**A PROJECT REPORT ON**

XXXXXXXXXX

**Submitted to Osmania University for the partial fulfillment of the requirement for the**

**Award of Degree for**

XXXXXXXXXXXXXXXXXXXXXXX

**Done by**

Mr. /Miss. XXXXXX

**XXXXX** Institute of Management & Computer Sciences
**Hyderabad**

## CERTIFICATE

This is to certify that Mr. XXXX, bearing Roll No. XXXXXXXXXXX have developed Software project titled XXXXXXXX for **xxx SOFTWARE SOLUTIONS** as a partial Fulfillment for the award of the Degree of XXXXXXX.

**HEAD OF DEPARTMENT**

**PRINCIPAL**
**XXX institute of Management & Computer Sciences**

**EXTERNAL**

# ACKNOWLEDGMENT

My express thanks and gratitude and thanks to Almighty God, my parents and other family members and friends without whose uncontained support, I could not have made this career in XXXX.

I wish to place on my record my deep sense of gratitude to my project guide, **Mr. XXXXX, xxx Software Solutions, Hyderabad** for his constant motivation and valuable help through the project work. Express my gratitude to **Mr. XXXX**, Director of XXXXX **Institute of Management & Computer Sciences** for his valuable suggestions and advices through out the XXX course. I also extend my thanks to other Faculties for their Cooperation during my Course.

Finally I would like to thank my friends for their cooperation to complete this project.

XXXXXXX

# ABSTRACT

An online comprehensive Customer Care Solution is to manage customer interaction and complaints with the Service Providers over phone or through and e-mail. The system should have capability to integrate with any Service Provider from any domain or industry like Banking, Telecom, Insurance, etc.

Customer Service also known as Client Service is the provision of service to customers its significance varies by product, industry and domain. In many cases customer services is more important if the purchase relates to a service as opposed to a product.

Customer Service may be provided by a Person or Sales & Service Representatives Customer Service is normally an integral part of a company's customer value proposition.

# CONTENTS

# CHAPTER 1

## INTRODUCTION

## 1.1. INTRODUCTION TO PROJECT

The Customer Service Desk is a web based project.

Customer Service also known as Client Service is the provision of service to customers' .Its significance varies by product, industry and domain. In many cases customer services is more important if the information relates to a service as opposed to a Customer.

Customer Service may be provided by a Service Representatives Customer Service is normally an integral part of a company's customer value proposition.

## 1.2. ORGANIZATION PROFILE

## SOFTWARE SOLUTIONS

xxx Software Solutions is an IT solution provider for a dynamic environment where business and technology strategies converge. Their approach focuses on new ways of business combining IT innovation and adoption while also leveraging an organization's current IT assets. Their work with large global corporations and new products or services and to implement prudent business and technology strategies in today's environment.

## XXX'S RANGE OF EXPERTISE INCLUDES:

- Software Development Services
- Engineering Services
- Systems Integration
- Customer Relationship Management
- Product Development
- Electronic Commerce
- Consulting

- IT Outsourcing

We apply technology with innovation and responsibility to achieve two broad objectives:

- Effectively address the business issues our customers face today.

## THIS APPROACH RESTS ON:

- A strategy where we architect, integrate and manage technology services and solutions - we call it AIM for success.
- A robust offshore development methodology and reduced demand on customer resources.
- A focus on the use of reusable frameworks to provide cost and times benefits.

They combine the best people, processes and technology to achieve excellent results - consistency. We offer customers the advantages of:

## SPEED:

They understand the importance of timing, of getting there before the competition. A rich portfolio of reusable, modular frameworks helps jump-start projects. Tried and tested methodology ensures that we follow a predictable, low - risk path to achieve results. Our track record is testimony to complex projects delivered within and evens before schedule.

## EXPERTISE:

Our teams combine cutting edge technology skills with rich domain expertise. What's equally important - they share a strong customer orientation that means they actually start by listening to the customer. They're focused on coming up with solutions that serve customer requirements today and anticipate future needs.

## A FULL SERVICE PORTFOLIO:

They offer customers the advantage of being able to Architect, integrate and manage technology services. This means that they can rely on one, fully accountable source instead of trying to integrate disparate multi vendor solutions.

## SERVICES:

Xxx is providing it's services to companies which are in the field of production, quality control etc With their rich expertise and experience and information technology they are in best position to provide software solutions to distinct business requirements.

## 1.3. PURPOSE OF THE PROJECT

An online comprehensive Customer Care Solution is to manage customer interaction and complaints with the Service Providers over phone or through and e-mail. The system should have capability to integrate with any Service Provider from any domain or industry like Banking, Telecom, Insurance, etc.

Customer Service also known as Client Service is the provision of service to customers Its significance varies by product, industry and domain. In many cases customer services is more important if the information relates to a service as opposed to a Customer.

Customer Service may be provided by a Service Representatives Customer Service is normally an integral part of a company's customer value proposition.

# CHAPTER 2

## 2.1.PROBLEM IN EXISTING SYSTEM

The existing system is a semi-automated at where the information is stored in the form of excel sheets in disk drives. The information sharing to the Volunteers, Group members, etc. is through mailing feature only. The information storage and maintenance is more critical in this system. Tracking the member's activities and progress of the work is a tedious job here. This system cannot provide the information sharing by 24x7 days.

## 2.2.SOLUTION OF THESE PROBLEMS

The development of this new system objective is to provide the solution to the problems of existing system. By using this new system, we can fully automate the entire process of the current system. The new system would like to make as web-enabled so that the information can be shared between the members at any time using the respective credentials. To track the status of an individual process, the status update can be centralized using the new system. Being a web-enabled system, the process can be accessed across the world over net.

This system also providing the features like Chatting, Mailing between the members; Images Upload – Download via the web site; updating the process status in centralized location; generated reports can also be exporting to the applications like MS-Excel, PDF format, etc. In this new system, the members like Donors can give their valuable feedback to the Volunteers so that the Volunteers can check their progress of the tasks.

The entire process categorized as different modules like Admin module, Volunteer module, etc. at where we can classify the functionality as an individual process.
Using the new system entering into Admin module we can perform….
In this new system using the Volunteer module we can do….
In the Reports module we can generate reports like Weekly Status Report,

# CHAPTER 3

# 3. Feasibility Report

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operation Feasibility
- Economical Feasibility

## 3.1. Technical Feasibility

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipments have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it

provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

## 3.2. Operational Feasibility

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

## 3.3. Economic Feasibility

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

# CHAPTER 4

## 4.1 SOFTWARE REQUIREMENT SPECIFICATION

### Overview:

An online comprehensive Customer Care Solution is to manage customer interaction and complaints with the Service Providers over phone or through and e-mail. The system should have capability to integrate with any Service Provider from any domain or industry like Banking, Telecom, Insurance, etc.

Customer Service also known as Client Service is the provision of service to customers Its significance varies by product, industry and domain.

## Modules:

### 1.Admin:

Administrator or Manger is treated as a super user in this system. He can have all the privileges in this system. He can track all customers with voice recording facility.

He can track the customer complaints ticket create, edit facilities by customer or service providers. Admin facilities to add, view, edit different types of interactions and complaints.

### Provisions:

### Login:

- ✔ LoginId
- ✔ Password
- ✔ Emp_Customer_Id
- ✔ Role

### Department:

- DeptId
- DeptName

- Abbreviation
- DeptinchargeId
- Description

## Designation:

- DesgId
- DesgName
- Abbreviation
- DesgInchargeId
- superiorDesgId
- Description

## Domain:

- DomainId
- DomainName
- DomainAbbreviation
- Domainesc
- DomainInchargeId
- DeptId

## Complaint:

- serviceCutomerDomainId
- ComplaintTypeId

## ComplaintType:

- ✔ ComplaintTypeId
- ✔ ComplaintTypeName
- ✔ ComplaintTypeDesc
- ✔ ComplaintTypeDesc
- ✔ InchargeId

## Report:

- ReportId

- ReportDate
- RepotrTime
- ReportGeneratedEmpId

## Functionality:

- Association of domain and Employee.
- Association of ComplaintType and Employee.
- Searching of all domain categories.
- Add the new Designation Records.
- Adding Departments.
- Modify the designation Records.

## Queries:

- How many Employees  are registered on a day?
- How many customers are registered on a particular date?
- How many complaints are posted on a day?
- How many departments are added on aday?
- How many Designations are addedv on aday?

## Alerts:

- ✔ Please select the departmentType.
- ✔ Enter valid Id & Password.
- ✔ Please enter Domain Incharge.
- ✔ All fields are mandatory.

## Reports:

- To generate the reports of all customers.
- To generate the reports of all departments.
- To generate the reports of all Survice Providers.
- To generate the reports of all complaints received on a day.

## 2. Customers:

Customer is also a client; buyer or purchaser is the buyer or user of the paid products of an individual or an organization, mostly called the supplier or seller. This is typically through purchasing or renting goods or services.

A group of services are provided to customer like banking, telecom, railway etc. Customer need to select help from any of these sectors.

A customer complaint with voice is recorded in this system, and process the complaint within a certain amount of time.

### Provisions:

### Customer:

- ✔ CustomerId
- ✔ CustomerName
- ✔ DOB
- ✔ EmailId
- ✔ PhoneNumber
- ✔ CustDesc
- ✔ CustDOR
- ✔ Address

### CustComplaintReg:

- ✔ ComplaintRegId
- ✔ ComplaintRegDate
- ✔ RegistrationTime
- ✔ CustomerId
- ✔ ServiceCustDomain
- ✔ PhoneNum
- ✔ CustComplaintDomain
- ✔ TextOfComplaint

- ✔ EmpId
- ✔ VoiceTextOfCompaint
- ✔ TextFile
- ✔ ComplaintStatus
- ✔ ComplaintEscalatedStatus
- ✔ CountOfEscalation
- ✔ ComplaintResponseText
- ✔ VoiceFileName
- ✔ Complaintsevearity

## Custfeedback:

- FeedBackId
- CustId
- Date
- Text
- EmailId
- FeedbackVoiceFile
- FileName
- ComplaintRegId

## Functionality:

- ✔ Association of Customer and FeedbackDetails.
- ✔ Association of CustComplaintReg and Employee.
- ✔ Association of feedback evaluation and feedback.
- ✔ Association of Customer feedback and customer.
- ✔ Viewing Customer Details.
- ✔ Sending feedback to the ServiceCustomers.

## Queries:

How many complaints are posted on a day?

- How many Customers are registered on a day?

- What is the complaintType which was posted by a customer?

### Alerts:

- All fields are mandatory.
- Please enter valid Id & password.
- Please select the mode of operation.

### Reports:

- To generate the reports of all types of domain.
- To generate the reports of all complaints posted by a customer.

## 3.Service Providers:

### Provisions:

### Employee:

- EmployeeId
- EmpName
- EmpDOB
- EmpDOJ
- Address
- Email
- PhoneNumber
- DeptId
- DesgId
- EmpPhoto
- EmpFileName

### ServiceCustDomain:

- ServiceCustDomainId
- ServiceCustId
- ServiceCustDomainMasterId
- ServiceCustDomainPhoneNum

- ServiceCustDomainInchargeId
- ServiceCustDomainManagerId
- ServiceCustDomainEmailId
- ServiceCustDomainAddress
- ServiceCustDomainDesc

### ServiceCustDomainComplaintTable:

- ✔ ServiceCustDomainId
- ✔ ComplaintTypeId

### ServiceCustDomainCustomers:

- ✔ ServiceCustDomainId
- ✔ CustomerId

### ServiceCustDomain Incharge:

- ✔ ServiceCustDomainId
- ✔ ServiceCustDomainInchargeId
- ✔ EmpId

### ServiceCustDomainPhone:

- ServiceCustDomainId
- PhoneNum1
- PhoneNum2
- PhoneNum3

### ServiceCustomers:

- ✔ ServiceCustId
- ✔ ServiceCustName
- ✔ ServiceCustDOR
- ✔ CustAddress
- ✔ CustPhoneNum
- ✔ CustEmail
- ✔ serviceCustInchargeId

- ✔ ServiceCustDesc

## Functionality:

- ✔ Association of Employerr and Service Customers.
- ✔ Association of ServiceCustDomain and ServiceCust.
- ✔ Association of ServiceCustDomainCust and ServiceCust.
- ✔ Association of ServiceCustDomainComplaint and complaintType.
- ✔ Association of Emp and Domain.
- ✔ Viewing Mail Details.
- ✔ Viewing Complaint Details.
- ✔ Compose Emails.

## Queries:

- How many Complaints are received on a day?
- How many Emails are received on a day.
- How many Mails are send to a particular EmailId on a day.

## Alerts:

- All fields are mandatory.
- Please select the mode of operation.
- Enter valid EmailId.
- Please attatch file.

## Reports:

- To generate the reports of all Complaints received on a day.
- To generate the reports of all Mails sent on a day.
- To generate the reports of all mails in inbox.

## Features Of This Project:

### 1.Search Engine:

It is a tool used to provide the search option to the job seekers like based on the functional area and location. If the job seekers selects any location it shows list of all available jobs on that place.

### 2.Job Calender:

If the user selects any date in the job calendar then it displays list of jobs available on that particular date in the same page. This feature completely developed by implementing Ajax features.

### 4.1.HARDWARE REQUIREMENTS:

- ✔ P4 2.8GB processor and above.
- ✔ Ram 512 MB and above.
- ✔ HDD 20 GB Hard Disk and above.

### 4.2. SOFTWARE REQUIREMENTS:

- Microsoft .Net framework 2.0.
- Microsoft ASP.Net.
- AJAX Tool kit.
- Microsoft C#.Net language.
- Microsoft SQL Server 2005
- HTML.

# CHAPTER 5

# SYSTEM DESIGN

## 5.1. Module design:

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

## 5.2. DATA FLOW DIAGRAMS

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

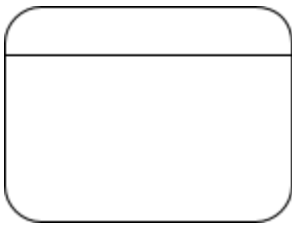
A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

## DFD SYMBOLS:

In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.

4. An open rectangle is a data store, data at rest or a temporary repository of data



Process that transforms data flow.

