
CITY_ID	NUMBER	10	YES	NO
HOTLE_NAME	VARCHAR2	20	YES	NO
RATING	VARCHAR2	12	YES	NO
ADDRESS	VARCHAR2	100	YES	NO
PHONE_NUMBER	VARCHAR2	10	YES	NO

LOGINDETAILS

Column Name	Data Type	Size	Accept Null Value	Unique
LOGINNAME	VARCHAR2	30	NO	YES
PASSWORD	VARCHAR2	30	NO	NO
LOGINTYPE	VARCHAR2	20	YES	NO
FIRST_NAME	VARCHAR2	40	YES	NO
LAST_NAME	VARCHAR2	40	YES	NO
LOGIN_STATUS	NUMBER	30	YES	NO
REG_DATE	DATE	7	YES	NO
SQUESTIONID	NUMBER	5	YES	NO
SANSWER	VARCHAR2	40	YES	NO
PASSMODIFIEDDATE	DATE	7	YES	NO
FIRSTLOGIN	NUMBER	5	YES	NO

LOGIN_AUDIT

Column Name	Data Type	Size	Accept Null Value	Unique
LOGINID	VARCHAR2	40	YES	NO
LOGINDATE	DATE	7	YES	NO
LOGIN_DESC	VARCHAR2	40	YES	NO

LOGIN_PROFILE

Column Name	Data Type	Size	Accept Null Value	Unique
LOGINID	VARCHAR2	40	YES	NO
BIRTHDATE	DATE	7	YES	NO
CITY	VARCHAR2	40	YES	NO
STATE	VARCHAR2	40	YES	NO
COUNTRY	VARCHAR2	40	YES	NO
EMAIL	VARCHAR2	40	YES	NO

MAILS

Column Name	Data Type	Size	Accept Null Value	Unique
MESSAGEID	NUMBER	4	NO	YES
FROMNAME	VARCHAR2	20	YES	NO
TONAME	VARCHAR2	20	YES	NO
SENDDATE	DATE	7	YES	NO
SUBJECT	VARCHAR2	30	YES	NO
MESSAGE	VARCHAR2	200	YES	NO
SENDERSTATUS	NUMBER	2	YES	NO
RECEIVERSTATUS	NUMBER	2	YES	NO

MINISTERS

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	5	YES	NO
MINISTRY	VARCHAR2	30	YES	NO
MINISTER	VARCHAR2	30	YES	NO

PLACES

Column Name	Data Type	Size	Accept Null Value	Unique
CITYID	NUMBER	10	YES	NO
PLACEID	NUMBER	10	YES	YES
PLACE	VARCHAR2	50	YES	NO
LOCATION	VARCHAR2	150	YES	NO
HISTORY	VARCHAR2	2000	YES	NO
IMG	BLOB	4000	YES	NO
PHONE	VARCHAR2	15	YES	NO

QUESTION_BASE

Column Name	Data Type	Size	Accept Null Value	Unique
QUESTION_ID	NUMBER	22	NO	YES
QUESTION_DETAIL	VARCHAR2	70	YES	NO

SHOPS

Column Name	Data Type	Size	Accept Null Value	Unique
SHOPID	NUMBER	5	YES	YES
CITYID	NUMBER	10	YES	NO
SHOPNAME	VARCHAR2	30	YES	NO
ADDR	VARCHAR2	200	YES	NO
FAMOUSFOR	VARCHAR2	30	YES	NO
OFFERS	VARCHAR2	300	YES	NO
PHONE	VARCHAR2	20	YES	NO

STATE_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	12	NO	YES
STATE_NAME	VARCHAR2	20	YES	NO
STATE_CAPITAL	VARCHAR2	20	YES	NO
CM	VARCHAR2	30	YES	NO
GOVERNOR	VARCHAR2	40	YES	NO
SITEMAP	BLOB	4000	YES	NO
LANGUAGES	VARCHAR2	50	YES	NO

SUBSCRIPTION

Column Name	Data Type	Size	Accept Null Value	Unique
SID	NUMBER	6	NO	YES
USERNAME	VARCHAR2	40	YES	NO
EMAIL	VARCHAR2	40	YES	NO
CITY_ID	NUMBER	10	YES	NO
ALERTTYPE	VARCHAR2	40	YES	NO

THEATRES

Column Name	Data Type	Size	Accept Null Value	Unique
THEATREID	NUMBER	5	YES	YES
CITYID	NUMBER	10	YES	NO
THEATRENAME	VARCHAR2	30	YES	NO
CINEMA	VARCHAR2	30	YES	NO
CINEMAIMG	BLOB	4000	YES	NO
MORSHOWTIME	VARCHAR2	30	YES	NO
MATSHOWTIME	VARCHAR2	30	YES	NO
FIRSHOWTIME	VARCHAR2	30	YES	NO
SECSHOWTIME	VARCHAR2	30	YES	NO
ADDRESS	VARCHAR2	200	YES	NO
PHONE	VARCHAR2	15	YES	NO

ALERTS

Column Name	Data Type	Size	Accept Null Value	Unique
CITY_ID	NUMBER	10	YES	NO
ALERTTYPE	VARCHAR2	40	YES	NO
ALERT	VARCHAR2	200	YES	NO

BUSINESS_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
CITY_ID	NUMBER	10	YES	NO
COMP_NAME	VARCHAR2	20	YES	NO
DESCRIPTION	VARCHAR2	2000	YES	NO
ADDRESS	VARCHAR2	100	YES	NO
PHONE	VARCHAR2	10	YES	NO
EMAIL	VARCHAR2	30	YES	NO
CATEGORY	VARCHAR2	20	YES	NO
BUSINESS_ID	NUMBER	10	YES	NO

BUSROUTES

Column Name	Data Type	Size	Accept Null Value	Unique
CITYID	NUMBER	10	YES	NO
BUSNO	VARCHAR2	10	YES	YES
ORIGIN	VARCHAR2	30	YES	NO
VIA	VARCHAR2	200	YES	NO
DESTINATION	VARCHAR2	30	YES	NO

CITIES_JOBS

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
CITY_ID	NUMBER	10	YES	NO
COMP_NAME	VARCHAR2	10	YES	NO
COMP_ADDRESS	VARCHAR2	100	YES	NO
COMP_PHNO	VARCHAR2	10	YES	NO
JOB_NAME	VARCHAR2	20	YES	NO
QUALIFICATION	VARCHAR2	20	YES	NO

CITY_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	12	YES	NO
ZONE_INDIA	VARCHAR2	20	YES	NO
CITY_NAME	VARCHAR2	20	YES	NO
CITY_ID	NUMBER	10	NO	YES
CITYMAP	BLOB	4000	YES	NO
HISTORY	VARCHAR2	2000	YES	NO
MAYOR	VARCHAR2	30	YES	NO
OFFICEADDR	VARCHAR2	150	YES	NO
PHONE	VARCHAR2	15	YES	NO
NEWSPAPER	VARCHAR2	50	YES	NO
TVCHANNELS	VARCHAR2	20	YES	NO

CITY_TOURINFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
LOCATION	VARCHAR2	20	YES	NO
LATITUDE	VARCHAR2	20	YES	NO
LOGITUDE	VARCHAR2	20	YES	NO
AREA	VARCHAR2	20	YES	NO
CLIMATE	VARCHAR2	20	YES	NO
MAX_TEMP	VARCHAR2	20	YES	NO
MIN_TEM	VARCHAR2	20	YES	NO
AVG_RAINFALL	VARCHAR2	20	YES	NO
POPULATION	VARCHAR2	20	YES	NO
LANGUAGES	VARCHAR2	30	YES	NO
BEST_TIMEVISIT	VARCHAR2	20	YES	NO
CLOTHING	VARCHAR2	30	YES	NO
CITY_ID	NUMBER	12	YES	NO

EMERGENCY_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
CITY_ID	NUMBER	10	YES	NO
POLICE_NUMBER	VARCHAR2	20	YES	NO
FIRE_NUBMER	VARCHAR2	20	YES	NO
APUT_NUMBER	VARCHAR2	20	YES	NO
DISTCOLLEC_NUMBER	VARCHAR2	20	YES	NO
HOSPITAL_NAME	VARCHAR2	20	YES	NO
HADDRESS	VARCHAR2	100	YES	NO
HPHONE_NUMBER	VARCHAR2	20	YES	NO
BBANK_NAME	VARCHAR2	20	YES	NO
BBADDRESS	VARCHAR2	20	YES	NO
BBPHONE_NUMBER	VARCHAR2	20	YES	NO
ENQ_NAME	VARCHAR2	20	YES	NO
ENQ_NUMBER	VARCHAR2	20	YES	NO

HISTORY_CITY

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	12	YES	NO
CITY_ID	NUMBER	10	YES	NO
CITY_IMAGE	BLOB	4000	YES	NO
CITY_HISTORY	VARCHAR2	2000	YES	NO

HOTEL_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
CITY_ID	NUMBER	10	YES	NO
HOTLE_NAME	VARCHAR2	20	YES	NO
RATING	VARCHAR2	12	YES	NO
ADDRESS	VARCHAR2	100	YES	NO
PHONE_NUMBER	VARCHAR2	10	YES	NO

LOGINDETAILS

Column Name	Data Type	Size	Accept Null Value	Unique
LOGINNAME	VARCHAR2	30	NO	YES
PASSWORD	VARCHAR2	30	NO	NO
LOGINTYPE	VARCHAR2	20	YES	NO
FIRST_NAME	VARCHAR2	40	YES	NO
LAST_NAME	VARCHAR2	40	YES	NO
LOGIN_STATUS	NUMBER	30	YES	NO
REG_DATE	DATE	7	YES	NO
SQUESTIONID	NUMBER	5	YES	NO
SANSWER	VARCHAR2	40	YES	NO
PASSMODIFIEDDATE	DATE	7	YES	NO
FIRSTLOGIN	NUMBER	5	YES	NO

LOGIN_AUDIT

Column Name	Data Type	Size	Accept Null Value	Unique
LOGINID	VARCHAR2	40	YES	NO
LOGINDATE	DATE	7	YES	NO
LOGIN_DESC	VARCHAR2	40	YES	NO

LOGIN_PROFILE

Column Name	Data Type	Size	Accept Null Value	Unique
LOGINID	VARCHAR2	40	YES	NO
BIRTHDATE	DATE	7	YES	NO
CITY	VARCHAR2	40	YES	NO
STATE	VARCHAR2	40	YES	NO
COUNTRY	VARCHAR2	40	YES	NO
EMAIL	VARCHAR2	40	YES	NO

MAILS

Column Name	Data Type	Size	Accept Null Value	Unique
MESSAGEID	NUMBER	4	NO	YES
FROMNAME	VARCHAR2	20	YES	NO
TONAME	VARCHAR2	20	YES	NO
SENDDATE	DATE	7	YES	NO
SUBJECT	VARCHAR2	30	YES	NO
MESSAGE	VARCHAR2	200	YES	NO
SENDERSTATUS	NUMBER	2	YES	NO
RECEIVERSTATUS	NUMBER	2	YES	NO

MINISTERS

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	5	YES	NO
MINISTRY	VARCHAR2	30	YES	NO
MINISTER	VARCHAR2	30	YES	NO

PLACES

Column Name	Data Type	Size	Accept Null Value	Unique
CITYID	NUMBER	10	YES	NO
PLACEID	NUMBER	10	YES	YES
PLACE	VARCHAR2	50	YES	NO
LOCATION	VARCHAR2	150	YES	NO
HISTORY	VARCHAR2	2000	YES	NO
IMG	BLOB	4000	YES	NO
PHONE	VARCHAR2	15	YES	NO

QUESTION_BASE

Column Name	Data Type	Size	Accept Null Value	Unique
QUESTION_ID	NUMBER	22	NO	YES
QUESTION_DETAIL	VARCHAR2	70	YES	NO

SHOPS

Column Name	Data Type	Size	Accept Null Value	Unique
SHOPID	NUMBER	5	YES	YES
CITYID	NUMBER	10	YES	NO
SHOPNAME	VARCHAR2	30	YES	NO
ADDR	VARCHAR2	200	YES	NO
FAMOUSFOR	VARCHAR2	30	YES	NO
OFFERS	VARCHAR2	300	YES	NO
PHONE	VARCHAR2	20	YES	NO

STATE_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	12	NO	YES
STATE_NAME	VARCHAR2	20	YES	NO
STATE_CAPITAL	VARCHAR2	20	YES	NO
CM	VARCHAR2	30	YES	NO
GOVERNOR	VARCHAR2	40	YES	NO
SITEMAP	BLOB	4000	YES	NO
LANGUAGES	VARCHAR2	50	YES	NO