Source or Destination of data



Data flow

Data Store

## CONSTRUCTING A DFD:

Several rules of thumb are used in drawing DFD'S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

## SAILENT FEATURES OF DFD'S

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

## TYPES OF DATA FLOW DIAGRAMS

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

## CURRENT PHYSICAL:

In Current Physical DFD proecess label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

## CURRENT LOGICAL:

The physical aspects at the system are removed as mush as possible so that the current system is reduced to its essence to the data and the processors that transform them regardless of actual physical form.

## NEW LOGICAL:

This is exactly like a current logical model if the user were completely happy with he user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

## NEW PHYSICAL:

The new physical represents only the physical implementation of the new system.

## RULES GOVERNING THE DFD'S

## PROCESS

1) No process can have only outputs.
2) No process can have only inputs. If an object has only inputs than it must be a sink.
3) A process has a verb phrase label.

## DATA STORE

1) Data cannot move directly from one data store to another data store, a process must move data.
2) Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
3) A data store has a noun phrase label.

## SOURCE OR SINK

The origin and /or destination of data.

1) Data cannot move direly from a source to sink it must be moved by a process
2) A source and /or sink has a noun phrase land

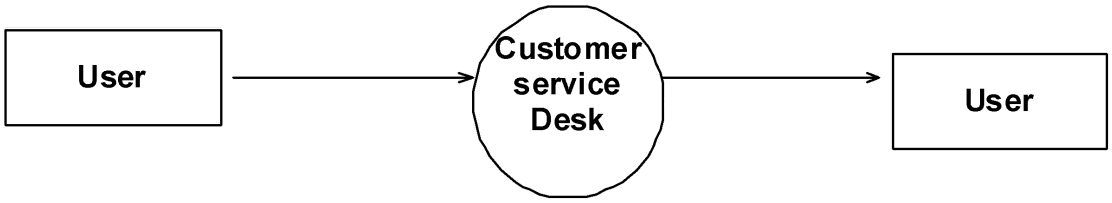
## DATA FLOW

1) A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The later is usually indicated however by two separate arrows since these happen at different type.
2) A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
3) A data flow cannot go directly back to the same process it leads. There must be atleast one other process that handles the data flow produce some other data flow returns the original data into the beginning process.

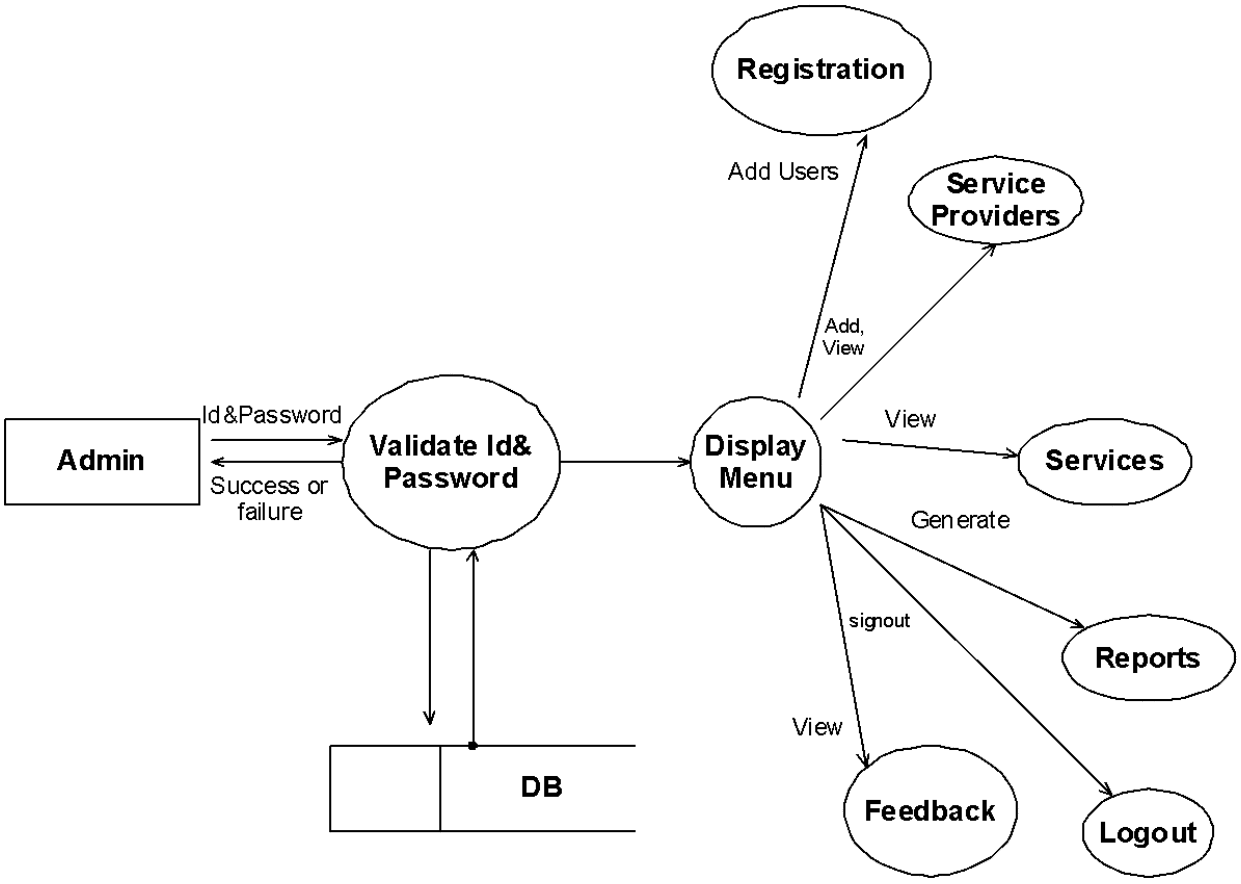
4) A Data flow to a data store means update (delete or change).
5) A data Flow from a data store means retrieve or use.

A data flow has a noun phrase label more than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

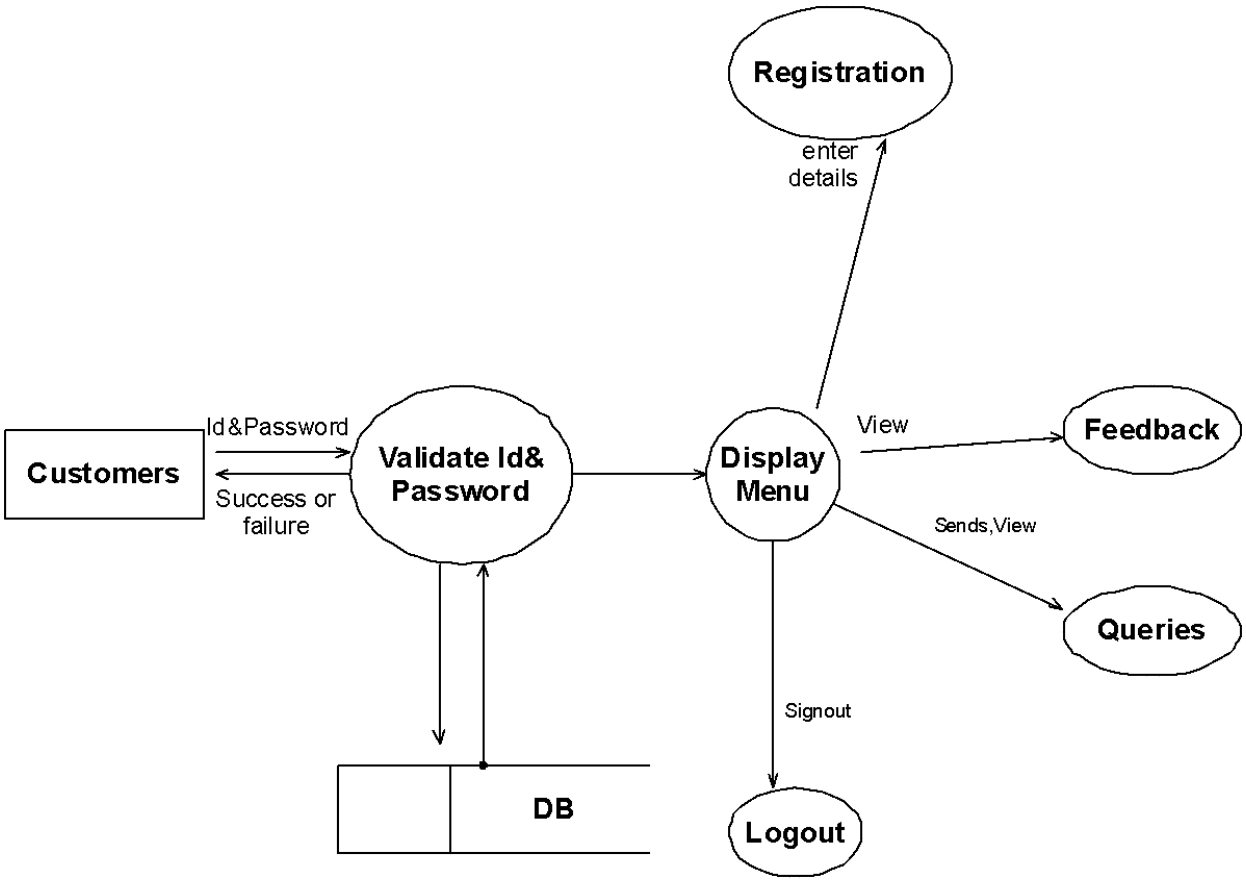
## Level 0 Diagram :