SUBSCRIPTION

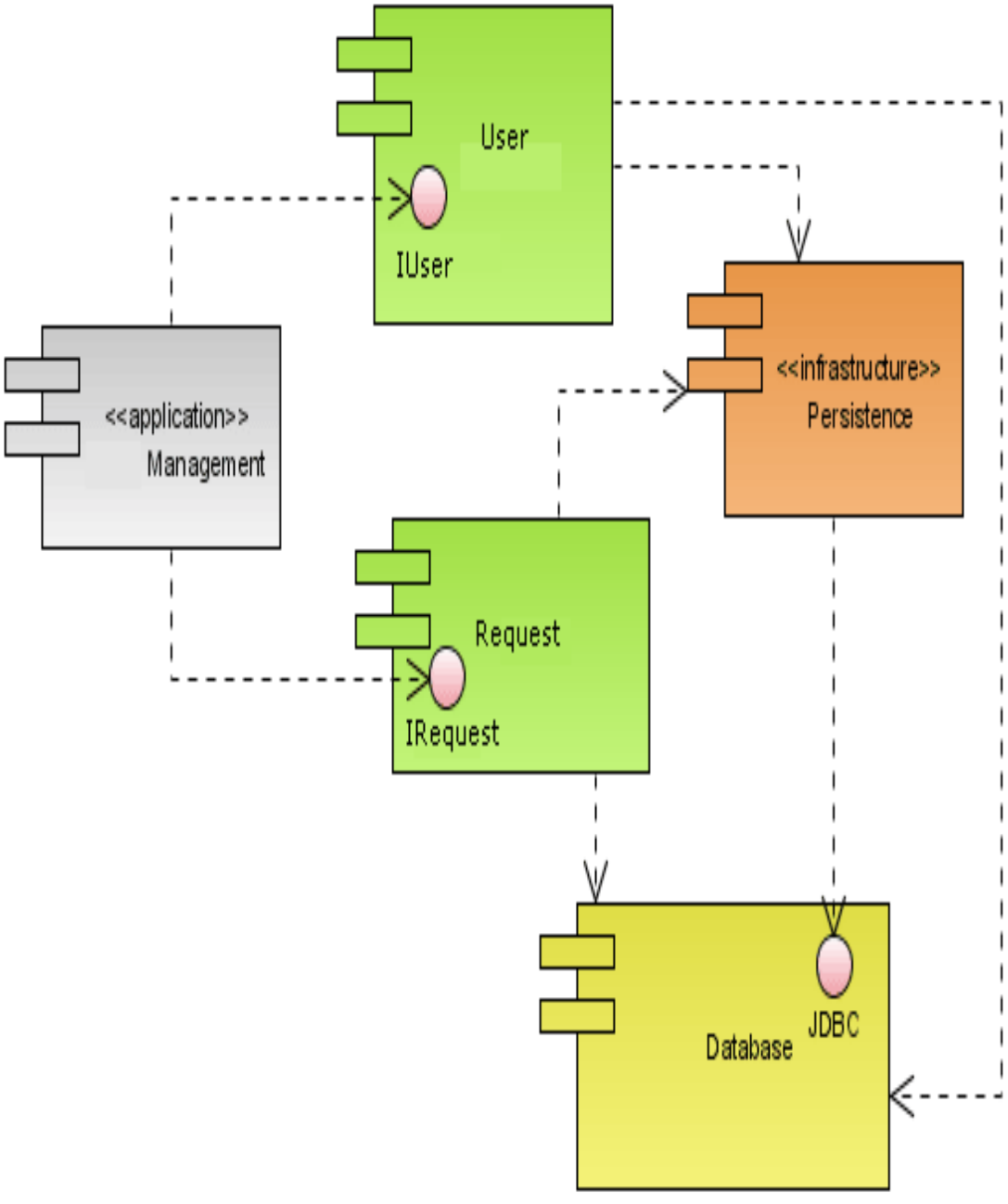
Column Name	Data Type	Size	Accept Null Value	Unique
SID	NUMBER	6	NO	YES
USERNAME	VARCHAR2	40	YES	NO
EMAIL	VARCHAR2	40	YES	NO
CITY_ID	NUMBER	10	YES	NO
ALERTTYPE	VARCHAR2	40	YES	NO

THEATRES

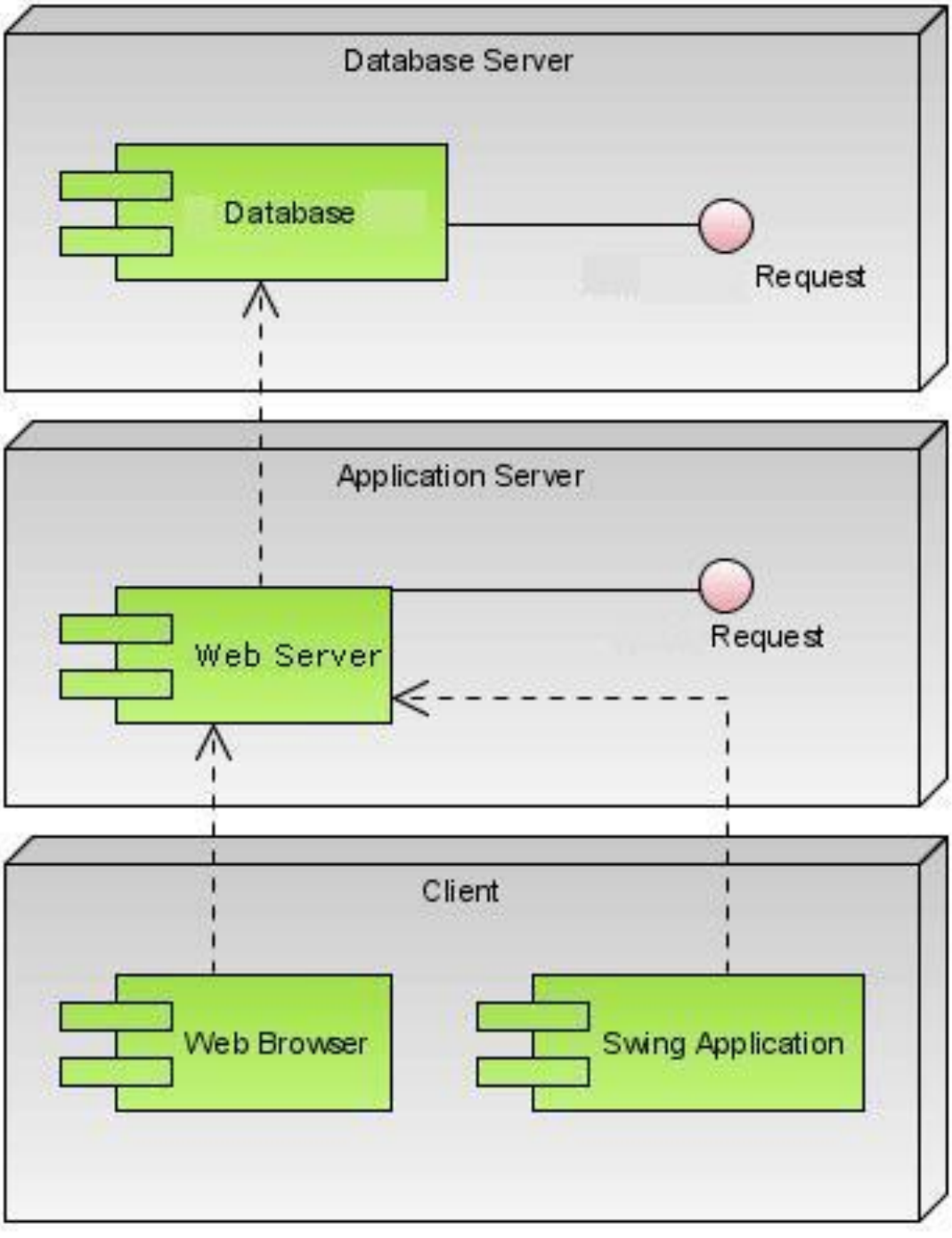
Column Name	Data Type	Size	Accept Null Value	Unique
THEATREID	NUMBER	5	YES	YES
CITYID	NUMBER	10	YES	NO
THEATRENAME	VARCHAR2	30	YES	NO
CINEMA	VARCHAR2	30	YES	NO
CINEMAIMG	BLOB	4000	YES	NO
MORSHOWTIME	VARCHAR2	30	YES	NO
MATSHOWTIME	VARCHAR2	30	YES	NO
FIRSHOWTIME	VARCHAR2	30	YES	NO
SECSHOWTIME	VARCHAR2	30	YES	NO
ADDRESS	VARCHAR2	200	YES	NO
PHONE	VARCHAR2	15	YES	NO

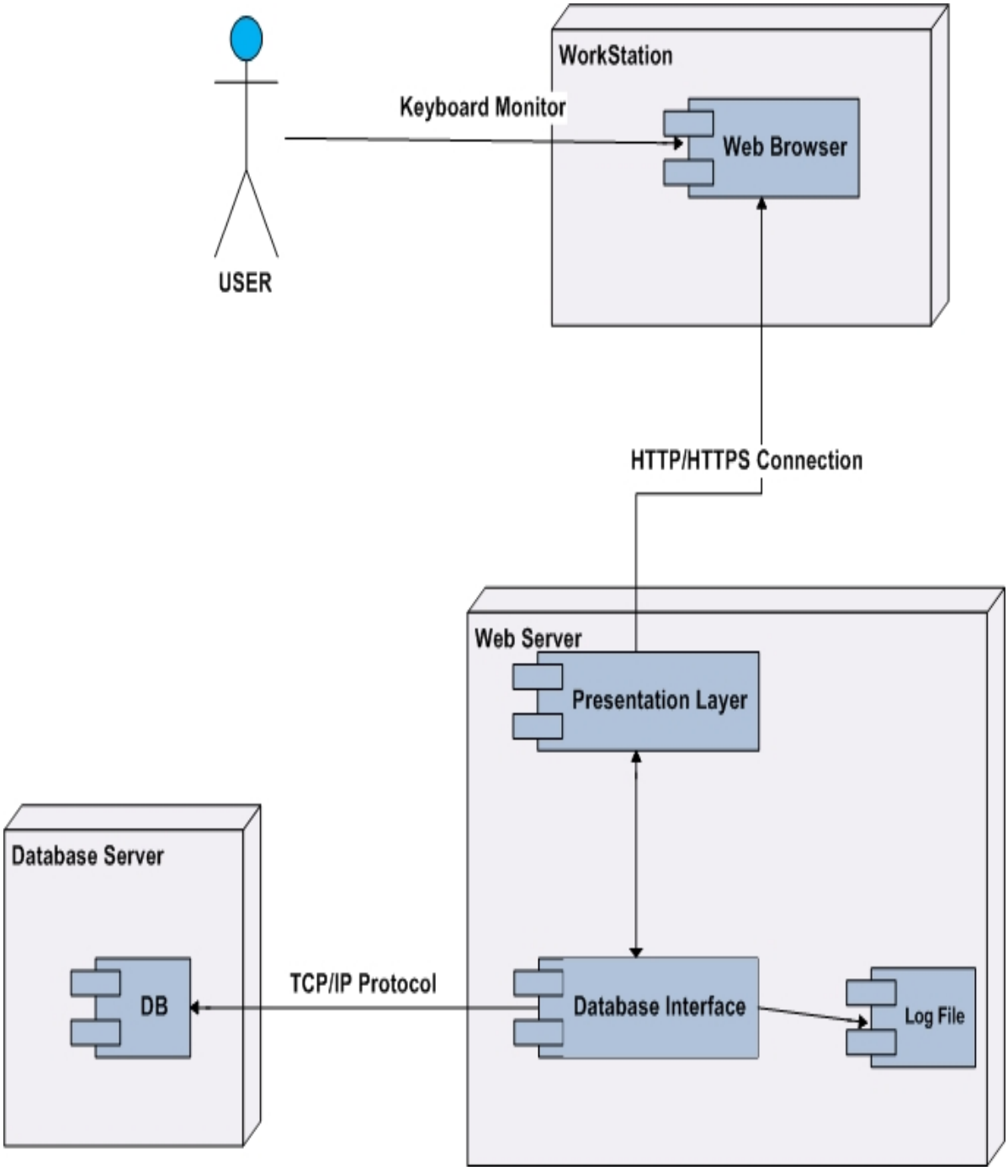
Activity Diagram:

Component Diagram



Deployment Diagram




WorkStation
Keyboard Monitor
Web Browser
USER
HTTP/HTTPS Connection
Web Server
Presentation Layer
Database Server
DB
TCP/IP Protocol
Database Interface
Log File

Data Dictionary

ALERTS

Column Name	Data Type	Size	Accept Null Value	Unique
CITY_ID	NUMBER	10	YES	NO
ALERTTYPE	VARCHAR2	40	YES	NO
ALERT	VARCHAR2	200	YES	NO

BUSINESS_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
CITY_ID	NUMBER	10	YES	NO
COMP_NAME	VARCHAR2	20	YES	NO
DESCRIPTION	VARCHAR2	2000	YES	NO
ADDRESS	VARCHAR2	100	YES	NO
PHONE	VARCHAR2	10	YES	NO
EMAIL	VARCHAR2	30	YES	NO
CATEGORY	VARCHAR2	20	YES	NO
BUSINESS_ID	NUMBER	10	YES	NO

BUSROUTES

Column Name	Data Type	Size	Accept Null Value	Unique
CITYID	NUMBER	10	YES	NO
BUSNO	VARCHAR2	10	YES	YES
ORIGIN	VARCHAR2	30	YES	NO
VIA	VARCHAR2	200	YES	NO
DESTINATION	VARCHAR2	30	YES	NO

CITIES_JOBS

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
CITY_ID	NUMBER	10	YES	NO
COMP_NAME	VARCHAR2	10	YES	NO
COMP_ADDRESS	VARCHAR2	100	YES	NO
COMP_PHNO	VARCHAR2	10	YES	NO
JOB_NAME	VARCHAR2	20	YES	NO
QUALIFICATION	VARCHAR2	20	YES	NO

CITY_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	12	YES	NO
ZONE_INDIA	VARCHAR2	20	YES	NO
CITY_NAME	VARCHAR2	20	YES	NO
CITY_ID	NUMBER	10	NO	YES
CITYMAP	BLOB	4000	YES	NO
HISTORY	VARCHAR2	2000	YES	NO
MAYOR	VARCHAR2	30	YES	NO
OFFICEADDR	VARCHAR2	150	YES	NO
PHONE	VARCHAR2	15	YES	NO
NEWSPAPER	VARCHAR2	50	YES	NO
TVCHANNELS	VARCHAR2	20	YES	NO

CITY_TOURINFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
LOCATION	VARCHAR2	20	YES	NO
LATITUDE	VARCHAR2	20	YES	NO
LOGITUDE	VARCHAR2	20	YES	NO
AREA	VARCHAR2	20	YES	NO
CLIMATE	VARCHAR2	20	YES	NO
MAX_TEMP	VARCHAR2	20	YES	NO
MIN_TEM	VARCHAR2	20	YES	NO
AVG_RAINFALL	VARCHAR2	20	YES	NO
POPULATION	VARCHAR2	20	YES	NO
LANGUAGES	VARCHAR2	30	YES	NO
BEST_TIMEVISIT	VARCHAR2	20	YES	NO
CLOTHING	VARCHAR2	30	YES	NO
CITY_ID	NUMBER	12	YES	NO

EMERGENCY_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
CITY_ID	NUMBER	10	YES	NO
POLICE_NUMBER	VARCHAR2	20	YES	NO
FIRE_NUBMER	VARCHAR2	20	YES	NO
APUT_NUMBER	VARCHAR2	20	YES	NO
DISTCOLLEC_NUMBER	VARCHAR2	20	YES	NO
HOSPITAL_NAME	VARCHAR2	20	YES	NO
HADDRESS	VARCHAR2	100	YES	NO
HPHONE_NUMBER	VARCHAR2	20	YES	NO
BBANK_NAME	VARCHAR2	20	YES	NO
BBADDRESS	VARCHAR2	20	YES	NO
BBPHONE_NUMBER	VARCHAR2	20	YES	NO
ENQ_NAME	VARCHAR2	20	YES	NO
ENQ_NUMBER	VARCHAR2	20	YES	NO

HISTORY_CITY

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	12	YES	NO
CITY_ID	NUMBER	10	YES	NO
CITY_IMAGE	BLOB	4000	YES	NO
CITY_HISTORY	VARCHAR2	2000	YES	NO

HOTEL_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	10	YES	NO
CITY_ID	NUMBER	10	YES	NO
HOTLE_NAME	VARCHAR2	20	YES	NO
RATING	VARCHAR2	12	YES	NO
ADDRESS	VARCHAR2	100	YES	NO
PHONE_NUMBER	VARCHAR2	10	YES	NO

LOGINDETAILS

Column Name	Data Type	Size	Accept Null Value	Unique
LOGINNAME	VARCHAR2	30	NO	YES
PASSWORD	VARCHAR2	30	NO	NO
LOGINTYPE	VARCHAR2	20	YES	NO
FIRST_NAME	VARCHAR2	40	YES	NO
LAST_NAME	VARCHAR2	40	YES	NO
LOGIN_STATUS	NUMBER	30	YES	NO
REG_DATE	DATE	7	YES	NO
SQUESTIONID	NUMBER	5	YES	NO
SANSWER	VARCHAR2	40	YES	NO
PASSMODIFIEDDATE	DATE	7	YES	NO
FIRSTLOGIN	NUMBER	5	YES	NO

LOGIN_AUDIT

Column Name	Data Type	Size	Accept Null Value	Unique
LOGINID	VARCHAR2	40	YES	NO
LOGINDATE	DATE	7	YES	NO
LOGIN_DESC	VARCHAR2	40	YES	NO

LOGIN_PROFILE

Column Name	Data Type	Size	Accept Null Value	Unique
LOGINID	VARCHAR2	40	YES	NO
BIRTHDATE	DATE	7	YES	NO
CITY	VARCHAR2	40	YES	NO
STATE	VARCHAR2	40	YES	NO
COUNTRY	VARCHAR2	40	YES	NO
EMAIL	VARCHAR2	40	YES	NO

MAILS

Column Name	Data Type	Size	Accept Null Value	Unique
MESSAGEID	NUMBER	4	NO	YES
FROMNAME	VARCHAR2	20	YES	NO
TONAME	VARCHAR2	20	YES	NO
SENDDATE	DATE	7	YES	NO
SUBJECT	VARCHAR2	30	YES	NO
MESSAGE	VARCHAR2	200	YES	NO
SENDERSTATUS	NUMBER	2	YES	NO
RECEIVERSTATUS	NUMBER	2	YES	NO

MINISTERS

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	5	YES	NO
MINISTRY	VARCHAR2	30	YES	NO
MINISTER	VARCHAR2	30	YES	NO

PLACES

Column Name	Data Type	Size	Accept Null Value	Unique
CITYID	NUMBER	10	YES	NO
PLACEID	NUMBER	10	YES	YES
PLACE	VARCHAR2	50	YES	NO
LOCATION	VARCHAR2	150	YES	NO
HISTORY	VARCHAR2	2000	YES	NO
IMG	BLOB	4000	YES	NO
PHONE	VARCHAR2	15	YES	NO

QUESTION_BASE

Column Name	Data Type	Size	Accept Null Value	Unique
QUESTION_ID	NUMBER	22	NO	YES
QUESTION_DETAIL	VARCHAR2	70	YES	NO

SHOPS

Column Name	Data Type	Size	Accept Null Value	Unique
SHOPID	NUMBER	5	YES	YES
CITYID	NUMBER	10	YES	NO
SHOPNAME	VARCHAR2	30	YES	NO
ADDR	VARCHAR2	200	YES	NO
FAMOUSFOR	VARCHAR2	30	YES	NO
OFFERS	VARCHAR2	300	YES	NO
PHONE	VARCHAR2	20	YES	NO

STATE_INFO

Column Name	Data Type	Size	Accept Null Value	Unique
STATE_ID	NUMBER	12	NO	YES
STATE_NAME	VARCHAR2	20	YES	NO
STATE_CAPITAL	VARCHAR2	20	YES	NO
CM	VARCHAR2	30	YES	NO
GOVERNOR	VARCHAR2	40	YES	NO
SITEMAP	BLOB	4000	YES	NO
LANGUAGES	VARCHAR2	50	YES	NO

SUBSCRIPTION

Column Name	Data Type	Size	Accept Null Value	Unique
SID	NUMBER	6	NO	YES
USERNAME	VARCHAR2	40	YES	NO
EMAIL	VARCHAR2	40	YES	NO
CITY_ID	NUMBER	10	YES	NO
ALERTTYPE	VARCHAR2	40	YES	NO

THEATRES

Column Name	Data Type	Size	Accept Null Value	Unique
THEATREID	NUMBER	5	YES	YES
CITYID	NUMBER	10	YES	NO
THEATRENAME	VARCHAR2	30	YES	NO
CINEMA	VARCHAR2	30	YES	NO
CINEMAIMG	BLOB	4000	YES	NO
MORSHOWTIME	VARCHAR2	30	YES	NO
MATSHOWTIME	VARCHAR2	30	YES	NO
FIRSHOWTIME	VARCHAR2	30	YES	NO
SECSHOWTIME	VARCHAR2	30	YES	NO
ADDRESS	VARCHAR2	200	YES	NO
PHONE	VARCHAR2	15	YES	NO

Coding

Sample Code

```
   <HTML>
<HEAD>
<TITLE>travel001</TITLE>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=iso-8859-1">
<style type="text/css">
<!--
.style1 {
	color: #333333;
	font-weight: bold;
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 12px;
}
.style2 {
	color: #FF0000;
	font-weight: bold;
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 16px;
}
.style3 {
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 10px;
	color: #FFFFFF;
}
.style5 {color: #333333; font-weight: bold; font-family:
Verdana, Arial, Helvetica, sans-serif; font-size: 14px; }
.style6 {
	color: #590660;
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 12px;
	font-weight: bold;
}
.style13 {font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 11px; color: #590660; }
.style14 {
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 10px;
```

```
	color: #0B6191;
	font-weight: bold;
}
.style15 {font-size: 10px; color: #0B6191; font-family:
Verdana, Arial, Helvetica, sans-serif;}
.style16 {
	color: #0B6191;
	font-weight: bold;
	font-size: 14px;
	font-family: Verdana, Arial, Helvetica, sans-serif;
}
.style18 {
	font-size: 11px;
	color: #000000;
}
.style20 {font-size: 10px; color: #590660; font-family:
Verdana, Arial, Helvetica, sans-serif;}
.style22 {
	color: #0E7DBB;
	font-weight: bold;
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 14px;
}
.style23 {
	color: #590660;
	font-weight: bold;
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 10px;
}
.style24 {
	color: #0E7DBB;
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 11px;
}
body {
	background-image: url(images/main_bg.jpg);
}
a:link {
	color: #000000;
```

```
	text-decoration: none;
}
a:visited {
	text-decoration: none;
	color: #000000;
}
a:hover {
	text-decoration: none;
	color: #000000;
}
a:active {
	text-decoration: none;
	color: #000000;
}<img src="img/bg.gif"></img>
.style31 {
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-weight: bold;
	font-size: 13px;
	color: #003366;
}
.style32 {
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 11px;
}
-->
</style>
</HEAD>
<BODY BGCOLOR=#FFFFFF LEFTMARGIN=0 TOPMARGIN=0
MARGINWIDTH=0 MARGINHEIGHT=0>
<!-- ImageReady Slices (travel001.psd) -->
<table width="770" border="0" align="center" cellpadding="0"
cellspacing="0" bgcolor="#FFFFFF">
  <tr>
    <td align="center" valign="top"><table width="770"
border="0" cellspacing="0" cellpadding="0">
      <tr>
        <td width="123"><IMG SRC="images/index_01.jpg"
WIDTH=123 HEIGHT=29 ALT=""></td>
        <td width="79" background="images/butbg.jpg"><div
```

```
align="center" class="style31">Home</div></td>
      <td width="107" background="images/butbg.jpg"><div
align="center"><span
class="style31">Packages</span></div></td>
      <td width="107" background="images/butbg.jpg"><div
align="center"><span
class="style31">Specials</span></div></td>
      <td width="112" background="images/butbg.jpg"><div
align="center"><span class="style31">About
us</span></div></td>
      <td width="119" background="images/butbg.jpg"><div
align="center"><span class="style31">Contact
us</span></div></td>
      <td><IMG SRC="images/index_07.jpg" WIDTH=123
HEIGHT=29 ALT=""></td>
    </tr>
  </table>
    <table width="770" border="0" cellspacing="0"
cellpadding="0">
      <tr>
        <td width="61"><IMG SRC="images/index_08.jpg"
WIDTH=61 HEIGHT=196 ALT=""></td>
        <td width="62"><IMG SRC="images/index_09.jpg"
WIDTH=62 HEIGHT=196 ALT="">
        <td width="66"><img
src="images/index_10.jpg"></img></td>
        <td width="135"><IMG SRC="images/index_11.jpg"
WIDTH=135 HEIGHT=196 ALT=""></td>
        <td width="299" valign="top"><table width="299"
border="0" cellspacing="0" cellpadding="0">
          <tr>
            <td><IMG SRC="images/index_12.jpg" WIDTH=299
HEIGHT=29 ALT=""></td>
          </tr>
          <tr>
            <td><IMG SRC="images/index_15.jpg" WIDTH=299
HEIGHT=75 ALT=""></td>
          </tr>
          <tr>
```

```
            <td><table width="299" border="0" cellspacing="0"
cellpadding="0">
              <tr>
                <td width="228"
background="images/index_16.jpg"><table width="213"
border="0" cellspacing="0" cellpadding="0" height="62">
                  <tr>
                    <td height="30" colspan="2"
align="center"><span class="style1">Destination
Search</span></td>
                  </tr>
                  <tr>
                    <td height="30" align="right"><input
name="textfield" type="text" size="22"></td>

                </table></td>
                <td><table width="71" border="0"
cellspacing="0" cellpadding="0">
                  <tr>
                    <td><IMG SRC="images/index_17.jpg"
WIDTH=71 HEIGHT=52 ALT=""></td>
                  </tr>
                  <tr>
                    <td><IMG SRC="images/index_18.jpg"
WIDTH=71 HEIGHT=19 ALT=""></td>
                  </tr>
                  <tr>
                    <td><IMG SRC="images/index_19.jpg"
WIDTH=71 HEIGHT=21 ALT=""></td>
                  </tr>
                </table></td>
              </tr>
            </table></td>
          </tr>
        </table></td>
        <td width="70"><IMG SRC="images/index_13.jpg"
WIDTH=70 HEIGHT=196 ALT=""></td>
        <td><IMG SRC="images/index_14.jpg" WIDTH=77
HEIGHT=196 ALT=""></td>
```

```
        </tr>
      </table>
      <table width="770" border="0" cellpadding="0"
cellspacing="0" background="images/index_21.jpg">
        <tr>
          <td><IMG SRC="images/index_21.jpg" WIDTH=10
HEIGHT=10 ALT=""></td>
        </tr>
      </table>
      <table width="770" border="0" cellspacing="0"
cellpadding="0">
        <tr>
          <td width="770" valign="top" bgcolor="#87ceeb">
          <pre>

          </pre>

          <table align="center">
 <%@ page
import="com.smart.struts.dao.*,com.smart.util.*,java.util.*,co
m.smart.struts.form.*"%>
<html><head>
<meta http-equiv="content-type" content="text/html;
charset=ISO-8859-1">
  <meta name="verify-v1"
content="bGDK2jk+Vhn+Kk2NrZFue5EbYn1Fed4v/Ej+U2bg7w
c=">
<script type="text/javascript"
src="scripts/project.js"></script>

              <script type="text/javascript"
src="scripts/submit.js"></script>
<script language="javascript">
function formCheck(formobj){

        var fieldRequired = Array("username","password");

        var fieldDescription = Array("Login Name","Password");
        // dialog message
```

```
	var alertMsg = "Please complete the following fields:\n";

	var l_Msg = alertMsg.length;


	for (var i = 0; i < fieldRequired.length; i++){
		var obj = formobj.elements[fieldRequired[i]];
		if (obj){
			switch(obj.type){
			case "select-one":
				if (obj.selectedIndex == -1 ||
obj.options[obj.selectedIndex].text == ""){
					alertMsg += " - " + fieldDescription[i]
+ "\n";
				}
				break;
			case "select-multiple":
				if (obj.selectedIndex == -1){
					alertMsg += " - " + fieldDescription[i]
+ "\n";
				}
				break;
			case "text":
			case "textarea":
				if (obj.value == "" || obj.value == null){
					alertMsg += " - " + fieldDescription[i]
+ "\n";
				}
				break;
			default:
			}
			if (obj.type == undefined){
				var blnchecked = false;
				for (var j = 0; j < obj.length; j++){
					if (obj[j].checked){
						blnchecked = true;
					}
				}
				if (!blnchecked){
```

```
                        alertMsg += " - " + fieldDescription[i]
+ "\n";
                    }
                }

            }
        }

        if (alertMsg.length == l_Msg){
            return true;
        }else{
            alert(alertMsg);
            return false;
        }
}

</script>
<script>



A=document.getElementById
B=document.all;
C=document.layers;
T1=new
Array("images/trail1.gif",38,35,"images/trail2.gif",30,31,"imag
es/trail3.gif",28,26,"images/trail4.gif",22,21,"images/trail5.gif"
,16,16,"images/trail6.gif",10,10)

var offsetx=15 //x offset of trail from mouse pointer
var offsety=10 //y offset of trail from mouse pointer

nos=parseInt(T1.length/3)
rate=50
ie5fix1=0;
ie5fix2=0;
rightedge=B? document.body.clientWidth-T1[1] :
window.innerWidth-T1[1]-20
bottomedge=B?
```

```
document.body.scrollTop+document.body.clientHeight-T1[2] :
window.pageYOffset+window.innerHeight-T1[2]

for (i=0;i<nos;i++){
createContainer("CUR"+i,i*10,i*10,i*3+1,i*3+2,"","<img
src='"+T1[i*3]+"' width="+T1[(i*3+1)]+"
height="+T1[(i*3+2)]+" border=0>")
}

function createContainer(N,Xp,Yp,W,H,At,HT,Op,St){
with (document){
write((!A && !B) ? "<layer id='"+N+"' left="+Xp+"
top="+Yp+" width="+W+" height="+H : "<div id='"+N+"'"+"
style='position:absolute;left:"+Xp+"; top:"+Yp+";
width:"+W+"; height:"+H+"; ");
if(St){
if (C)
write(" style='");
write(St+";' ")
}
else write((A || B)?"'":"");
write((At)? At+">" : ">");
write((HT) ? HT : "");
if (!Op)
closeContainer(N)
}
}

function closeContainer(){
document.write((A || B)?"</div>":"</layer>")
}

function getXpos(N){
if (A)
return parseInt(document.getElementById(N).style.left)
else if (B)
return parseInt(B[N].style.left)
else
return C[N].left
```

```
}

function getYpos(N){
if (A)
return parseInt(document.getElementById(N).style.top)
else if (B)
return parseInt(B[N].style.top)
else
return C[N].top
}

function moveContainer(N,DX,DY){
c=(A)? document.getElementById(N).style : (B)? B[N].style :
(C)? C[N] : "";
if (!B){
rightedge=window.innerWidth-T1[1]-20
bottomedge=window.pageYOffset+window.innerHeight-T1[2]
}
c.left=Math.min(rightedge, DX+offsetx);
c.top=Math.min(bottomedge, DY+offsety);
}
function cycle(){
//if (IE5)
if (document.all&&window.print){
ie5fix1=document.body.scrollLeft;
ie5fix2=document.body.scrollTop;
}
for (i=0;i<(nos-1);i++){
moveContainer("CUR"+i,getXpos("CUR"+(i+1)),getYpos("CUR"
+(i+1)))
}
}

function newPos(e){
moveContainer("CUR"+(nos-
1),(B)?event.clientX+ie5fix1:e.pageX+2,(B)?event.clientY+ie5f
ix2:e.pageY+2)
}
```

```
function getedgesIE(){
rightedge=document.body.clientWidth-T1[1]
bottomedge=document.body.scrollHeight-T1[2]
}

if (B){
window.onload=getedgesIE
window.onresize=getedgesIE
}

if(document.layers)
document.captureEvents(Event.MOUSEMOVE)
document.onmousemove=newPos
setInterval("cycle()",rate)
</script>
<style>
<!--
A           { color: #C0C0C0; font-weight:bold; text-
decoration: none; }
A:link          { color: #C0C0C0; font-weight:bold; text-
decoration: none; }
A:visited       { color: #C0C0C0; font-weight:bold; text-
decoration: none; }
A:active        { color: #358391; font-weight:bold; text-
decoration: none;  }
A:hover         { color: #358391; font-weight:bold; text-
decoration: none;  }

body, td, tr{
font-family: verdana;
color:#FFFFFF;
font-size:11;
font-weight:normal;
}

//-->
</style>

<script language="JavaScript" type="text/JavaScript"
```

```
src="java/javascript.js"></script>
<SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript"
SRC="java/base.js"></SCRIPT>
<script language="JavaScript" src="java/tabel.js"
type="text/javascript"></script>

<LINK REL="SHORTCUT ICON" type="picture"
title="SHORTCUT ICON"
HREF="http://www.veebimaja.net/vm01.png" />
</head>


      <table align="center">
      <td>
    <form action="login.do" method="post" name="register"
onsubmit="return formCheck(this)">
                  <table width="150" border="2" align="center"
bordercolor="red">
                    <tr>
                      <td><div align="right" class="style25">

                                    <%
if(request.getParameter("status")!=null)
                                    {%>

<strong><%=request.getParameter("status")%></strong>
                                    <%}%>
                                    </div></td>
                    </tr>
                    <tr>
                      <td><table width="350" height="206"
border="0" align="center" bordercolor="#CD817E"
bgcolor="#FAF9DE">
                        <tr>
                          <td width="1" height="57"
valign="top"></td>
                          <td width="280"><div align="center"
style="font-family: verdana; font-size: 14px; font-weight:
bold; color: rgb(193, 110, 238);"><u>L O G I
```

```
N</u></div></td>
                      <td width="29"></td>
                    </tr>
                    <tr>
                      <td height="120" align="right"></td>
                      <td><table width="226" border="0"
align="center">
                          <tr>
                            <td width="79"><font style="font-
family:verdana;font-size:14px;font-
weight:bold;color:#5aff19">Username</font></td>
                            <td width="189"><input type="text"
name="username" value="USER NAME" onfocus="value=''"
onmousedown="value='USER NAME'" >
</td>
                          </tr>
                          <tr>
                            <td><font style="font-
family:verdana;font-size:14px;font-
weight:bold;color:#5aff19">Password</font></td>
                            <td><label>
                              <input type="password"
name="password" value="******" onfocus="value=''">
                            </label></td>
                          </tr>
                          <tr>
                            <td colspan="2"><label>
                                <div align="center">
                                  <input type="submit"
name="Submit" value="Sign In">
                                   
                                  <input type="reset"
name="Input2" value="Clear">

                      </table></td>
                      <td> </td>
                    </tr>
                    <tr>
                      <td height="21"> </td>
```

```
                    <td valign="baseline">          <div
align="center"><span class="style26"><a
href="RecoverPassword.jsp">Forgot Password !
!......</a></span> <br>
                        <img src="images/regisbot.gif" alt="a"
width="280" height="3"></div></td>
                    <td> </td>
                  </tr>
                 <tr>
                    <td height="21"> </td>
                    <td valign="baseline">          <div
align="center"><span class="style26"><a
href="Changepassword.jsp">Change Password !
!......</a></span> <br>
                        <img src="images/regisbot.gif" alt="a"
width="280" height="3"></div></td>
                    <td> </td>
                  </tr>

                 </table></td>

                </tr>
              </table>
              <table align="center">
              <br/>
             <table align="center">
              <p align="center"><a href="Register.jsp"><img
src="./images/register.gif" width="126" height="40"
border="0"></a> </p>

              </table>
</table>
<pre>




</pre>
```

```
    </table>

      </table>
      <table width="770" border="0" cellspacing="0"
cellpadding="0" align="center">
        <tr>
          <td width="250" align="center"><br></td>
          <td width="520" align="center"
valign="top"><br></td>
        </tr>
      </table><br><table width="800" border="0"
cellspacing="0" cellpadding="0" align="center">
        <tr>
          <td height="20" align="center"><span
class="style32"><a
href="http://www.legalhelpers.com/chapter-13-
bankruptcy/chapter13.html"><br></a></span></td>
        </tr>
      </table></td>
  </tr>
</table>
<!-- End ImageReady Slices -->
```