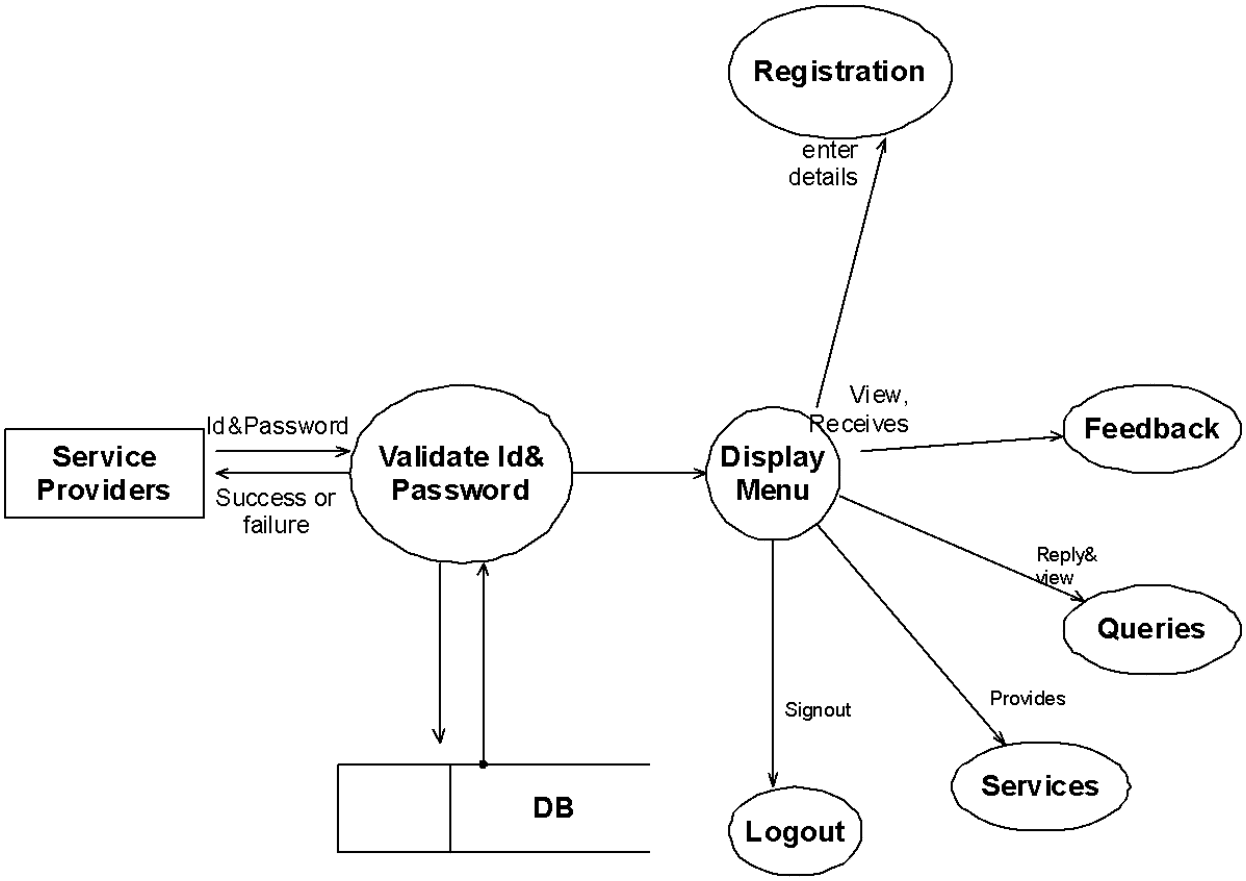
**Level 1 Diagram for Admin:**



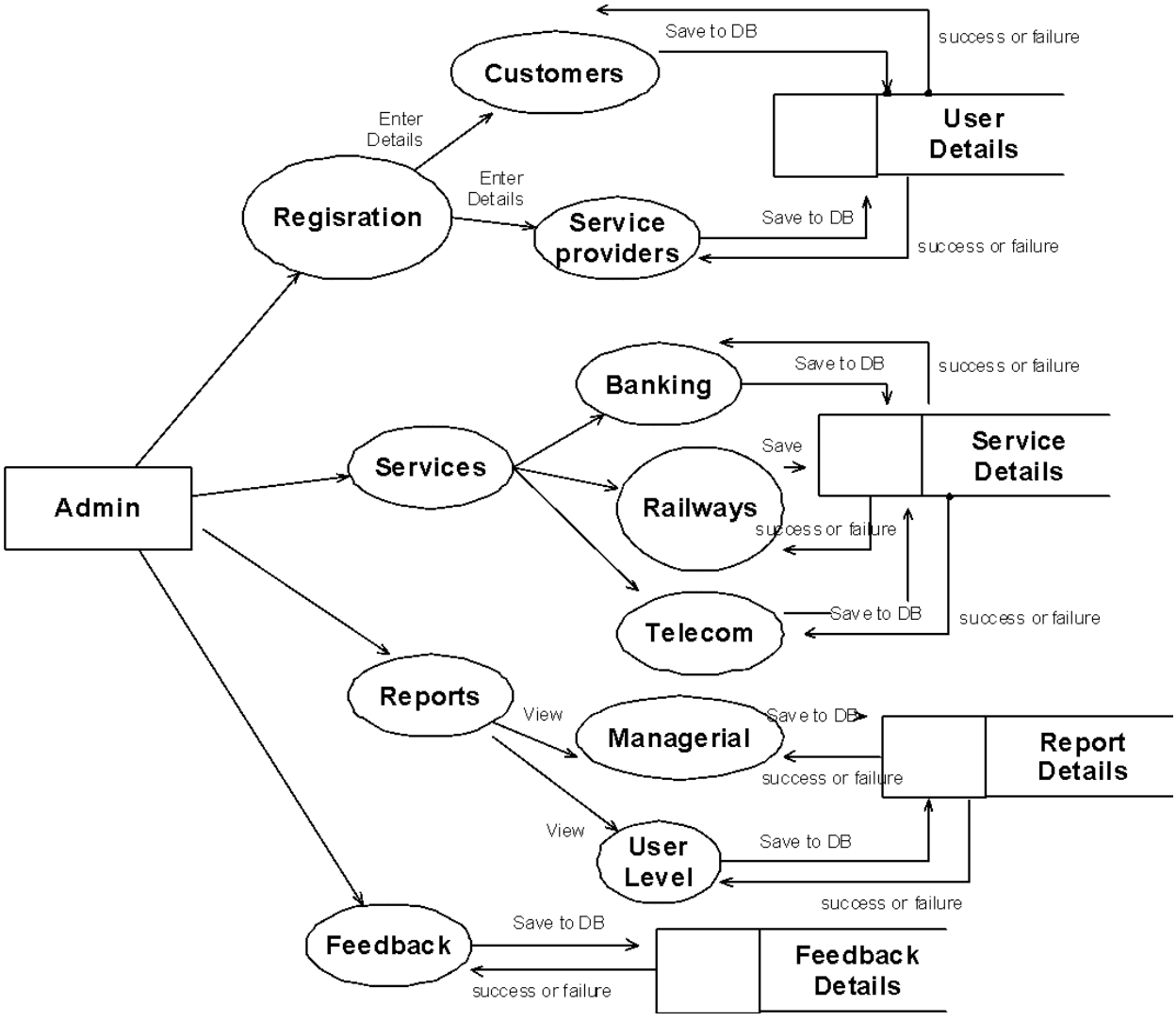
**Level 1 Diagram for Customers:**



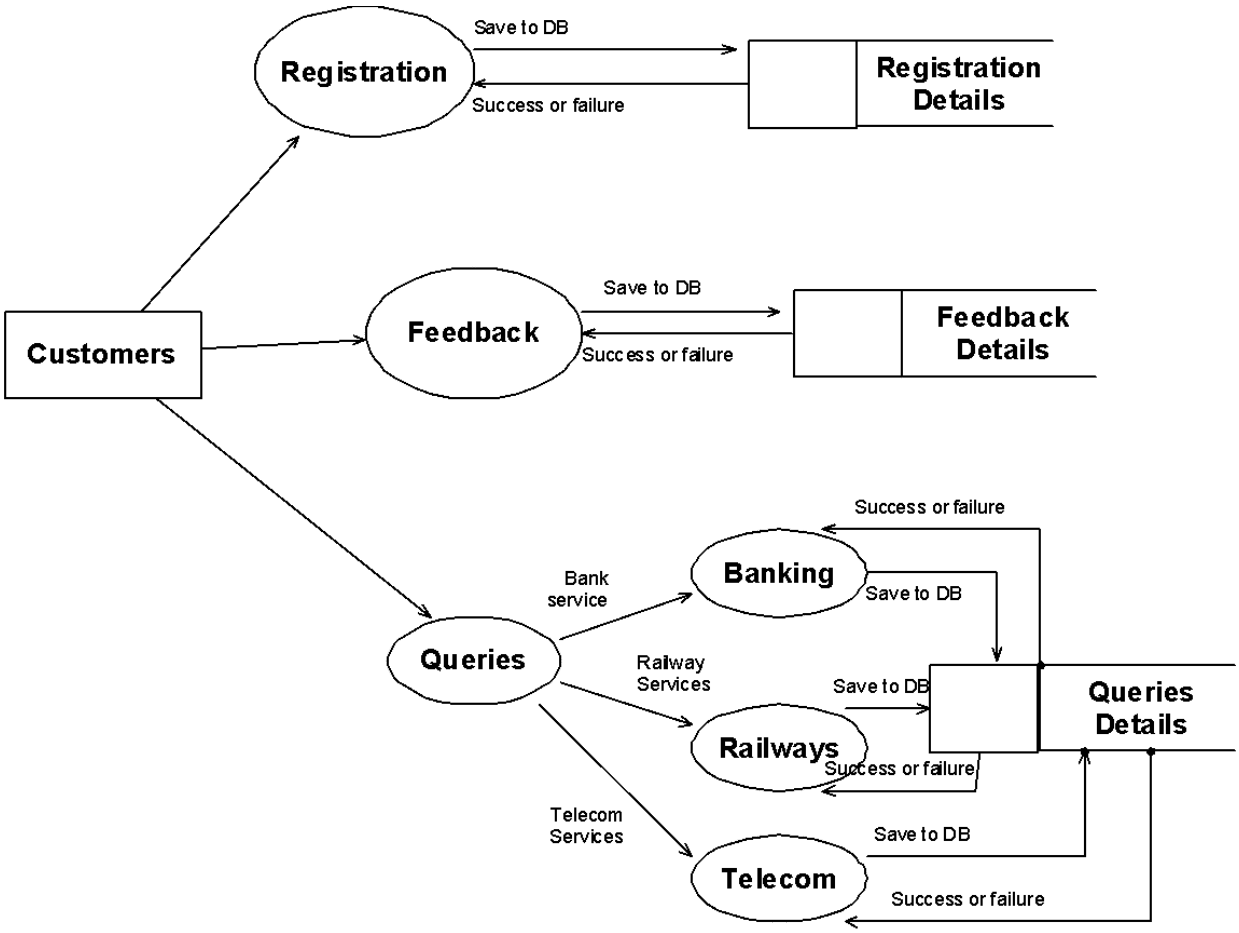
**Level 1 Diagram for service Providers:**



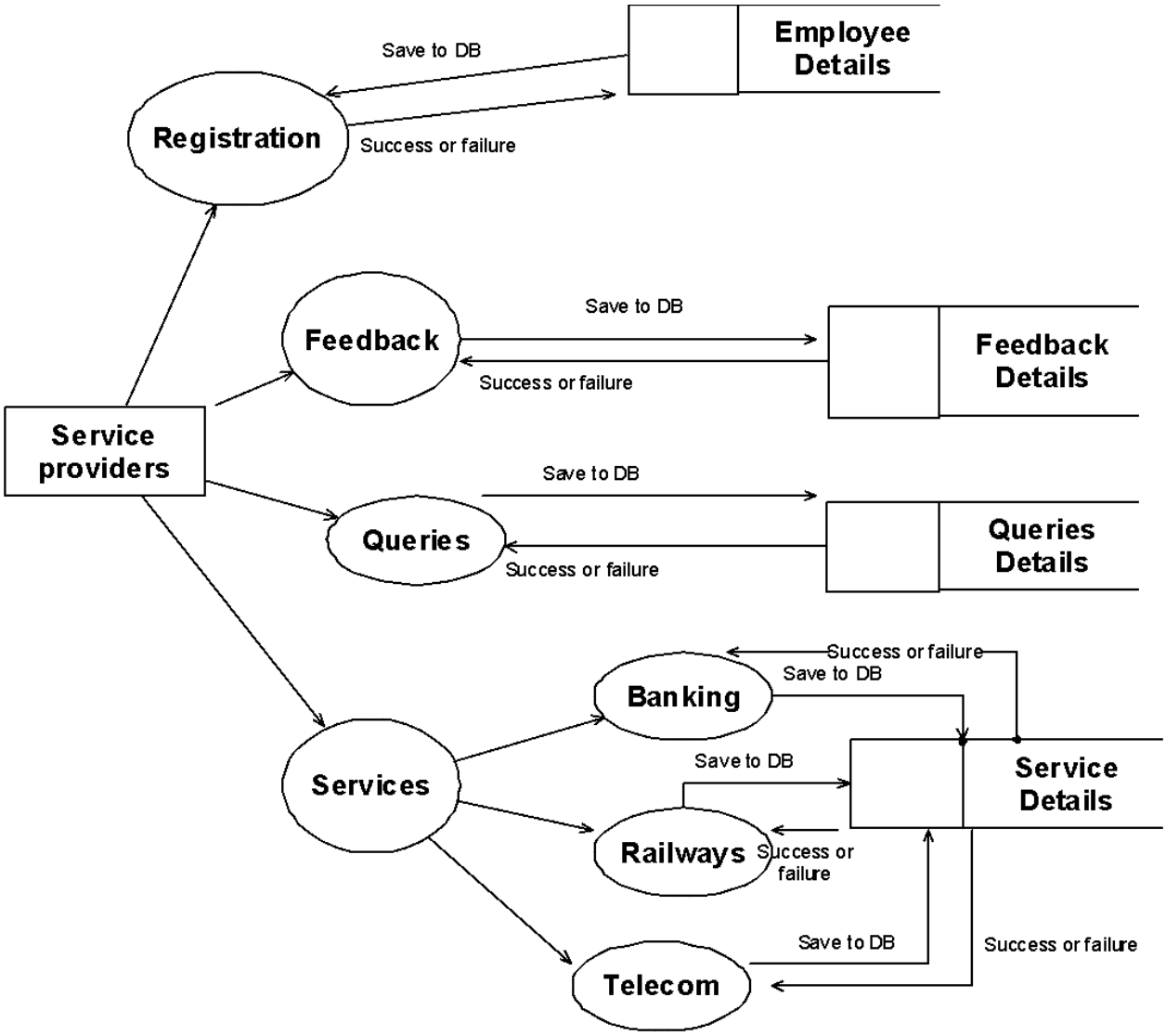
## Level 2 Diagram for Admin:



## Level 2 Diagram for Customers:

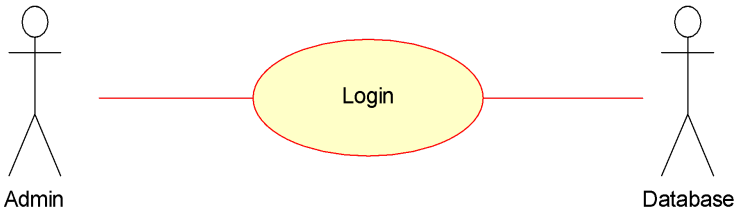


**Level 2 Diagram for Service Providers:**

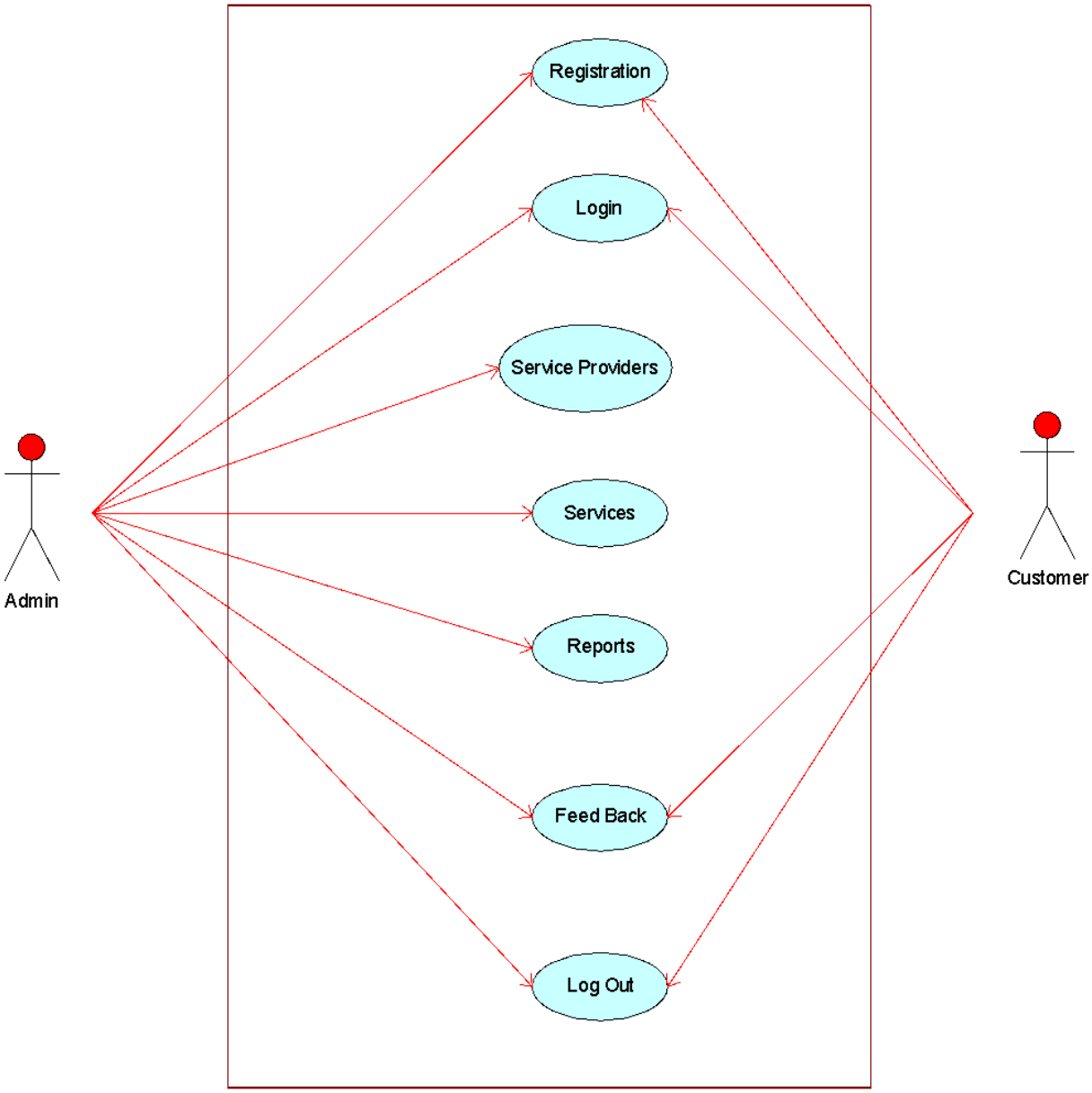


## 5.3. UML Diagrams:

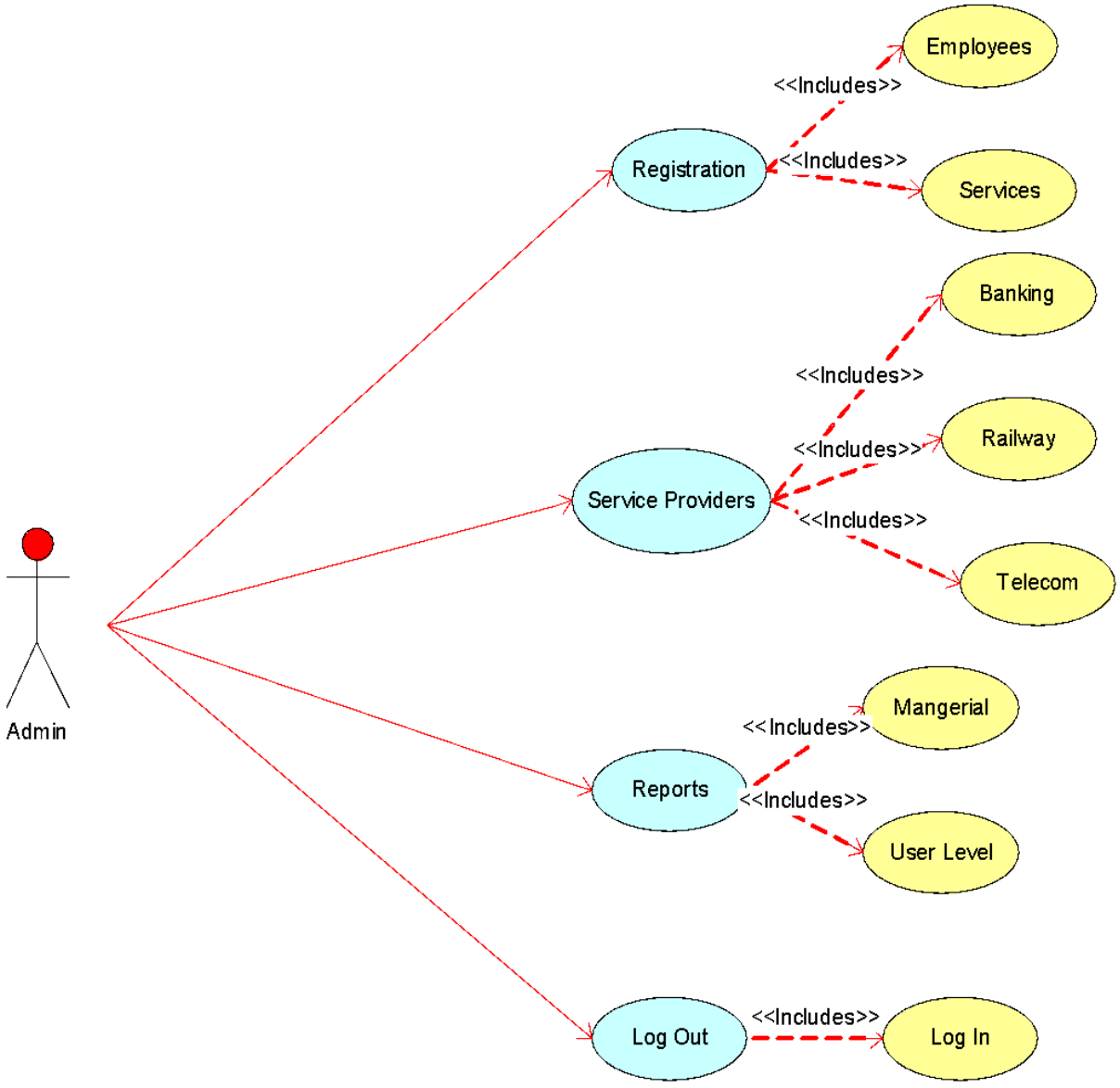
### USE CASE FOR LOGIN PROCESS



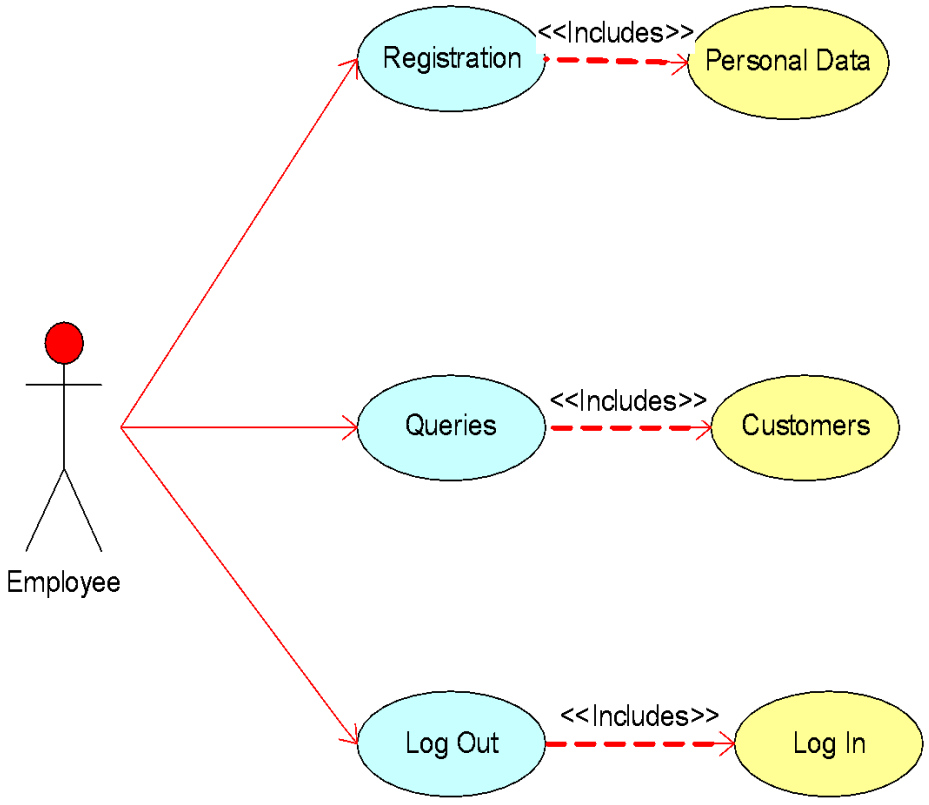
## Over view Use Case Diagram:



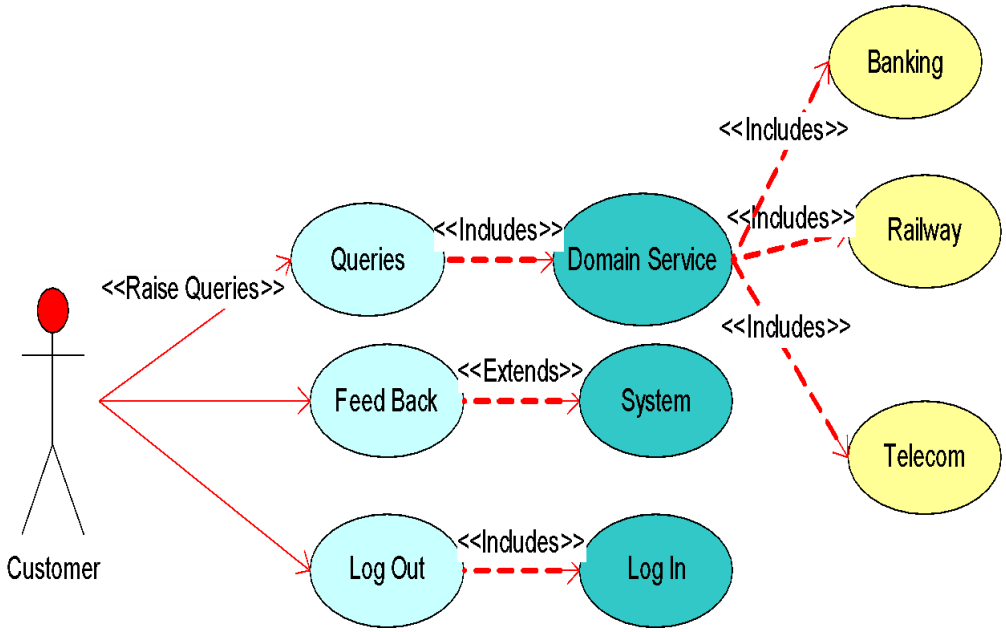
## Admin Use Case Diagram:



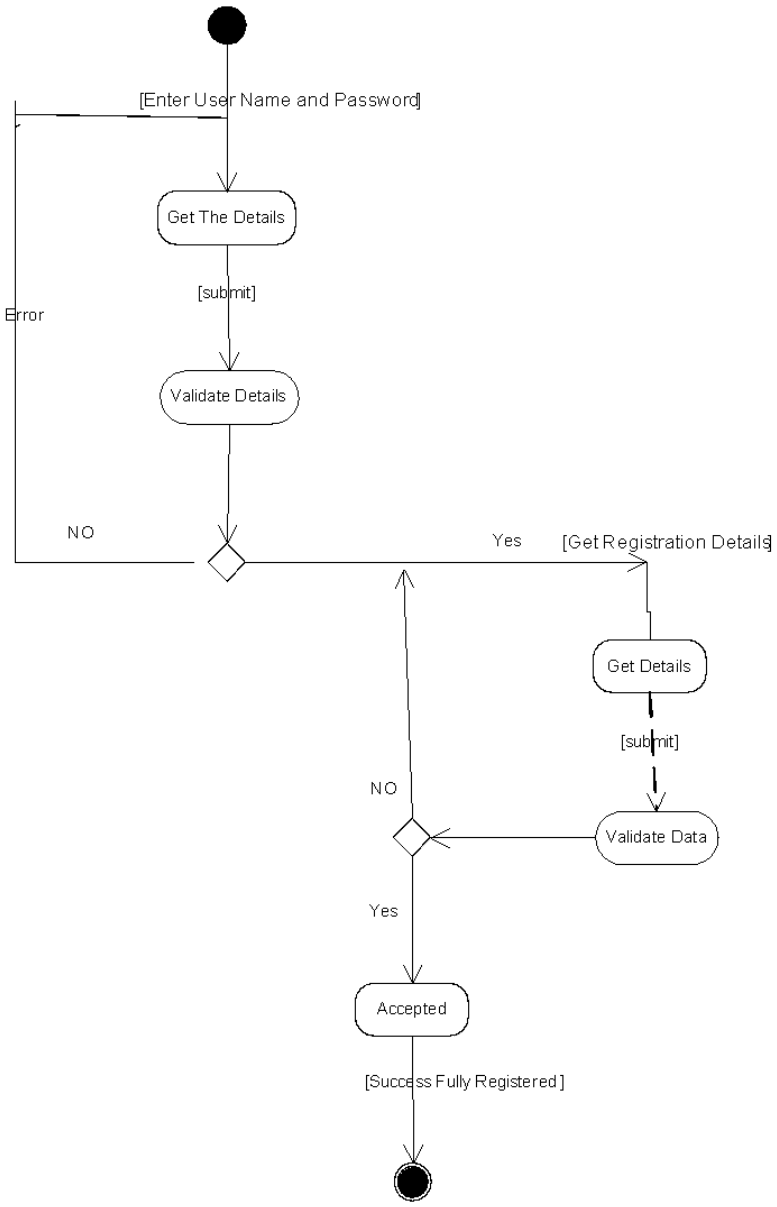
## Employee Use Case Diagram



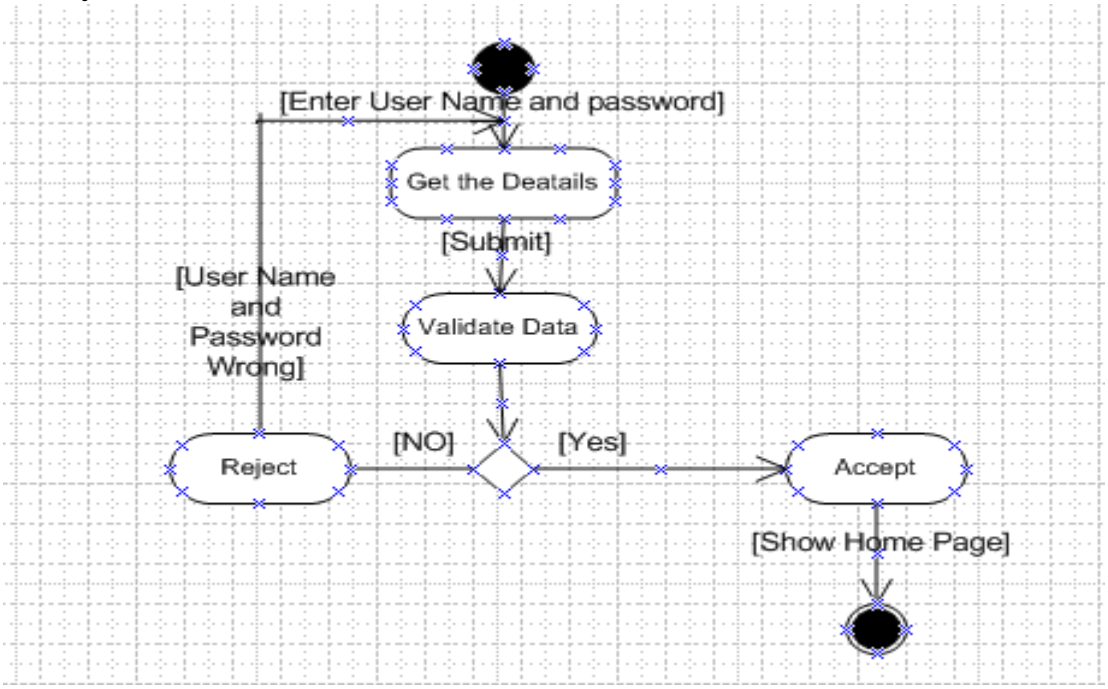
## Customer Use Case Diagram:



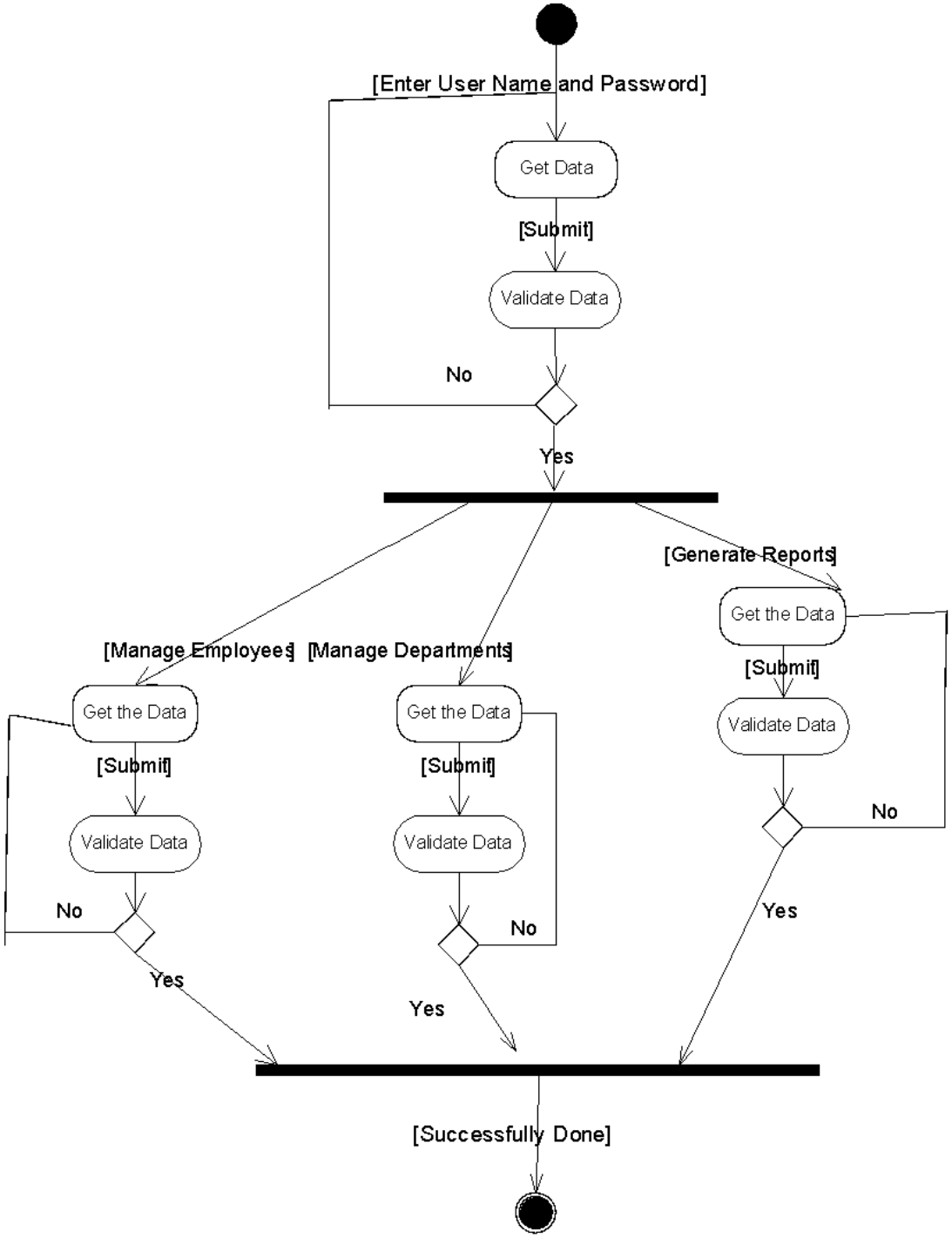
## Activity Diagrams:
### Registration Activity



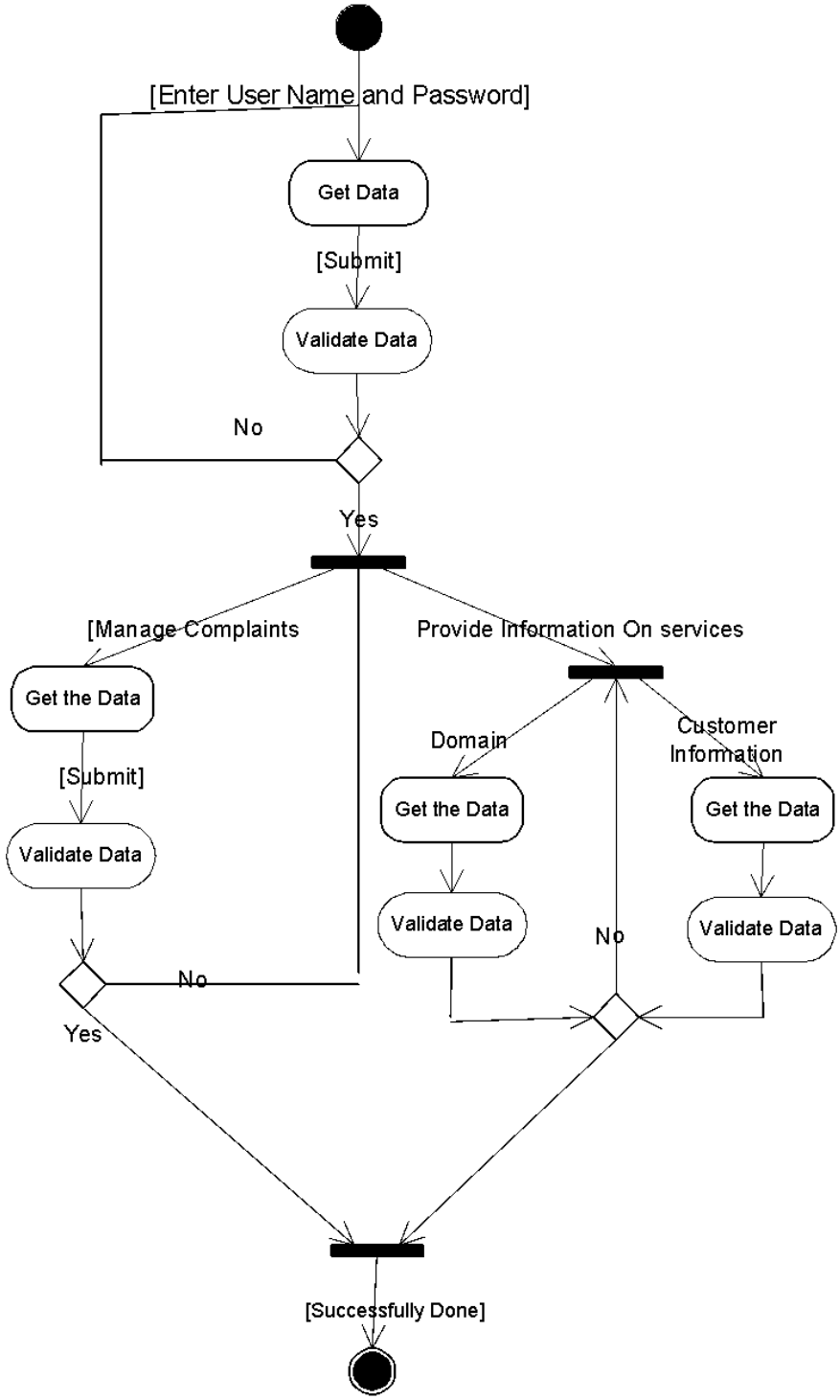
## Login Activity



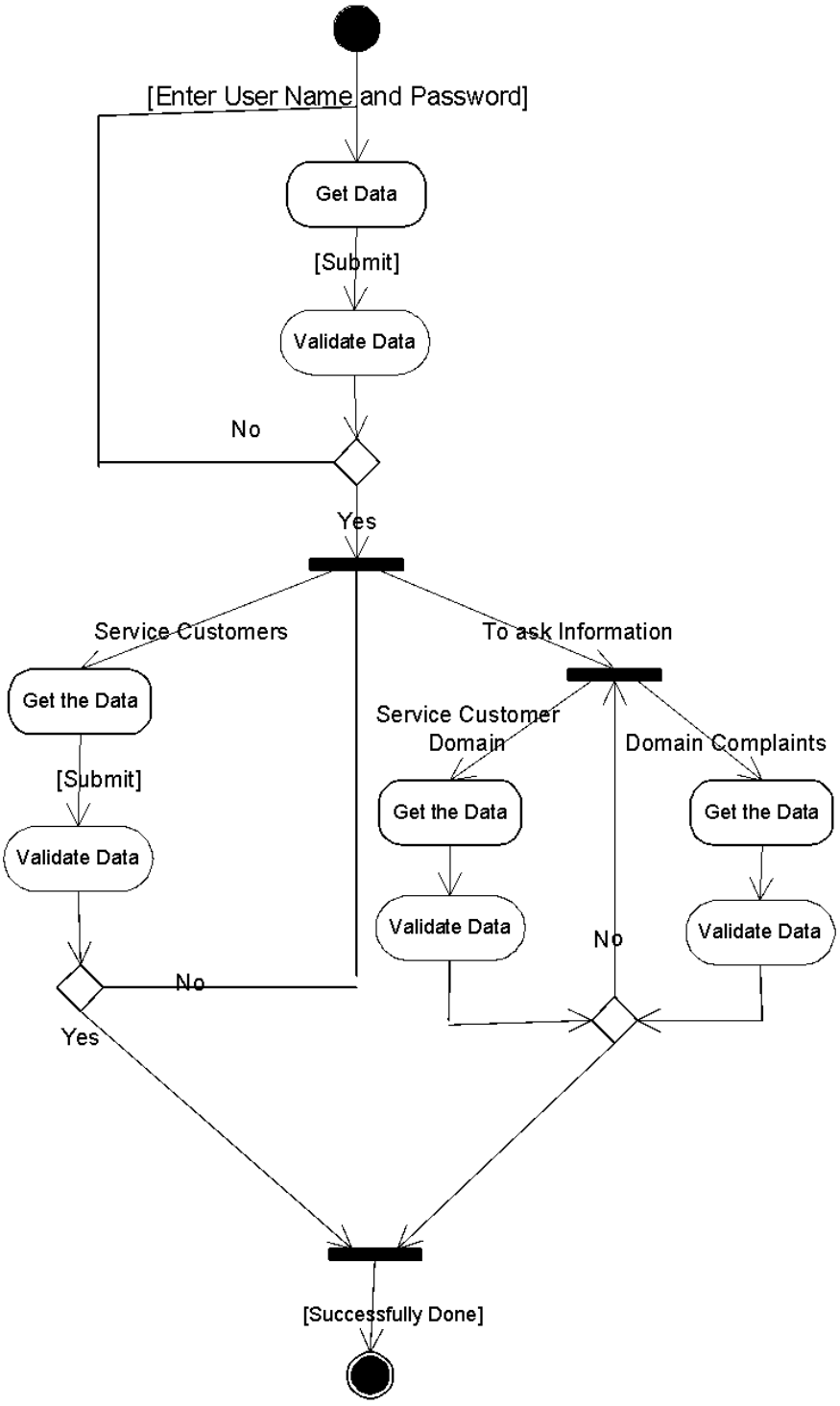
## Admin Activity Diagram:



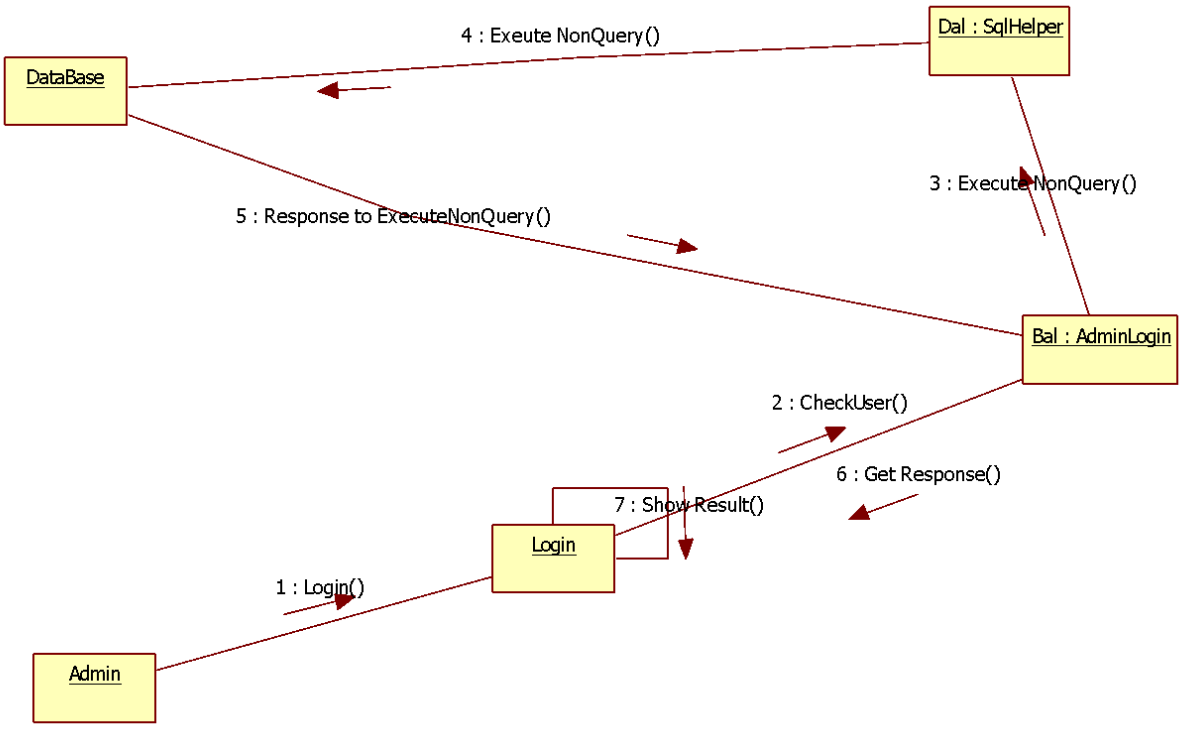
## Employee Activity Diagram:



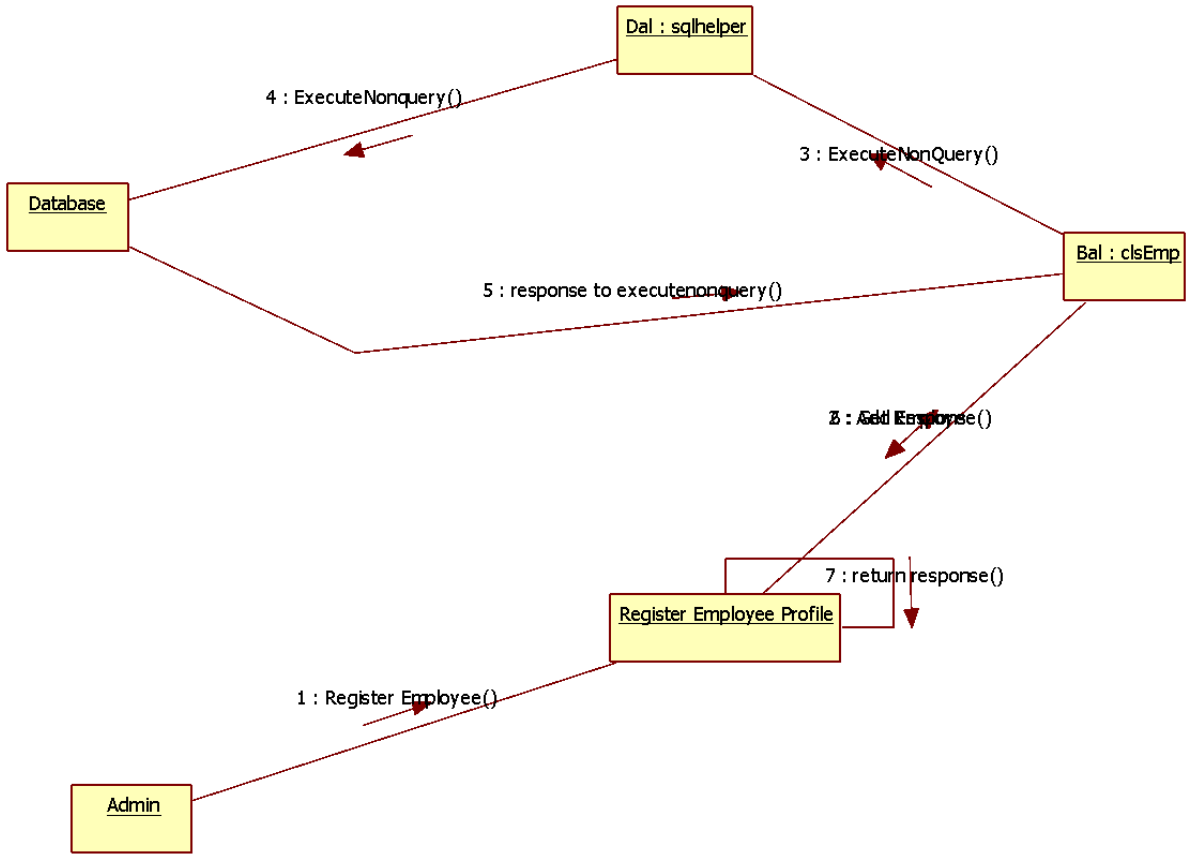
## Customer  Activity Diagram:



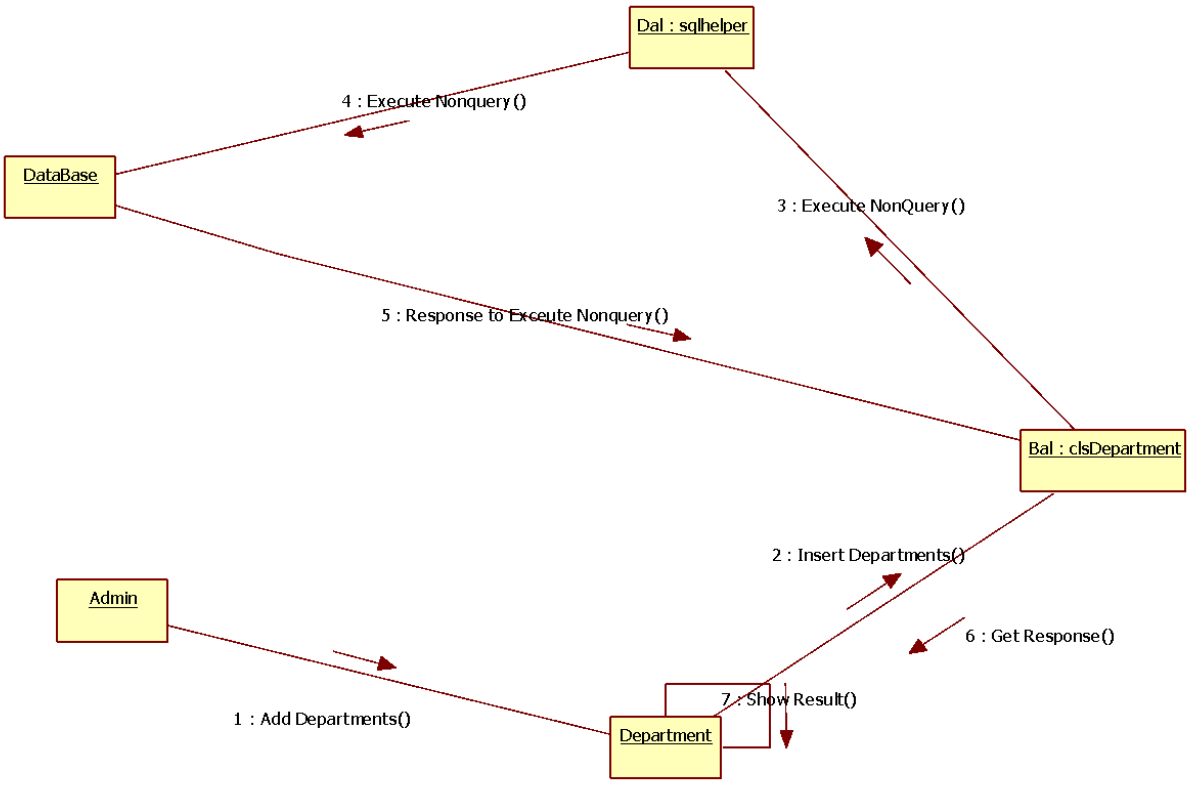
## Administrator Login Collaboration Diagram



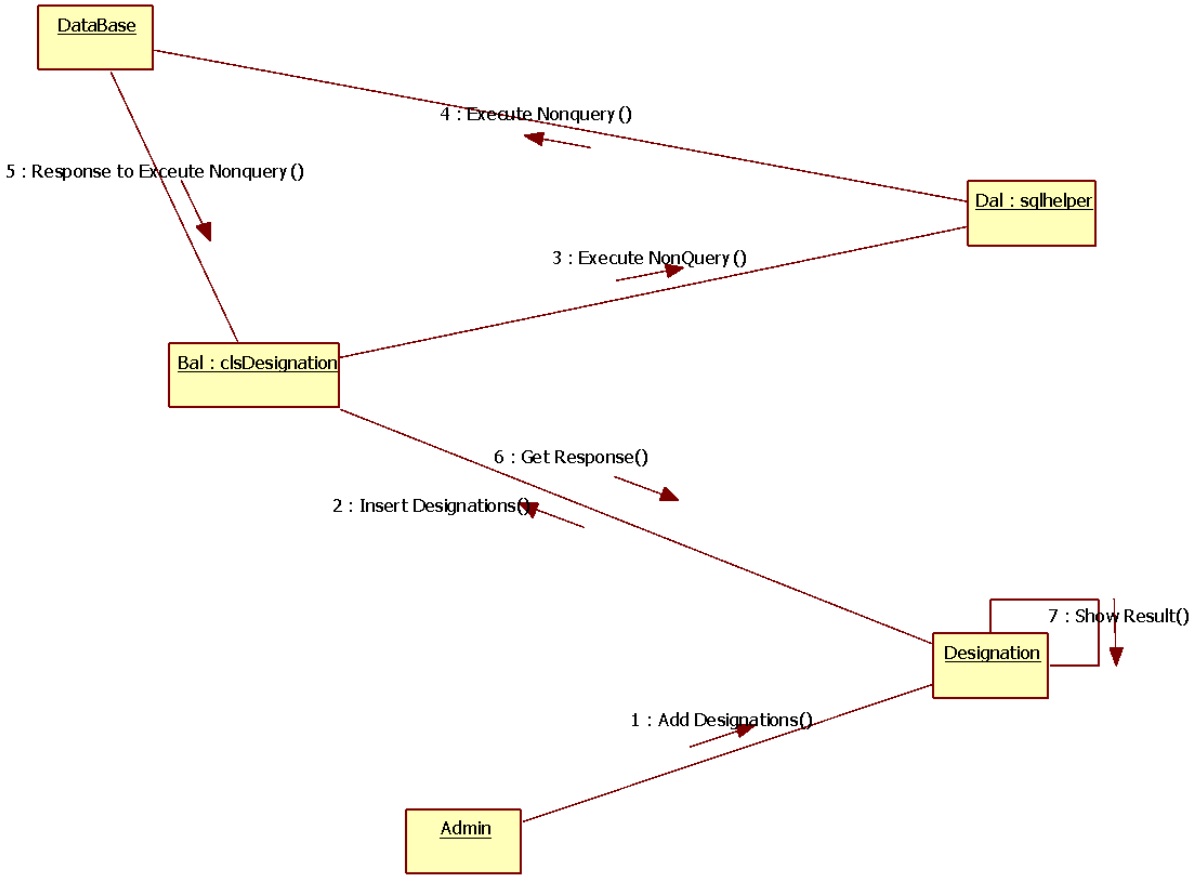
## Administrator Register Employees Collaboration Diagram



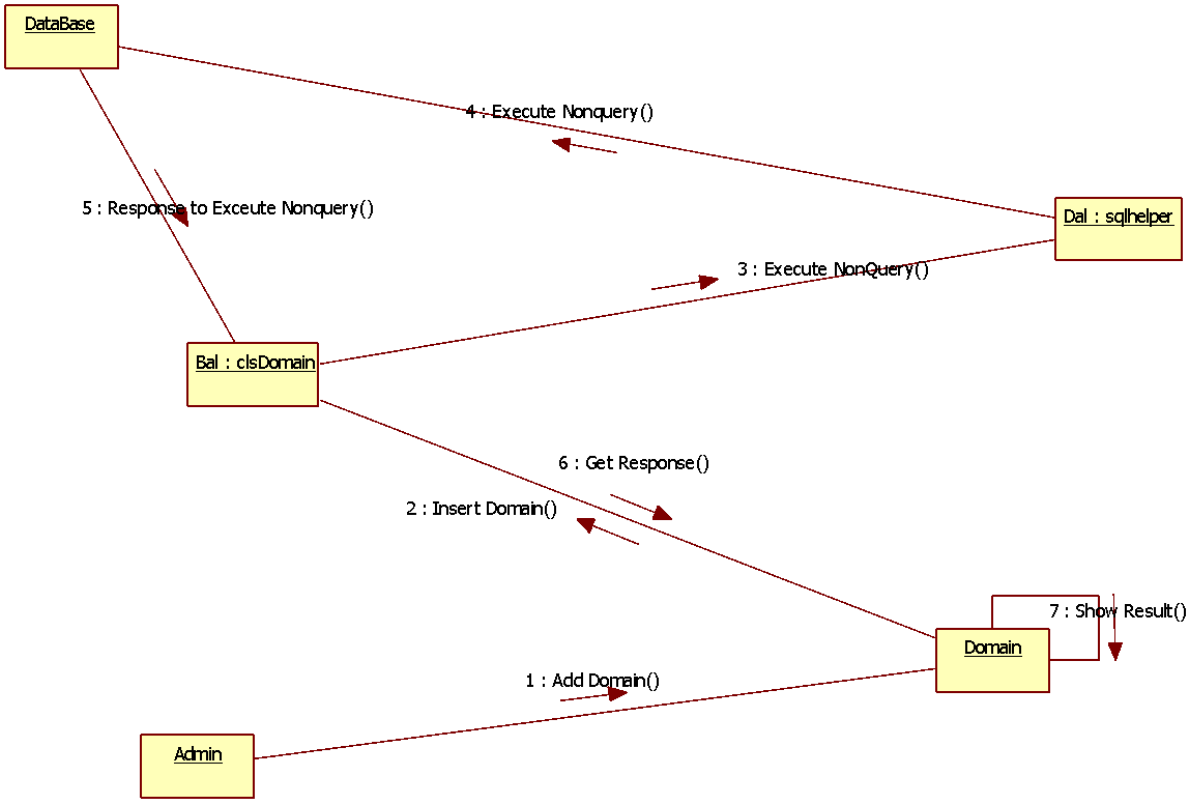
## Administration  Add Department Collaboration Diagram



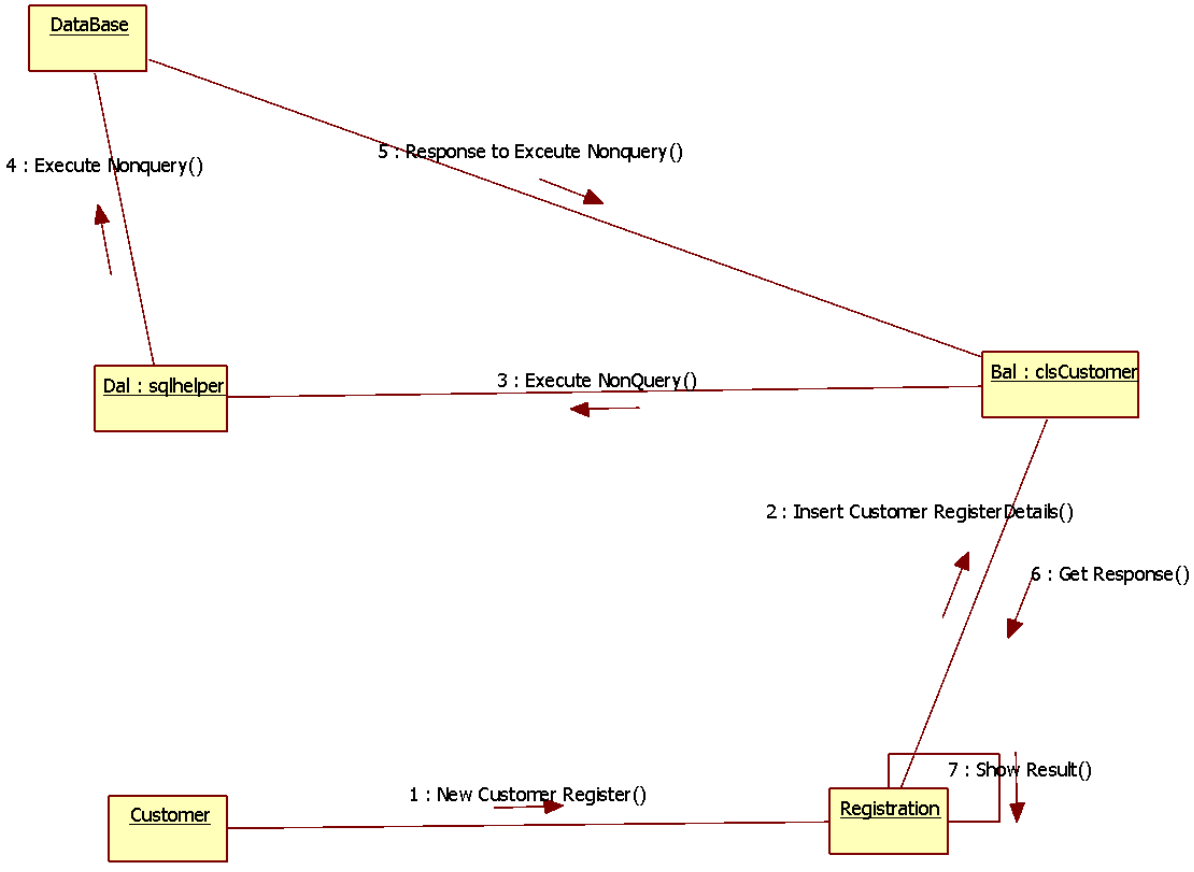
## Administration Add Designations Collaboration Diagram