Implementation

Technologies Used

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

Hyper Text Markup Language

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 headings 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-attribute 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 stylesheet, though sometimes this can be too cumbersome for a simple ad hoc application of styled properties. The title is used to attach subtextual explanation to an element. In most browsers this title attribute is displayed as what is often referred to as a tooltip. 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

<SCRIPTS>. </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 Vs Java

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.

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.

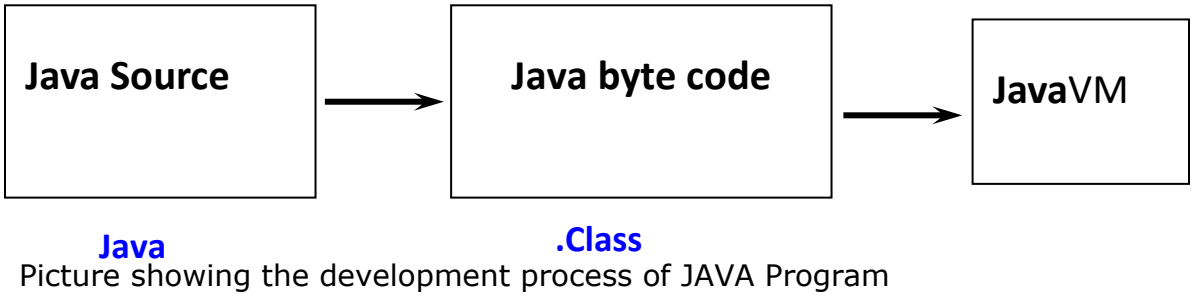
Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-fly compilation of byte code into native code. Sun has

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

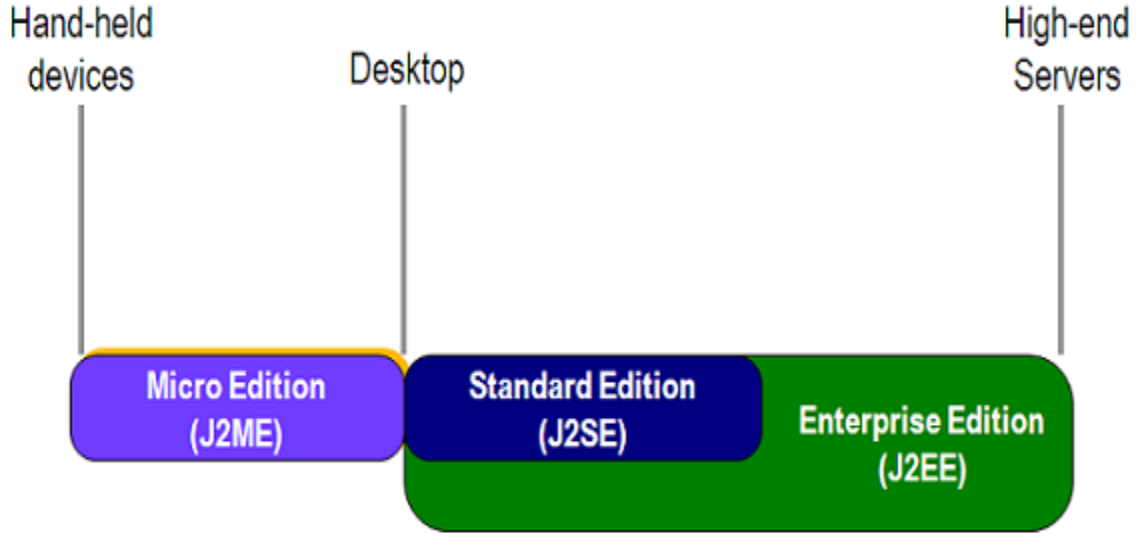


Picture showing the development process of JAVA Program

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.

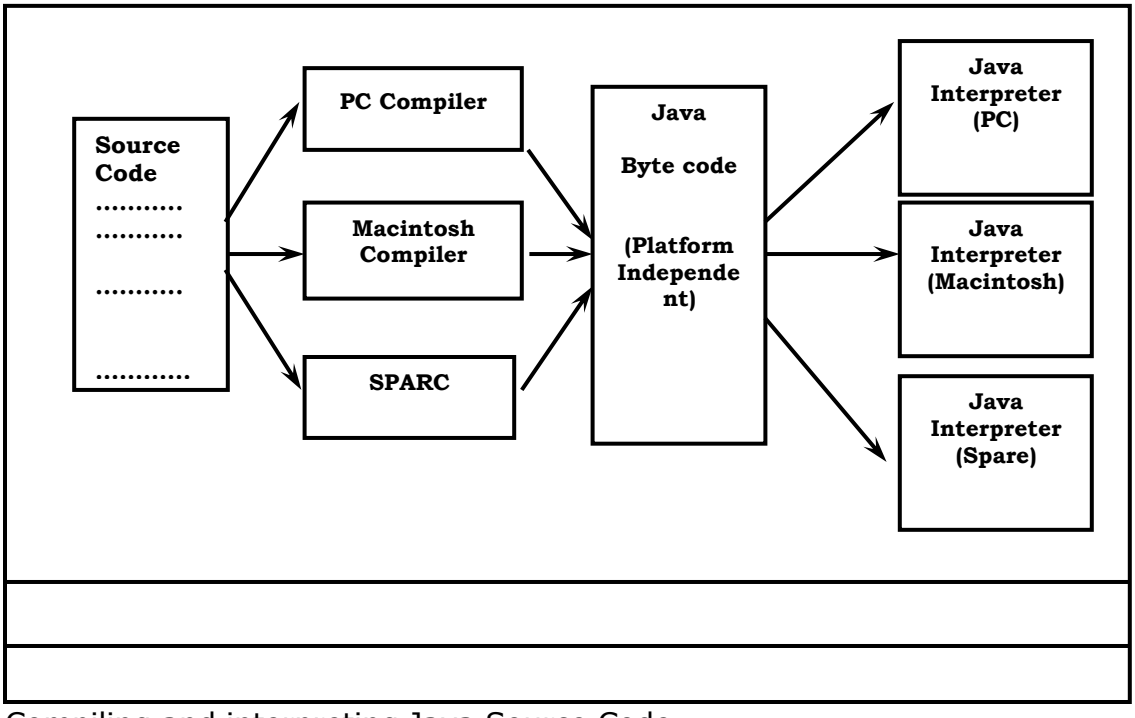
Java Architecture

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.



Compiling and interpreting Java Source Code

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

Simple

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

Object-Oriented

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

Robust

The multi-platform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and run time.

Java virtually eliminates the problems of memory management and de-allocation, which is completely automatic. In a well-written Java program, all run time errors can –and should –be managed by your program.

Java Database Connectivity

What Is JDBC?

JDBC is a Java API for executing SQL statements. (As a point of interest, JDBC is a trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the Java programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API.

Using JDBC, it is easy to send SQL statements to virtually any relational database. One can write a single program using the JDBC API, and the program will be able to send SQL statements to the appropriate database. The combinations of Java and JDBC lets a programmer write it once and run it anywhere.

What Does JDBC Do?

Simply put, JDBC makes it possible to do three things:

- Establish a connection with a database
- Send SQL statements
- Process the results.

JDBC versus ODBC and other APIs

At this point, Microsoft's ODBC (Open Database Connectivity) API is that probably the most widely used programming interface for accessing relational databases. It offers the ability to connect to almost all databases on almost all platforms.

So why not just use ODBC from Java? The answer is that you can use ODBC from Java, but this is best done with the help of JDBC in the form of the JDBC-ODBC Bridge, which we will cover shortly. The question now becomes "Why do you need JDBC?" There are several answers to this question:

1. ODBC is not appropriate for direct use from Java because it uses a C interface. Calls from Java to native C code have a number of drawbacks in

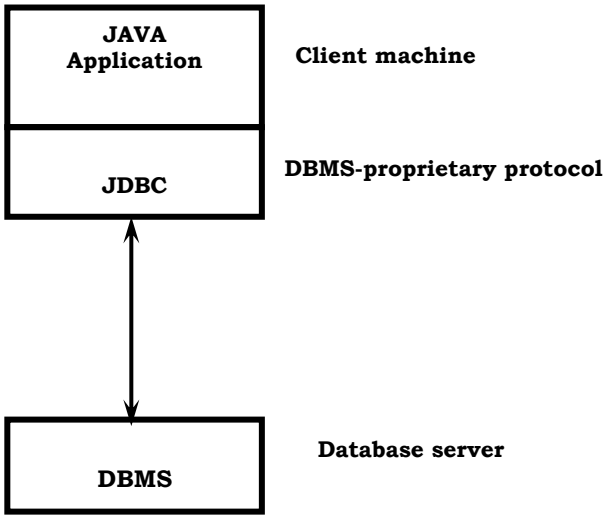
the security, implementation, robustness, and automatic portability of applications.

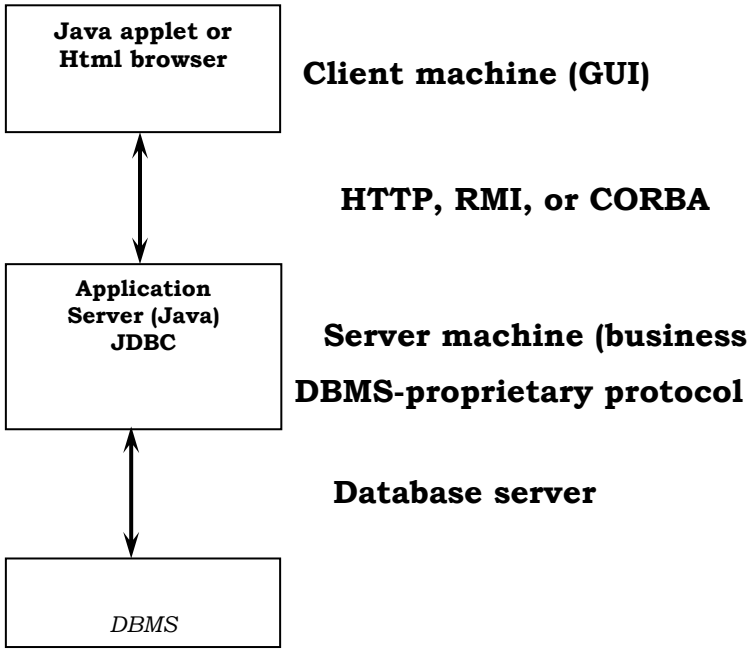
2. A literal translation of the ODBC C API into a Java API would not be desirable. For example, Java has no pointers, and ODBC makes copious use of them, including the notoriously error-prone generic pointer "void *". You can think of JDBC as ODBC translated into an object-oriented interface that is natural for Java programmers.
3. ODBC is hard to learn. It mixes simple and advanced features together, and it has complex options even for simple queries. JDBC, on the other hand, was designed to keep simple things simple while allowing more advanced capabilities where required.
4. A Java API like JDBC is needed in order to enable a "pure Java" solution. When ODBC is used, the ODBC driver manager and drivers must be manually installed on every client machine. When the JDBC driver is written completely in Java, however, JDBC code is automatically installable, portable, and secure on all Java platforms from network computers to mainframes.

Two-tier and Three-tier Models

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

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

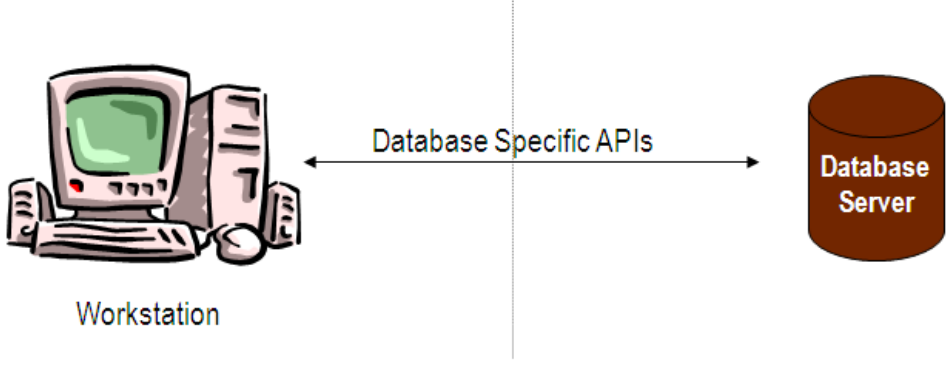




In the three-tier model, commands are sent to a "middle tier" of services, which then send SQL statements to the database. The database processes the SQL statements and sends the results back to the middle tier, which then sends them to the user. MIS directors find the three-tier model very attractive because the middle tier makes it possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is that when there is a middle tier, the user can employ an easy-to-use higher-level API which is translated by the middle tier into the appropriate low-level calls. Finally,

in many cases the three-tier architecture can provide performance advantages.

Until now the middle tier has typically been written in languages such as C or C++, which offer fast performance. However, with the introduction of optimizing compilers that translate Java byte code into efficient machine-specific code, it is becoming practical to implement the middle tier in Java. This is a big plus, making it possible to take advantage of Java's robustness, multithreading, and security features. JDBC is important to allow database access from a Java middle tier.



- Presentation, business and data model processing logic into client application
- Server is typically a database server
- Client sends SQL statements, retrieves raw data

JDBC Driver Types

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

- JDBC-ODBC bridge plus ODBC driver
- Native-API partly-Java driver
- JDBC-Net pure Java driver
- Native-protocol pure Java driver

JDBC-ODBC Bridge

If possible, use a Pure Java JDBC driver instead of the Bridge and an ODBC driver. This completely eliminates the client configuration required by ODBC. It also eliminates the potential that the Java VM could be corrupted by an error in the native code brought in by the Bridge (that is, the Bridge native library, the ODBC driver manager library, the ODBC driver library, and the database client library).

What Is the JDBC- ODBC Bridge?

The JDBC-ODBC Bridge is a JDBC driver, which implements JDBC operations by translating them into ODBC operations. To ODBC it appears as a normal application program. The Bridge implements JDBC for any database for which an ODBC driver is available. The Bridge is implemented as the

Sun.jdbc.odbc Java package and contains a native library used to access ODBC. The Bridge is a joint development of Innersole and Java Soft.

JDBC connectivity

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

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

Database

A database management system (DBMS) is computer software designed for the purpose of managing databases, a large set of structured data, and run operations on the data requested by numerous users. Typical examples of DBMSs include Oracle, DB2, Microsoft Access, Microsoft SQL Server, Firebird, PostgreSQL, MySQL, SQLite, FileMaker and Sybase Adaptive Server Enterprise. DBMSs are typically used by Database administrators in the creation of Database systems. Typical examples of DBMS use include accounting, human resources and customer support systems.

Originally found only in large companies with the computer hardware needed to support large data sets, DBMSs have more recently emerged as a fairly standard part of any company back office.

Description

A DBMS is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. A DBMS includes:

- ✓ A modeling language to define the schema of each database hosted in the DBMS, according to the DBMS data model.
    - The four most common types of organizations are the hierarchical, network, relational and object models. Inverted lists and other methods are also used. A given database management system may provide one or more of the four models. The optimal structure depends on the natural organization of the application's data, and on the application's requirements (which include transaction rate (speed), reliability, maintainability, scalability, and cost).
    - The dominant model in use today is the ad hoc one embedded in SQL, despite the objections of purists who believe this model is a corruption of the relational model, since it violates several of its fundamental principles for the sake of practicality and performance. Many DBMSs also support the Open Database Connectivity API that supports a standard way for programmers to access the DBMS.
- ✓ Data structures (fields, records, files and objects) optimized to deal with very large amounts of data stored on a permanent data storage device (which implies relatively slow access compared to volatile main memory).

- ✓ A database query language and report writer to allow users to interactively interrogate the database, analyze its data and update it according to the users privileges on data.
    - It also controls the security of the database.
    - Data security prevents unauthorized users from viewing or updating the database. Using passwords, users are allowed access to the entire database or subsets of it called subschemas. For example, an employee database can contain all the data about an individual employee, but one group of users may be authorized to view only payroll data, while others are allowed access to only work history and medical data.
    - If the DBMS provides a way to interactively enter and update the database, as well as interrogate it, this capability allows for managing personal databases. However, it may not leave an audit trail of actions or provide the kinds of controls necessary in a multi-user organization. These controls are only available when a set of application programs are customized for each data entry and updating function.

- ✓ A transaction mechanism, that ideally would guarantee the ACID properties, in order to ensure data integrity, despite concurrent user accesses (concurrency control), and faults (fault tolerance).
    - It also maintains the integrity of the data in the database.
    - The DBMS can maintain the integrity of the database by not allowing more than one user to update the same record at the same time. The DBMS can help prevent duplicate records via unique index constraints; for example, no two customers with the same customer numbers (key fields) can be entered into the database. See ACID properties for more information (Redundancy avoidance).

The DBMS accepts requests for data from the application program and instructs the operating system to transfer the appropriate data.

When a DBMS is used, information systems can be changed much more easily as the organization's information requirements change. New categories of data can be added to the database without disruption to the existing system.

Organizations may use one kind of DBMS for daily transaction processing and then move the detail onto another computer that uses another DBMS better suited for random inquiries and analysis. Overall systems design decisions are performed by data administrators and systems analysts. Detailed database design is performed by database administrators.

Database servers are specially designed computers that hold the actual databases and run only the DBMS and related software. Database servers are usually multiprocessor computers, with RAID disk arrays used for stable storage. Connected to one or more servers via a high-speed channel, hardware database accelerators are also used in large volume transaction processing environments.

DBMSs are found at the heart of most database applications. Sometimes DBMSs are built around a private multitasking kernel with built-in networking support although nowadays these functions are left to the operating system.

SQL

Structured Query Language (SQL) is the language used to manipulate relational databases. SQL is tied very closely with the relational model.

In the relational model, data is stored in structures called relations or tables. SQL statements are issued for the purpose of:

Data definition: Defining tables and structures in the database (DDL used to create, alter and drop schema objects such as tables and indexes).

Data manipulation: Used to manipulate the data within those schema objects (DML Inserting, Updating, Deleting the data, and Querying the Database).

A schema is a collection of database objects that can include: tables, views, indexes and sequences

List of SQL statements that can be issued against an Oracle database schema are:

- ALTER - Change an existing table, view or index definition (DDL)
- AUDIT - Track the changes made to a table (DDL)
- COMMENT - Add a comment to a table or column in a table (DDL)
- COMMIT - Make all recent changes permanent (DML - transactional)
- CREATE - Create new database objects such as tables or views (DDL)
- DELETE - Delete rows from a database table (DML)
- DROP - Drop a database object such as a table, view or index (DDL)
- GRANT - Allow another user to access database objects such as tables or views (DDL)
- INSERT - Insert new data into a database table (DML)
- No AUDIT - Turn off the auditing function (DDL)
- REVOKE - Disallow a user access to database objects such as tables and views (DDL)
- ROLLBACK - Undo any recent changes to the database (DML - Transactional)
- SELECT - Retrieve data from a database table (DML)
- TRUNCATE - Delete all rows from a database table (can not be rolled back) (DML)
- UPDATE - Change the values of some data items in a database table (DML)

SERVLETS

Introduction

The Java web server is JavaSoft's own web Server. The Java web server is just a part of a larger framework, intended to provide you not just with a web server, but also with tools. To build customized network servers for any Internet or Intranet client/server system. Servlets are to a web server, how applets are to the browser.

About Servlets

Servlets provide a Java-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, pluggable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

For example, an HTTP Servlets can be used to generate dynamic HTML content. When you use Servlets to do dynamic content you get the following advantages:

- They’re faster and cleaner than CGI scripts
- They use a standard API (the Servlets API)
- They provide all the advantages of Java (run on a variety of servers without needing to be rewritten).

Attractiveness of Servlets

There are many features of Servlets that make them easy and attractive to use. These include:

- Easily configured using the GUI-based Admin tool

- Can be loaded and invoked from a local disk or remotely across the network.
- Can be linked together, or chained, so that one Servlets can call another Servlets, or several Servlets in sequence.
- Can be called dynamically from within HTML pages, using server-side include tags.
- Are secure - even when downloading across the network, the Servlets security model and Servlets sandbox protect your system from unfriendly behavior.

Advantages of the Servlet API

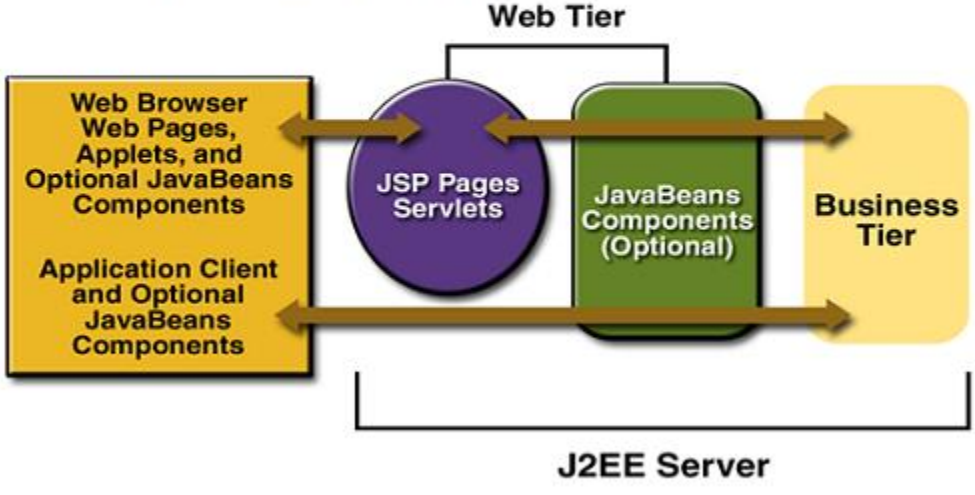
One of the great advantages of the Servlet API is protocol independence. It assumes nothing about:

- The protocol being used to transmit on the net
- How it is loaded
- The server environment it will be running in

These qualities are important, because it allows the Servlet API to be embedded in many different kinds of servers. There are other advantages to the Servlet API as well. These include:

- It’s extensible - you can inherit all your functionality from the base classes made available to you.
- It’s simple, small, and easy to use.

- Web components
  - Servlets or JSP pages
  - JavaBeans (optional)



Features of Servlets:

- Servlets are persistent. Servlet are loaded only by the web server and can maintain services between requests.
- Servlets are fast. Since Servlets only need to be loaded once, they offer much better performance over their CGI counterparts.
- Servlets are platform independent.
- Servlets are extensible. Java is a robust, object-oriented programming language, which easily can be extended to suit your needs
- Servlets are secure.
- Servlets can be used with a variety of clients.