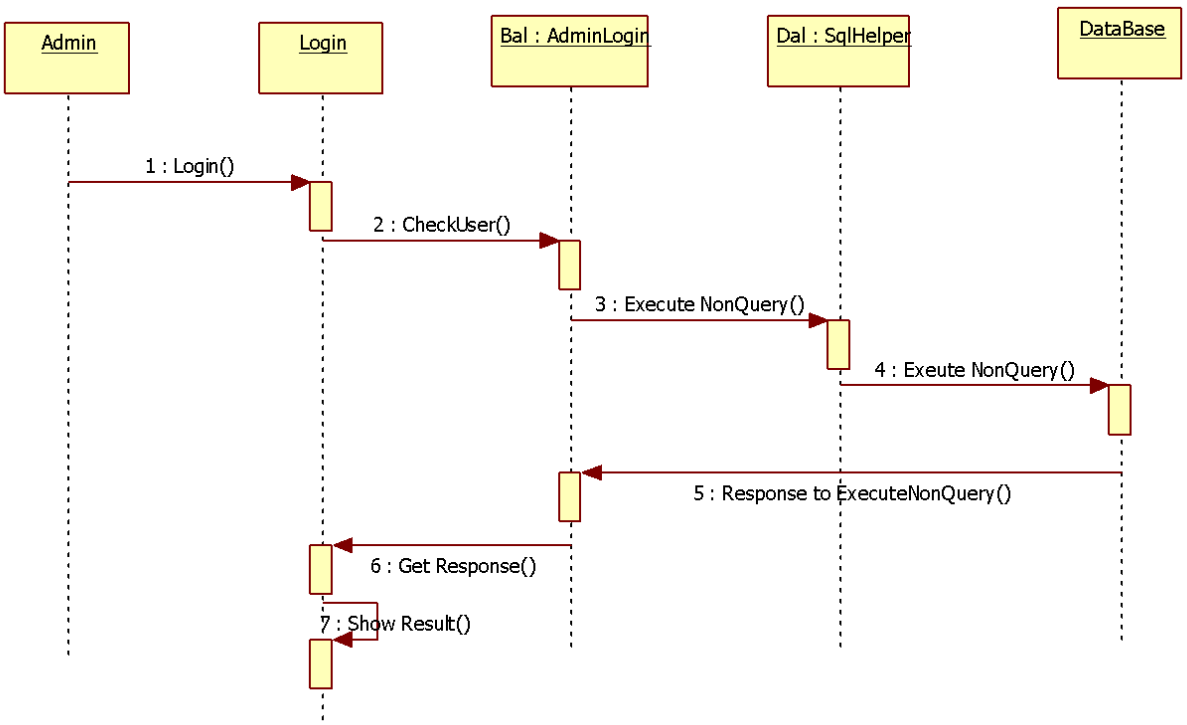
## Administration Add Domain Collaboration Diagram



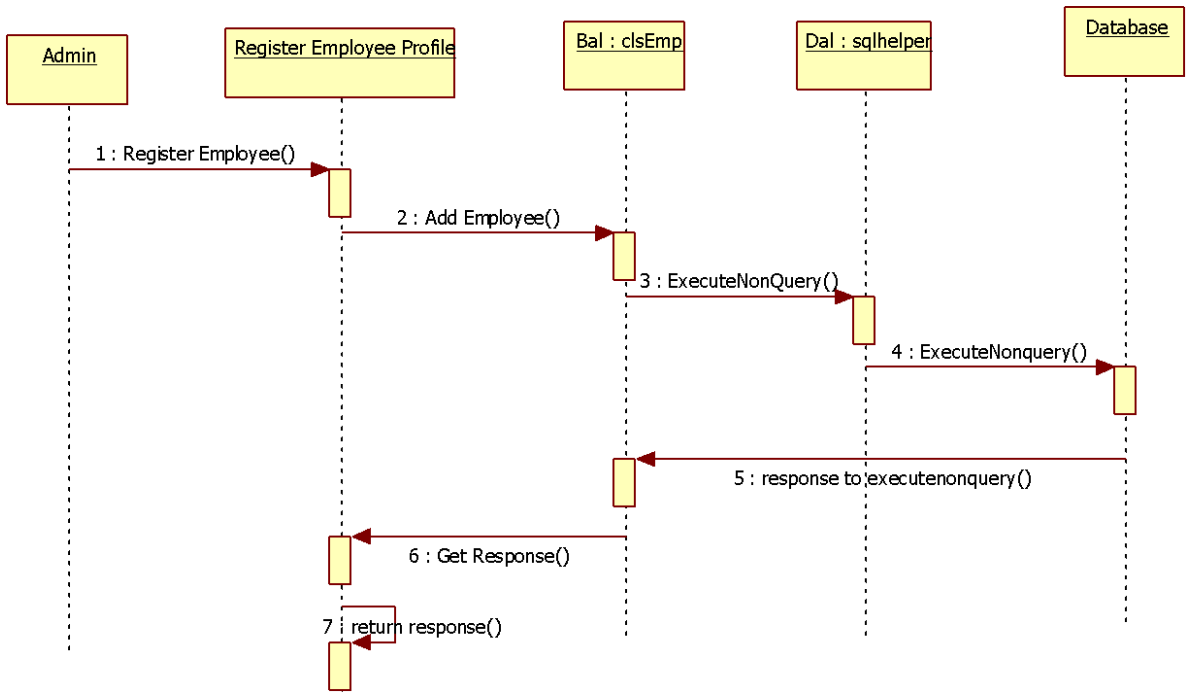
## Customer Registration Collaboration Diagram



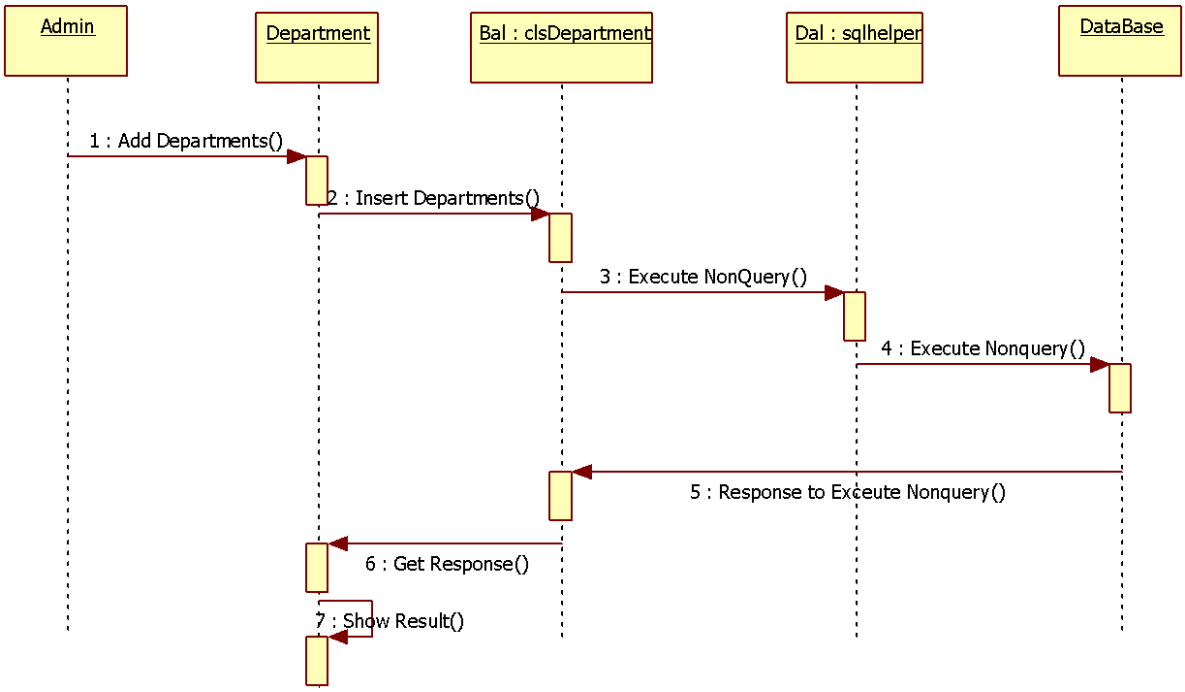
## Administrator Login Sequence Diagram



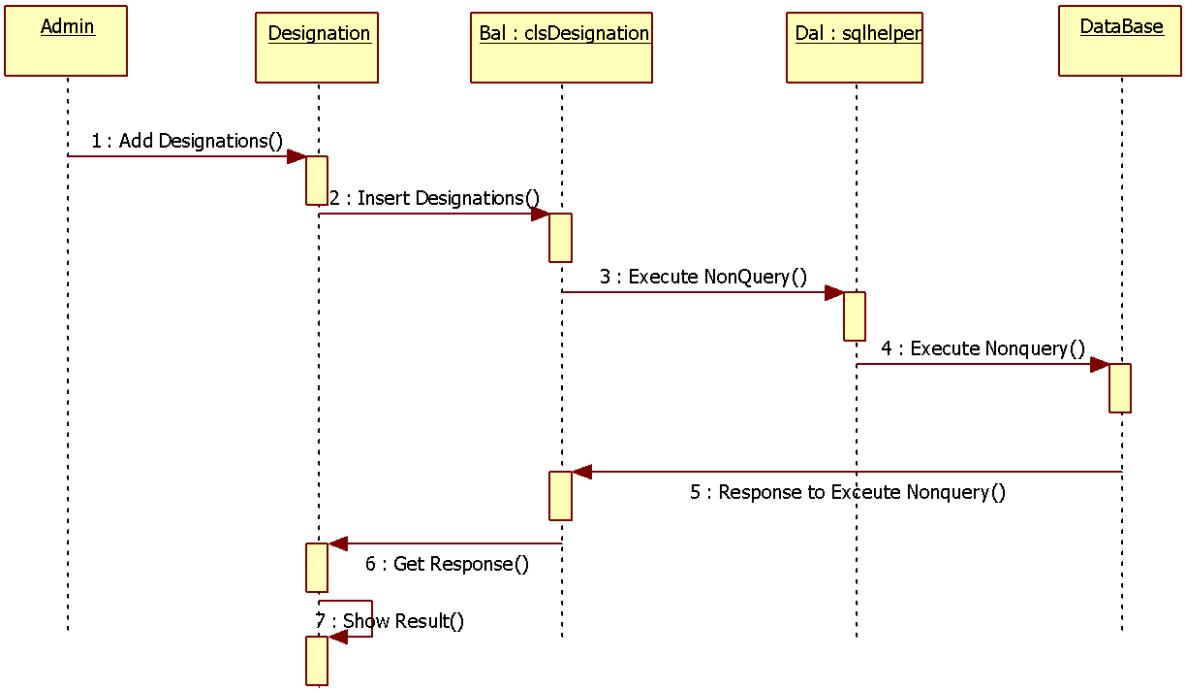
## Administrator Register Employees  Sequence Diagram



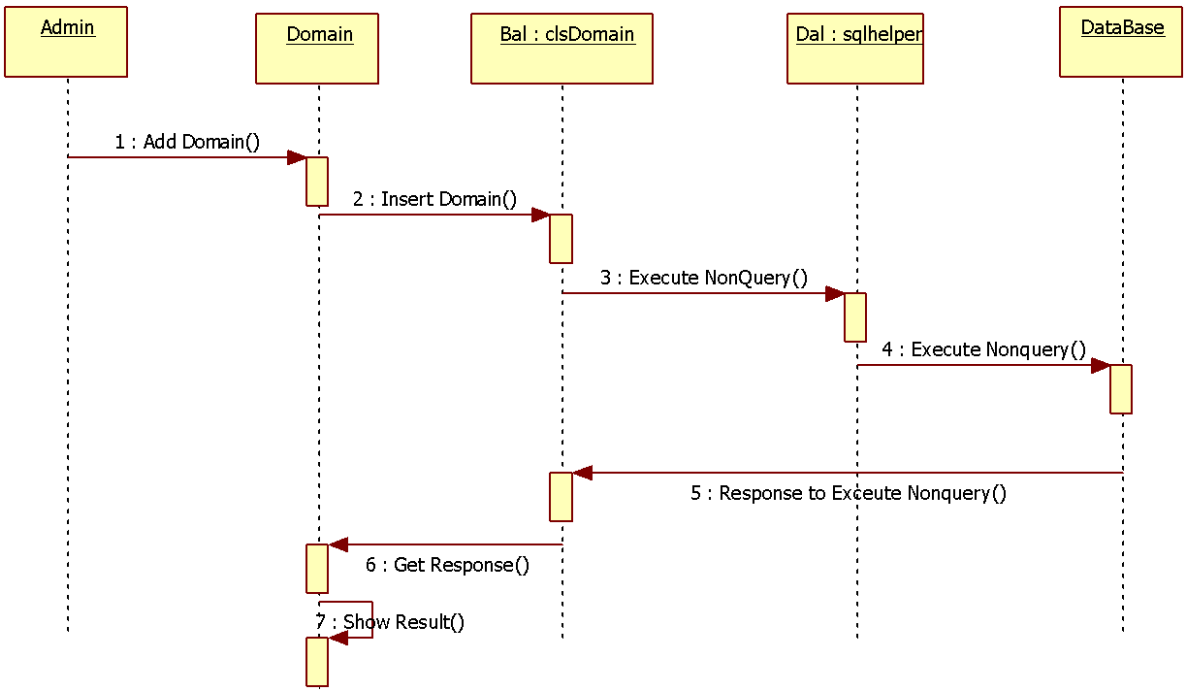
## Administration Add Department Sequence Diagram