Loading Servlets:

Servlets can be loaded from three places

From a directory that is on the CLASSPATH. The CLASSPATH of the JavaWebServer includes service root/classes/ which is where the system classes reside.

From the <SERVICE_ROOT /Servlets/ directory. This is *not* in the server's class path. A class loader is used to create Servlets from this directory. New Servlets can be added - existing Servlets can be recompiled and the server will notice these changes.

From a remote location, for this a code base like http: // nine.eng / classes / foo / is required in addition to the Servlets class name. Refer to the admin GUI docs on Servlet section to see how to set this up.

Loading Remote Servlets

Remote Servlets can be loaded by:

1. Configuring the Admin Tool to setup automatic loading of remote Servlets
2. Setting up server side include tags in. shtml files
3. Defining a filter chain configuration

Invoking Servlets

A Servlet invoker is a Servlet that invokes the "service" method on a named Servlet. If the Servlet is not loaded in the server, then the invoker first loads the Servlet (either from local disk or from the network) and the then invokes the "service" method. Also, like applets, local Servlets in the server can be identified by just the class name. In other words, if a Servlet name is not absolute, it is treated as local.
A client can invoke Servlets in the following ways**:**

- The client can ask for a document that is served by the Servlet.

- The client (browser) can invoke the Servlet directly using a URL, once it has been mapped using the Servlet Aliases section of the admin GUI.
- The Servlet can be invoked through server side include tags.
- The Servlet can be invoked by placing it in the Servlets/ directory.
- The Servlet can be invoked by using it in a filter chain.

Java Server Pages (JSP)

Java server Pages is a simple, yet powerful technology for creating and maintaining dynamic-content web pages. Based on the Java programming language, Java Server Pages offers proven portability, open standards, and a mature re-usable component model. The Java Server Pages architecture enables the separation of content generation from content presentation. This separation not eases maintenance headaches; it also allows web team members to focus on their areas of expertise. Now, web page designer can concentrate on layout, and web application designers on programming, with minimal concern about impacting each other's work.

Features of JSP

Portability:

Java Server Pages files can be run on any web server or web-enabled application server that provides support for them. Dubbed the JSP engine, this support involves recognition, translation, and management of the Java Server Page lifecycle and its interaction components.

Components

It was mentioned earlier that the Java Server Pages architecture can include reusable Java components. The architecture also allows for the embedding of a scripting language directly into the Java Server Pages file. The components current supported include Java Beans, and Servlets.

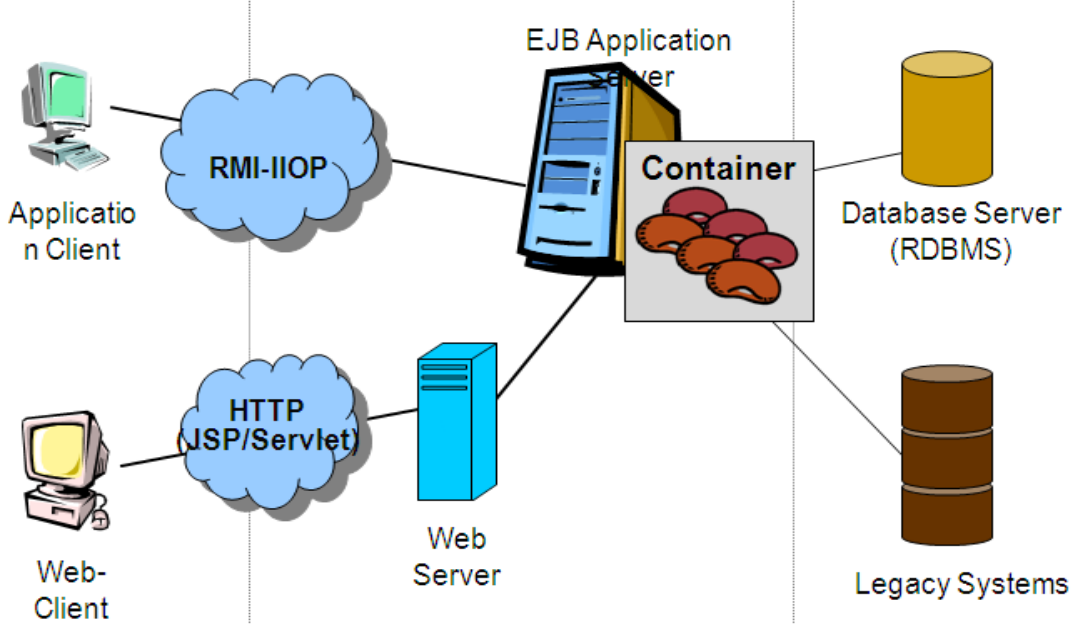
Processing

A Java Server Pages file is essentially an HTML document with JSP scripting or tags. The Java Server Pages file has a JSP extension to the server as a Java Server Pages file. Before the page is served, the Java Server Pages syntax is parsed and processed into a Servlet on the server side. The Servlet that is generated outputs real content in straight HTML for responding to the client.

Access Models:

A Java Server Pages file may be accessed in at least two different ways. A client's request comes directly into a Java Server Page. In this scenario, suppose the page accesses reusable Java Bean components that perform particular well-defined computations like accessing a database. The result of the Beans computations, called result sets is stored within the Bean as properties. The page uses such Beans to generate dynamic content and present it back to the client.

In both of the above cases, the page could also contain any valid Java code. Java Server Pages architecture encourages separation of content from presentation.



Steps in the execution of a JSP Application:

1. The client sends a request to the web server for a JSP file by giving the name of the JSP file within the form tag of a HTML page.

2. This request is transferred to the JavaWebServer. At the server side JavaWebServer receives the request and if it is a request for a jsp file server gives this request to the JSP engine.
3. JSP engine is program which can under stands the tags of the jsp and then it converts those tags into a Servlet program and it is stored at the server side. This Servlet is loaded in the memory and then it is executed and the result is given back to the JavaWebServer and then it is transferred back to the result is given back to the JavaWebServer and then it is transferred back to the client.

Eclipse IDE

Eclipse is an open-source software framework written primarily in Java. In its default form it is an Integrated Development Environment (IDE) for Java developers, consisting of the Java Development Tools (JDT) and the Eclipse Compiler for Java (ECJ). Users can extend its capabilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Language packs are available for over a dozen languages.

Architecture

The basis for Eclipse is the Rich Client Platform (RCP). The following components constitute the rich client platform:

- ✓ OSGi - a standard bundling framework
- ✓ Core platform - boot Eclipse, run plug-ins
- ✓ the Standard Widget Toolkit (SWT) - a portable widget toolkit
- ✓ JFace - viewer classes to bring model view controller programming to SWT, file buffers, text handling, text editors

- ✓ the Eclipse Workbench - views, editors, perspectives, wizards

Eclipse's widgets are implemented by a widget toolkit for Java called SWT, unlike most Java applications, which use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse's user interface also leverages an intermediate GUI layer called JFace, which simplifies the construction of applications based on SWT.

Eclipse employs plug-ins in order to provide all of its functionality on top of (and including) the rich client platform, in contrast to some other applications where functionality is typically hard coded. This plug-in mechanism is a lightweight software componentry framework. In addition to allowing Eclipse to be extended using other programming languages such as C and Python, the plug-in framework allows Eclipse to work with typesetting languages like LaTeX, networking applications such as telnet, and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK.

The key to the seamless integration of tools with Eclipse is the plugin. With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that a plug-in you develop integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are created equal.

The Eclipse SDK includes the Eclipse Java Development Tools, offering an IDE with a built-in incremental Java compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a workspace, in this case a set of metadata over a flat filespace allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards. The Visual Editor project allows interfaces to be created interactively, hence allowing Eclipse to be used as a RAD tool.

The following is a list of notable projects and plugins for the Eclipse IDE.

These projects are maintained by the Eclipse community and hosted by the Eclipse Foundation.

1. Core projects

Rich Client Platform (Platform) is the core framework that all other Eclipse projects are built on.

Java Development Tools (JDT) provides support for core Java SE. This includes a standalone fast incremental compiler.

Tools projects

C/C++ Development Tools (CDT) adds support for C/C++ syntax highlighting, code formatting, debugger integration and project structures. Unlike the JDT project, the CDT project does not add a compiler and relies on an external tool chain.

Graphical Editing Framework (GEF) allows developers to build standalone graphical tools. Example use include circuit diagram design tools, activity diagram editors and WYSIWYG document editors.

Web projects

J2EE Standard Tools (JST) extends the core JDT to include support for Java EE projects. This includes EJBs, JSPs and Servlets.

PHP Development Tools (PDT)

Web Standard Tools (WST) adds standards compliant web development tools. These tools include editors for XML, HTML and CSS.

Modelling projects

Eclipse Modeling Framework (EMF) a modeling framework and code generation facility for building tools and other applications based on a structured data

model, from a model specification described in XMI.

Graphical Modeling Framework (GMF) is a generative component and runtime infrastructure for developing graphical editors based on EMF and GEF.

Other projects

Test and Performance Tools Platform (TPTP) which provides a platform that allows software developers to build test and performance tools, such as debuggers, profilers and benchmarking applications.

Business Intelligence and Reporting Tools Project (BIRT), an Eclipse-based open-source reporting system for web applications, especially those based on Java EE.

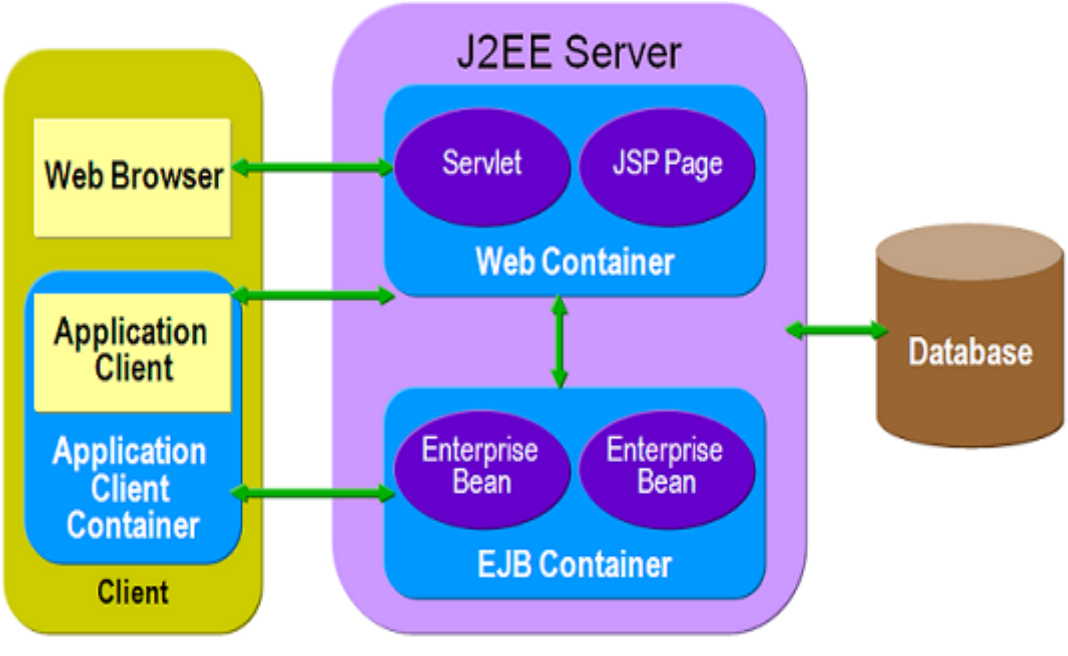
Applications Server

An application server is a software engine that delivers applications to client computers or devices, typically through the Internet and using the Hypertext Transfer Protocol. Application servers are distinguished from web servers by the extensive use of server-side dynamic content and frequent integration with database engines.

Common features

Application server products typically bundle middleware to enable applications to intercommunicate with dependent applications, like web servers, database management systems, and chart programs. Some application servers also provide an API, making them operating system independent. Portals are a common application server mechanism by which a single point of entry is provided to multiple devices.

Java application servers



Java EE Servers

Following the success of the Java platform, the term application server sometimes refers to a Java Platform--Enterprise Edition (J2EE) or Java EE 5 application server. Among the better-known Java Enterprise Edition application servers are WebLogic Server (BEA), JBoss (Red Hat), WebSphere (IBM), JRun (Adobe), Apache Geronimo (Apache Foundation, based on IBM WebSphere), Oracle OC4J (Oracle Corporation), Sun Java System Application Server (Sun Microsystems) and Glassfish Application Server (based on Sun Java System Application Server).

JOnAS application server was the first open-source application server to have achieved official compliance with the Java Enterprise Specification. BEA delivered the first Java EE 5 certified application server followed by Sun Microsystems' reference implementation GlassFish.

The Web modules are servlets and Java Server Pages, and business logic is built into Enterprise JavaBeans (EJB-3 and later). The Hibernate project offers an EJB-3 container implementation for the JBoss Application server. Tomcat from Apache and JOnAS from ObjectWeb are typical of containers into which these modules can be put.

A Java Server Page (JSP) is a servlet from Java that executes in a Web container—the Java equivalent of CGI scripts. JSPs are a way to create HTML pages by embedding references to the server logic within the page. HTML coders and Java programmers can work side by side by referencing each other's code from within their own. JavaBeans are the independent class components of the Java architecture from Sun Microsystems.

The application servers mentioned above mainly serve Web applications. Some application servers target networks other than the Web: Session Initiation Protocol servers, for instance, target telephony networks.

JBOSS

JBoss Application Server (or JBoss AS) is a free software / open-source Java EE-based application server. Because it is Java-based, JBoss AS is cross-platform, usable on any operating system that Java supports.

Environment

JBoss AS 4.0 is a J2EE 1.4 application server, with embedded Tomcat 5.5. Any JVM between 1.4 and 1.5 is supported. JBoss can run on numerous operating systems including Windows, Mac OS X, many POSIX platforms, and others, as long as a suitable JVM is present.

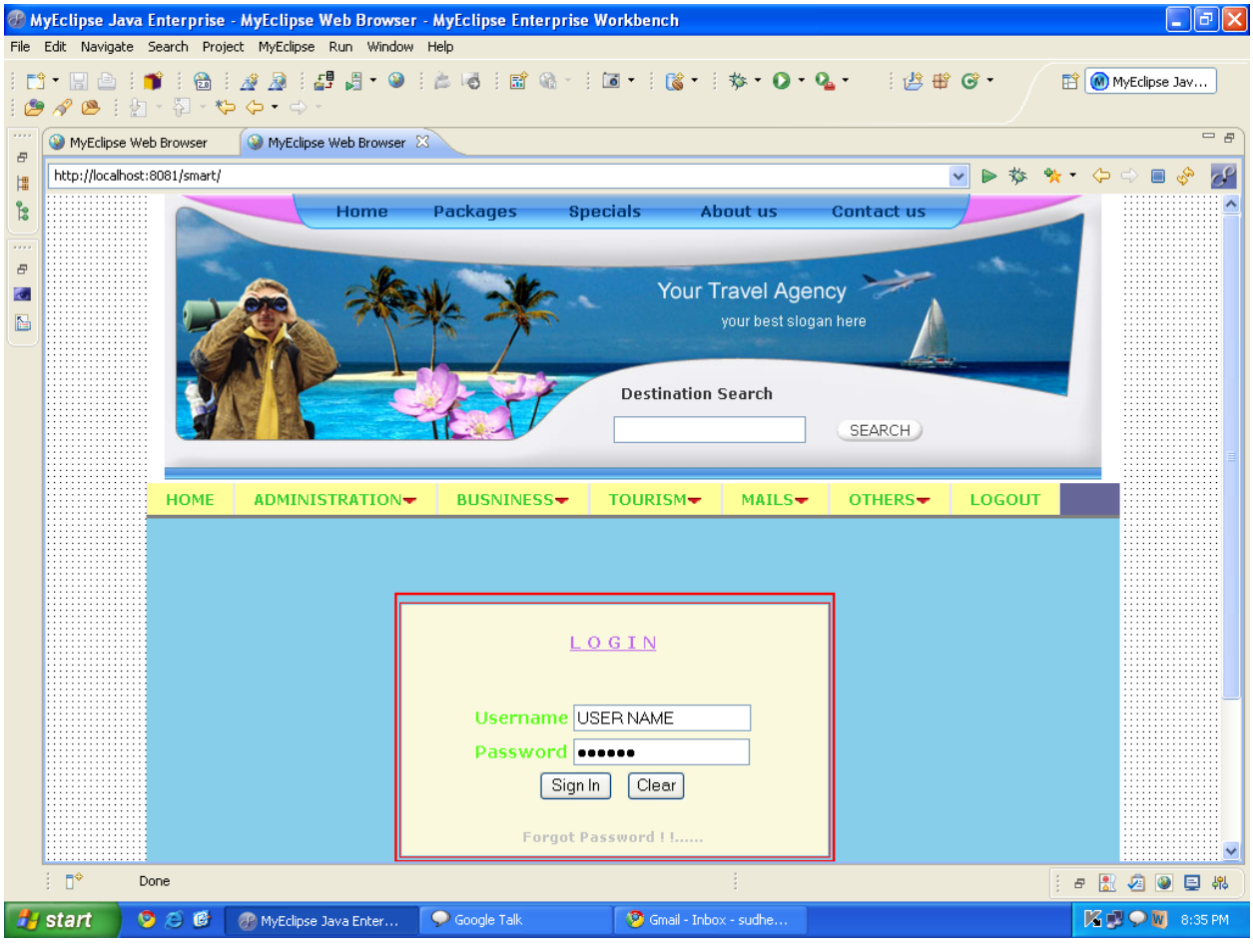
JBoss AS 4.2 is also a J2EE 1.4 application server, but EJB 3 is deployed by default. It requires JDK 5. Tomcat 6 is bundled with it.

Next JBoss AS 5 will be Java EE 5 application server.

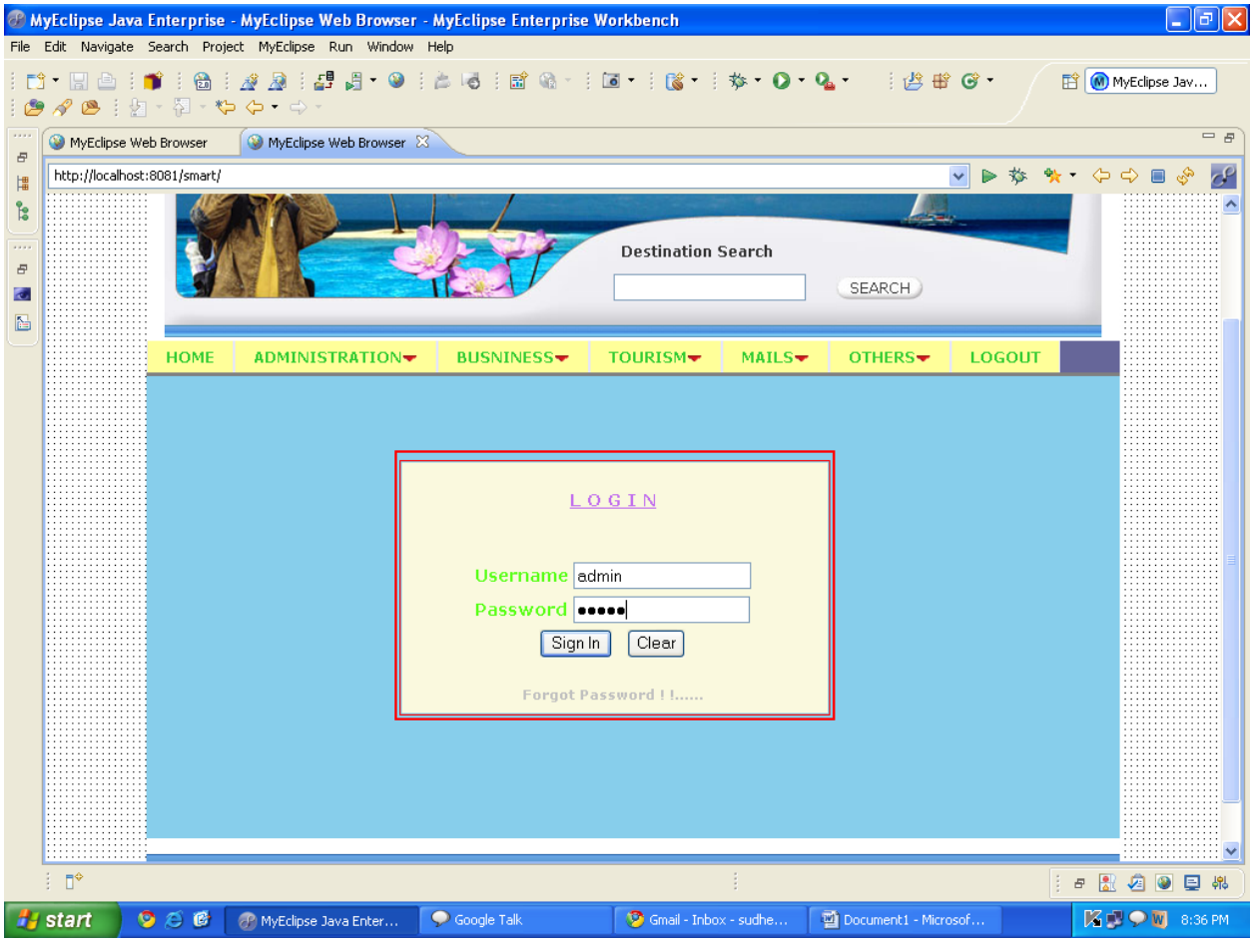
Product features

- ✓ Clustering
- ✓ Failover (including sessions)
- ✓ Load balancing
- ✓ Distributed caching (using JBoss Cache, a standalone product)
- ✓ Distributed deployment (farming)
- ✓ Enterprise JavaBeans version 3

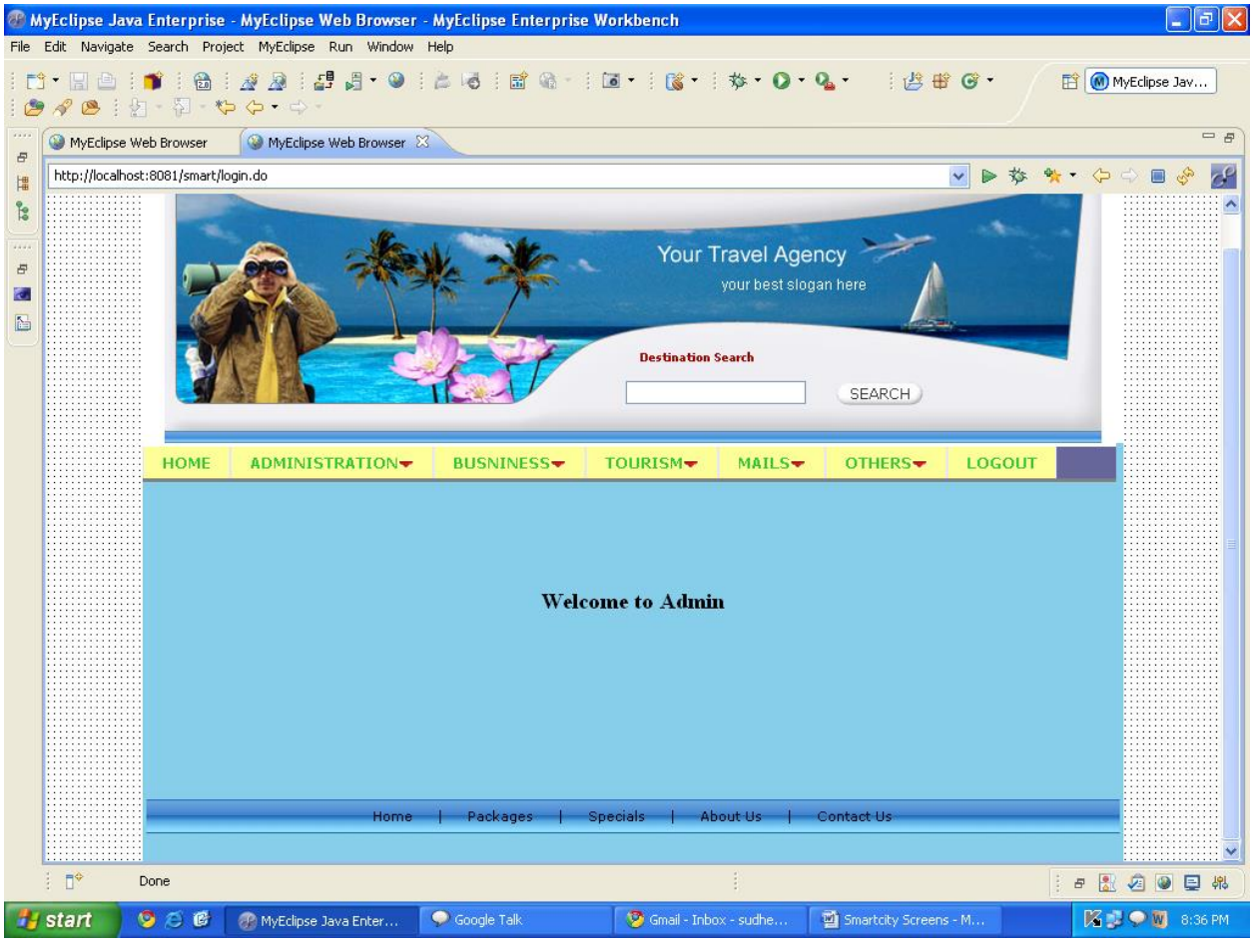
Snapshots


MyEclipse Java Enterprise - MyEclipse Web Browser - MyEclipse Enterprise Workbench
File Edit Navigate Search Project MyEclipse Run Window Help
MyEclipse Web Browser
MyEclipse Web Browser
http://localhost:8081/smart/
Home Packages Specials About us Contact us
Your Travel Agency
your best slogan here
Destination Search
SEARCH
HOME ADMINISTRATION BUSNINESS TOURISM MAILS OTHERS LOGOUT
LOGIN
Username USER NAME
Password
Sign In Clear
Forgot Password ! !.....
Done
start
MyEclipse Java Enter...
Google Talk
Gmail - Inbox - sudhe...
8:35 PM

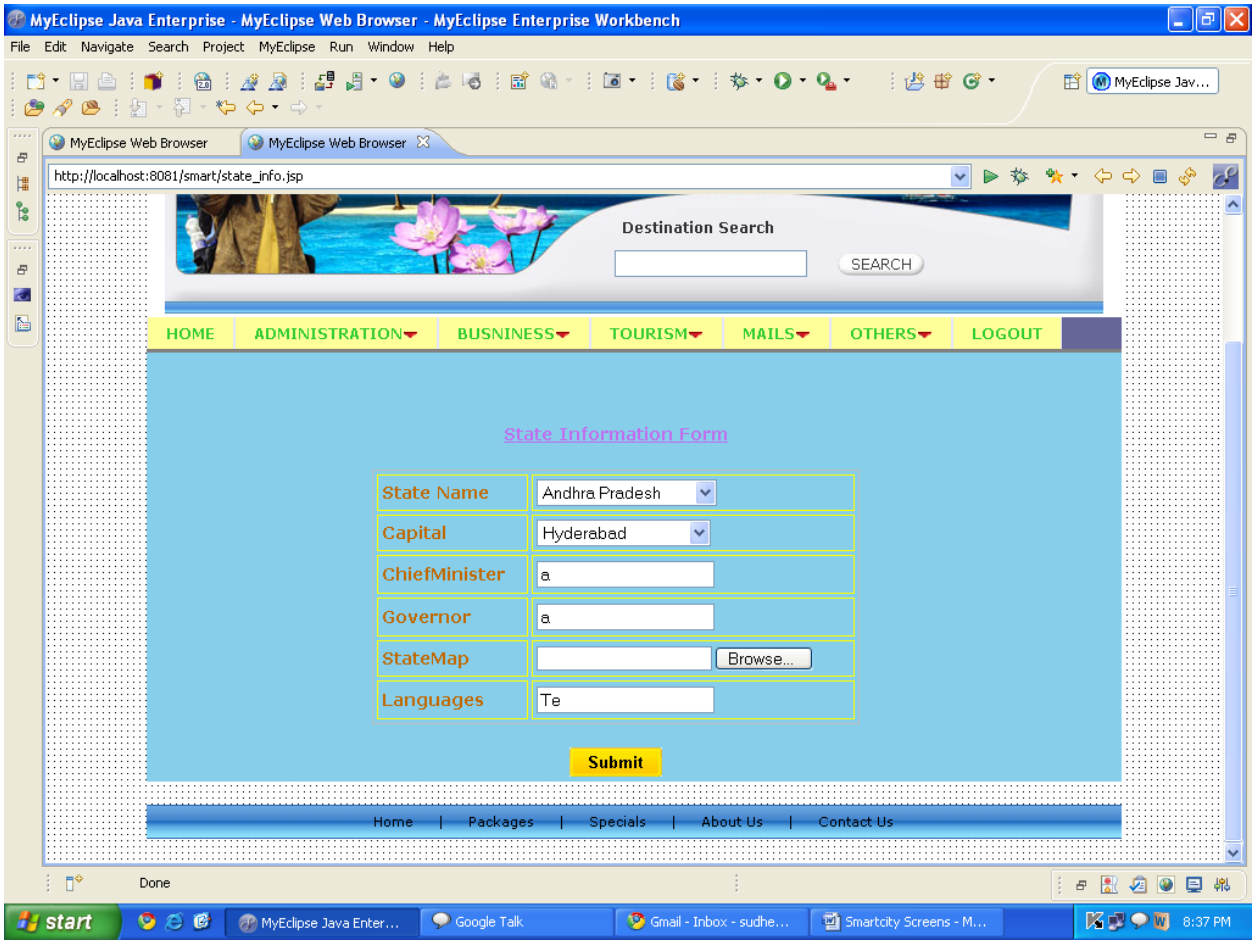
Admin Login:



Admin Home Page:


MyEclipse Java Enterprise - MyEclipse Web Browser - MyEclipse Enterprise Workbench
File Edit Navigate Search Project MyEclipse Run Window Help
MyEclipse Web Browser
MyEclipse Web Browser
http://localhost:8081/smart/login.do
Your Travel Agency
your best slogan here
Destination Search
SEARCH
HOME
ADMINISTRATION
BUSNINESS
TOURISM
MAILS
OTHERS
LOGOUT
Welcome to Admin
Home | Packages | Specials | About Us | Contact Us
Done
start
MyEclipse Java Enter...
Google Talk
Gmail - Inbox - sudhe...
Smartcity Screens - M...
8:36 PM

State Form:


MyEclipse Java Enterprise - MyEclipse Web Browser - MyEclipse Enterprise Workbench
File Edit Navigate Search Project MyEclipse Run Window Help
MyEclipse Web Browser
MyEclipse Web Browser
http://localhost:8081/smart/state_info.jsp
Destination Search
SEARCH
HOME
ADMINISTRATION
BUSNINESS
TOURISM
MAILS
OTHERS
LOGOUT
State Information Form
State Name
Andhra Pradesh
Capital
Hyderabad
ChiefMinister
a
Governor
a
StateMap
Browse...
Languages
Te
Submit
Home | Packages | Specials | About Us | Contact Us
Done
start
MyEclipse Java Enter...
Google Talk
Gmail - Inbox - sudhe...
Smartcity Screens - M...
8:37 PM

Testing

Software Testing is the process used to help identify the correctness, completeness, security, and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors. Quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; testing furnishes a criticism or comparison that compares the state and behavior of the product against a specification. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

There are many approaches to software testing, but effective testing of complex products is essentially a process of investigation, not merely a matter of creating and following routine procedure. One definition of testing is "the process of questioning a product in order to evaluate it", where the "questions" are operations the tester attempts to execute with the product, and the product answers with its behavior in reaction to the probing of the tester[citation needed]. Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product—putting the product through its paces. Some of the common quality attributes include capability, reliability, efficiency, portability, maintainability, compatibility and usability. A good test is sometimes described as one which reveals an error; however, more recent thinking suggests that a good test is one which reveals information of interest to someone who matters within the project community.

Introduction

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault

can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

Software testing may be viewed as a sub-field of Software Quality Assurance but typically exists independently (and there may be no SQA areas in some companies). In SQA, software process specialists and auditors take a broader view on software and its development. They examine and change the software engineering process itself to reduce the number of faults that end up in the code or deliver faster.

Regardless of the methods used or level of formality involved the desired result of testing is a level of confidence in the software so that the organization is confident that the software has an acceptable defect rate. What constitutes an acceptable defect rate depends on the nature of the software. An arcade video game designed to simulate flying an airplane would presumably have a much higher tolerance for defects than software used to control an actual airliner.

A problem with software testing is that the number of defects in a software product can be very large, and the number of configurations of the product larger still. Bugs that occur infrequently are difficult to find in testing. A rule of thumb is that a system that is expected to function without faults for a certain length of time must have already been tested for at least that length of time. This has severe consequences for projects to write long-lived reliable software.

A common practice of software testing is that it is performed by an independent group of testers after the functionality is developed but before it is shipped to the customer. This practice often results in the testing phase being used as project buffer to compensate for project delays. Another practice is to start software testing at the same moment the project starts and it is a continuous process until the project finishes.

Another common practice is for test suites to be developed during technical support escalation procedures. Such tests are then maintained in regression testing suites to ensure that future updates to the software don't repeat any of the known mistakes.

It is commonly believed that the earlier a defect is found the cheaper it is to fix it.

	Time Detected				
Time Introduced	Requirements	Architecture	Construction	System Test	Post-Release
Requirements	1	3	5-10	10	10-100
Architecture	-	1	10	15	25-100
Construction	-	-	1	10	10-25

In counterpoint, some emerging software disciplines such as extreme programming and the agile software development movement, adhere to a "test-driven software development" model. In this process unit tests are written first, by the programmers (often with pair programming in the extreme programming methodology). Of course, these tests fail initially; as they are expected to. Then as code is written it passes incrementally larger portions of the test suites. The test suites are continuously updated as new failure conditions and corner cases are discovered, and they are integrated with any regression tests that are developed.

Unit tests are maintained along with the rest of the software source code and generally integrated into the build process (with inherently interactive tests being relegated to a partially manual build acceptance process).

The software, tools, samples of data input and output, and configurations are all referred to collectively as a test harness.

History

The separation of debugging from testing was initially introduced by Glenford J. Myers in his 1978 book the "Art of Software Testing". Although his attention was on breakage testing it illustrated the desire of the software engineering community to separate fundamental development activities, such as debugging, from that of verification. Drs. Dave Gelperin and William C. Hetzel classified in 1988 the phases and goals in software testing as follows: until 1956 it was the debugging-oriented period, where testing was often associated to debugging: there was no clear difference between testing and debugging. From 1957-1978 there was the demonstration-oriented period where debugging and testing was distinguished now - in this period it was shown, that software satisfies the requirements. The time between 1979-1982 is announced as the destruction-oriented period, where the goal was to find errors. 1983-1987 is classified as the evaluation-oriented period: intention here is that during the software lifecycle a product evaluation is provided and measuring quality. From 1988 on it was seen as prevention-oriented period where tests were to demonstrate that software satisfies its specification, to detect faults and to prevent faults. Dr. Gelperin chaired the IEEE 829-1988 (Test Documentation Standard) with Dr. Hetzel writing the book "The Complete Guide of Software Testing". Both works were pivotal in to today's testing culture and remain a consistent source of reference. Dr. Gelperin and Jerry E. Durant also went on to develop High Impact Inspection Technology that builds upon traditional Inspections but utilizes a test-driven additive.

White-box and black-box testing

To meet Wikipedia's quality standards, this section may require cleanup. Please discuss this issue on the talk page, and/or replace this tag with a more specific message.

White box and black box testing are terms used to describe the point of view a test engineer takes when designing test cases. Black box being an external view of the test object and white box being an internal view. Software testing is partly intuitive, but largely systematic. Good testing involves much more than

just running the program a few times to see whether it works. Thorough analysis of the program under test, backed by a broad knowledge of testing techniques and tools are prerequisites to systematic testing. Software Testing is the process of executing software in a controlled manner; in order to answer the question "Does this software behave as specified?" Software testing is used in association with Verification and Validation. Verification is the checking of or testing of items, including software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques as reviews, inspections, walk-through. Validation is the process of checking what has been specified is what the user actually wanted.

- Validation: Are we doing the right job?
- Verification: Are we doing the job right?

In order to achieve consistency in the Testing style, it is imperative to have and follow a set of testing principles. This enhances the efficiency of testing within SQA team members and thus contributes to increased productivity. The purpose of this document is to provide overview of the testing, plus the techniques.

At SDEI, 3 levels of software testing are done at various SDLC phases

- Unit Testing: in which each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented
- Integration testing: in which progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a whole.
- System testing: in which the software is integrated to the overall product and tested to show that all requirements are met

A further level of testing is also done, in accordance with requirements:

- Acceptance testing: upon which the acceptance of the complete software is based. The clients often do this.

- Regression testing: is used to refer the repetition of the earlier successful tests to ensure that changes made in the software have not introduced new bugs/side effects.

In recent years the term grey box testing has come into common usage. The typical grey box tester is permitted to set up or manipulate the testing environment, like seeding a database, and can view the state of the product after his actions, like performing a SQL query on the database to be certain of the values of columns. It is used almost exclusively of client-server testers or others who use a database as a repository of information, but can also apply to a tester who has to manipulate XML files (DTD or an actual XML file) or configuration files directly. It can also be used of testers who know the internal workings or algorithm of the software under test and can write tests specifically for the anticipated results. For example, testing a data warehouse implementation involves loading the target database with information, and verifying the correctness of data population and loading of data into the correct tables.

Test levels

- Unit testing tests the minimal software component and sub-component or modules by the programmers.
- Integration testing exposes defects in the interfaces and interaction between integrated components (modules).
- Functional testing tests the product according to programmable work.
- System testing tests an integrated system to verify/validate that it meets its requirements.
- Acceptance testing testing can be conducted by the client. It allows the end-user or customer or client to decide whether or not to accept the product. Acceptance testing may be performed after the testing and before the implementation phase. See also Development stage
    - Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.
    - Beta testing comes after alpha testing. Versions of the software, known as beta versions, are released to a limited audience outside of the company. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

It should be noted that although both Alpha and Beta are referred to as testing it is in fact use emersion. The rigors that are applied are often unsystematic and many of the basic tenets of testing process are not used. The Alpha and Beta period provides insight into environmental and utilization conditions that can impact the software.

After modifying software, either for a change in functionality or to fix defects, a regression test re-runs previously passing tests on the modified software to ensure that the modifications haven't unintentionally caused a regression of

previous functionality. Regression testing can be performed at any or all of the above test levels. These regression tests are often automated.

Test cases, suites, scripts and scenarios

A test case is a software testing document, which consists of event, action, input, output, expected result and actual result. Clinically defined (IEEE 829-1998) a test case is an input and an expected result. This can be as pragmatic as 'for condition x your derived result is y', whereas other test cases described in more detail the input scenario and what results might be expected. It can occasionally be a series of steps (but often steps are contained in a separate test procedure that can be exercised against multiple test cases, as a matter of economy) but with one expected result or expected outcome. The optional fields are a test case ID, test step or order of execution number, related requirement(s), depth, test category, author, and check boxes for whether the test is automatable and has been automated. Larger test cases may also contain prerequisite states or steps, and descriptions. A test case should also contain a place for the actual result. These steps can be stored in a word processor document, spreadsheet, database or other common repository. In a database system, you may also be able to see past test results and who generated the results and the system configuration used to generate those results. These past results would usually be stored in a separate table.

The term test script is the combination of a test case, test procedure and test data. Initially the term was derived from the byproduct of work created by automated regression test tools. Today, test scripts can be manual, automated or a combination of both.

The most common term for a collection of test cases is a test suite. The test suite often also contains more detailed instructions or goals for each collection of test cases. It definitely contains a section where the tester identifies the system configuration used during testing. A group of test cases may also contain prerequisite states or steps, and descriptions of the following tests.

Collections of test cases are sometimes incorrectly termed a test plan. They might correctly be called a test specification. If sequence is specified, it can be called a test script, scenario or procedure.

A sample testing cycle

Although testing varies between organizations, there is a cycle to testing:

1. Requirements Analysis: Testing should begin in the requirements phase of the software development life cycle.

   During the design phase, testers work with developers in determining what aspects of a design are testable and under what parameter those tests work.

2. Test Planning: Test Strategy, Test Plan(s), Test Bed creation.
3. Test Development: Test Procedures, Test Scenarios, Test Cases, Test Scripts to use in testing software.
4. Test Execution: Testers execute the software based on the plans and tests and report any errors found to the development team.
5. Test Reporting: Once testing is completed, testers generate metrics and make final reports on their test effort and whether or not the software tested is ready for release.
6. Retesting the Defects

Not all errors or defects reported must be fixed by a software development team. Some may be caused by errors in configuring the test software to match the development or production environment. Some defects can be handled by a workaround in the production environment. Others might be deferred to future releases of the software, or the deficiency might be accepted by the business user. There are yet other defects that may be rejected by the development team (of course, with due reason) if they deem it inappropriate to be called a defect.

Limitations and Scope for Future Enhancements

Limitations of the system:

- Only the permanent employees can access the system.
- System works in all platforms and its compatible environments.
- Advanced techniques are not used to check the authorization.

Future Enhancements:

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

- As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment.
- Because it is based on object-oriented design, any further changes can be easily adaptable.
- Based on the future security issues, security can be improved using emerging technologies.
- Attendance module can be added
- sub admin module can be added

Project Summary

The **SMART CITY** is a web-based application for primarily providing training to the employees who provide customized solutions to meet organizational needs.

This application software has been computed successfully and was also tested successfully by taking “test cases”. It is user friendly, and has required options, which can be utilized by the user to perform the desired operations.

The software is developed using Java as front end and Oracle as back end in Windows environment. The goals that are achieved by the software are:

- ✓ Instant access.
- ✓ Improved productivity.
- ✓ Optimum utilization of resources.
- ✓ Efficient management of records.
- ✓ Simplification of the operations.
- ✓ Less processing time and getting required information.
- ✓ User friendly.
- ✓ Portable and flexible for further enhancement.

References

Core Java™ 2 Volume I – Fundamentals 7th Edition
Pearson Education – Sun Microsystems
- Cay S. Hortsman
Gary Cornell

Core Java™ 2 Volume II – Advanced
Pearson Education – Sun Microsystems
- Cay S. Hortsman
Gary Cornell

Head First Servlets & JSP
O'Reilly – SPD
- Eric Freeman
Elisabeth Freeman

The Book of JavaScript 2nd Edition
SPD
- thau

Effective Java – Programming Language Guide
Pearson Education – Sun Microsystems
- Joshua Bloch

Java Database Best Practices
O'Reilly – SPD
- George Reese

JBoss – A Developers Notebook
O'Reilly – SPD
- Norman Richards
Sam Griffith