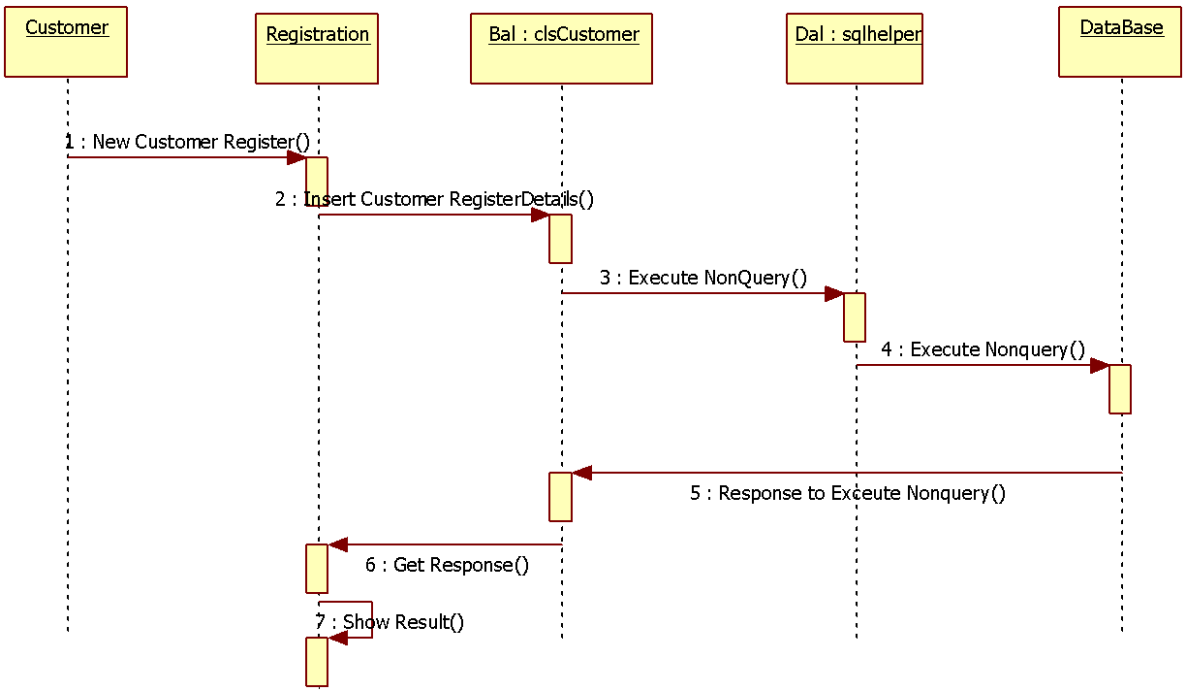
## Administration Add Designations Sequence Diagram



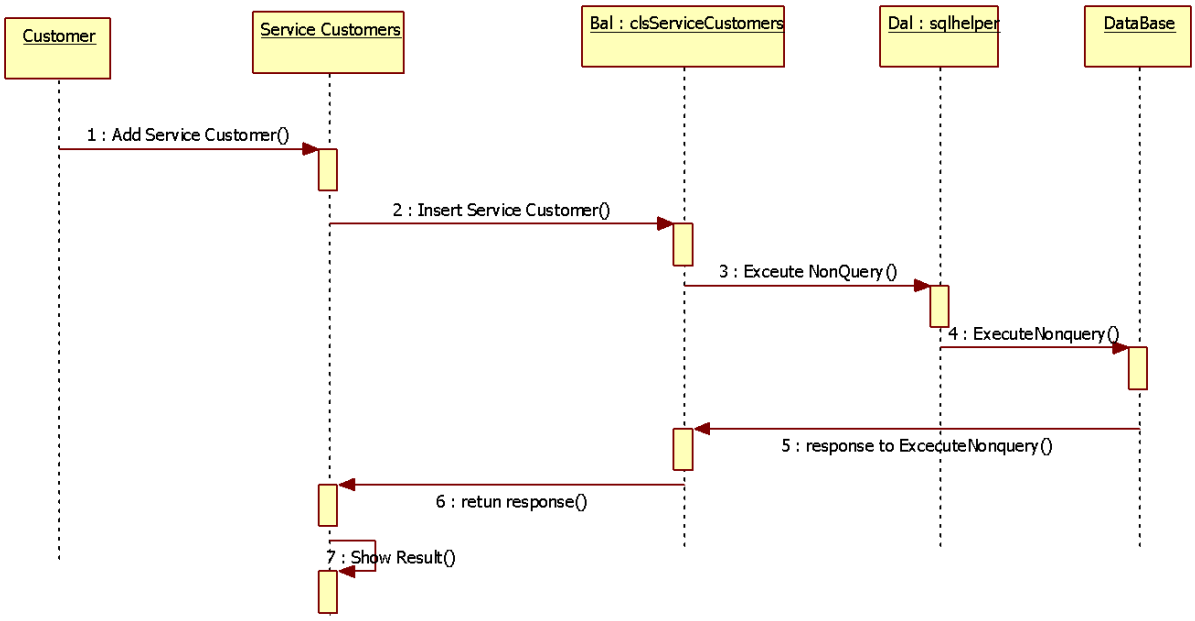
## Administration Add Domain Sequence Diagram



## Customer Registration Sequence Diagram



## Customer Services Customer Sequence Diagram



## 5.4 Data Dictionary:

### Entities:

Login

Department

Designation

Domain

Complaint

ComplaintType

Report

Customer

CustComplaintReg

Custfeedback

Employee

ServiceCustDomain

ServiceCustDomainComplaintTable

ServiceCustDomainCustomers

ServiceCustDomain Incharge

ServiceCustDomainPhone

ServiceCustomers

### Entities with attributes:

### Login:

- ✔ LoginId
- ✔ Password

- ✔ Emp_Customer_Id
- ✔ Role

### Department:

- DeptId
- DeptName
- Abbreviation
- DeptinchargeId
- Description

### Designation:

- DesgId
- DesgName
- Abbreviation
- DesgInchargeId
- superiorDesgId
- Description

### Domain:

- DomainId
- DomainName
- DomainAbbreviation
- Domainesc
- DomainInchargeId
- DeptId

### Complaint:

- serviceCutomerDomainId
- ComplaintTypeId

### ComplaintType:

- ✔ ComplaintTypeId

- ✔ ComplaintTypeName
- ✔ ComplaintTypeDesc
- ✔ ComplaintTypeDesc
- ✔ InchargeId

**Report:**

- ☐ ReportId
- ☐ ReportDate
- ☐ RepotrTime
- ☐ ReportGeneratedEmpId
- ☐ ReportFileToSave

**Customer:**

- ✔ CustomerId
- ✔ CustomerName
- ✔ DOB
- ✔ EmailId
- ✔ PhoneNumber
- ✔ CustDesc
- ✔ CustDOR
- ✔ Address

**CustComplaintReg:**

- ✔ ComplaintRegId
- ✔ ComplaintRegDate
- ✔ RegistrationTime
- ✔ CustomerId
- ✔ ServiceCustDomain
- ✔ PhoneNum
- ✔ CustComplaintDomain
- ✔ TextOfComplaint

- ✔ EmpId
- ✔ VoiceTextOfCompaint
- ✔ TextFile
- ✔ ComplaintStatus
- ✔ ComplaintEscalatedStatus
- ✔ CountOfEscalation
- ✔ ComplaintResponseText
- ✔ VoiceFileName
- ✔ Complaintsevearity

## Custfeedback:

- FeedBackId
- CustId
- Date
- Text
- EmailId
- FeedbackVoiceFile
- FileName
- ComplaintRegId

## Employee:

- EmployeeId
- EmpName
- EmpDOB
- EmpDOJ
- Address
- Email
- PhoneNumber
- DeptId
- DesgId
- EmpPhoto
- EmpFileName

**ServiceCustDomain:**

- ServiceCustDomainId
- ServiceCustId
- ServiceCustDomainMasterId
- ServiceCustDomainPhoneNum
- ServiceCustDomainInchargeId
- ServiceCustDomainManagerId
- ServiceCustDomainEmailId
- ServiceCustDomainAddress
- ServiceCustDomainDesc

**ServiceCustDomainComplaintTable:**

- ✔ ServiceCustDomainId
- ✔ ComplaintTypeId

**ServiceCustDomainCustomers:**

- ✔ ServiceCustDomainId
- ✔ CustomerId

**ServiceCustDomain Incharge:**

- ✔ ServiceCustDomainId
- ✔ ServiceCustDomainInchargeId
- ✔ EmpId

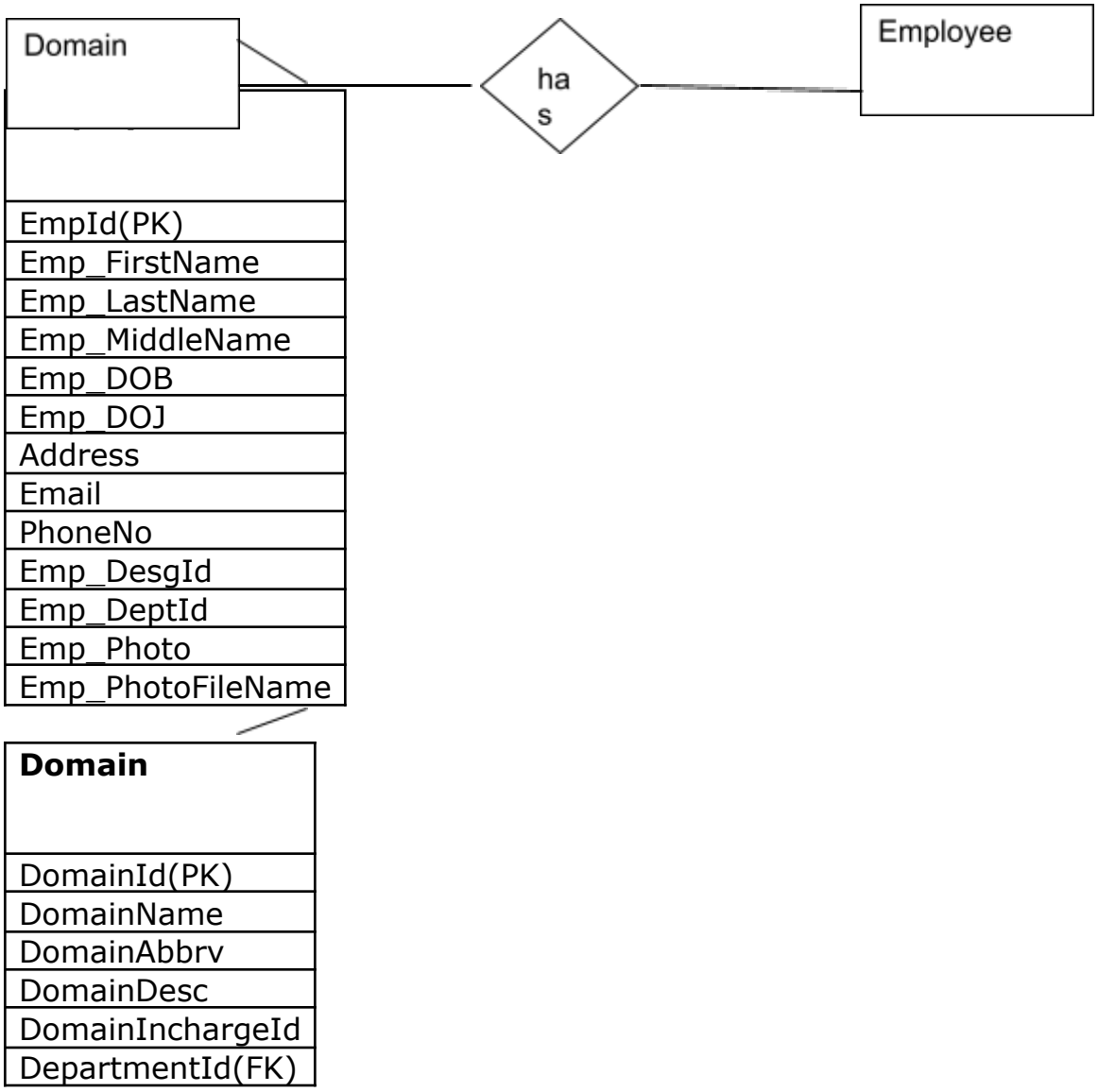
**ServiceCustDomainPhone:**

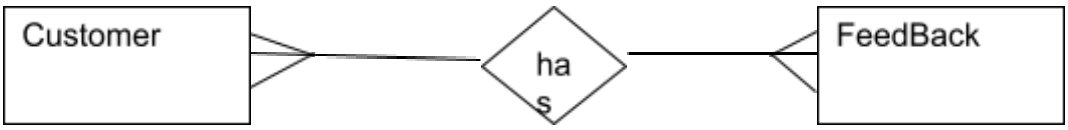
- ServiceCustDomainId
- PhoneNum1
- PhoneNum2
- PhoneNum3

**ServiceCustomers:**

- ✔ ServiceCustId
- ✔ ServiceCustName

- ✔ ServiceCustDOR
- ✔ CustAddress
- ✔ CustPhoneNum
- ✔ CustEmail
- ✔ serviceCustInchargeId

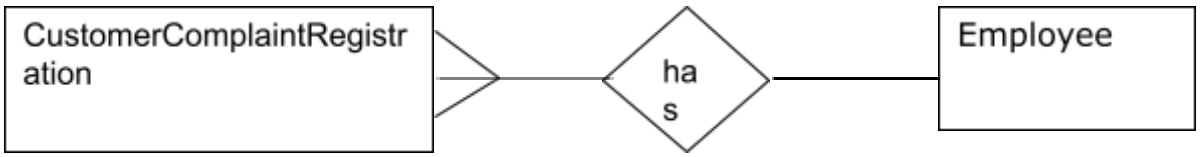





| **Customer** |
|---|
| CustomerId(PK) |
| CustomerName |
| CustomerDOB |
| CustomerPhoneNo |
| CustomerEmailId |
| CustomerDesc |
| CustomerDOR |
| CustomerAddress |

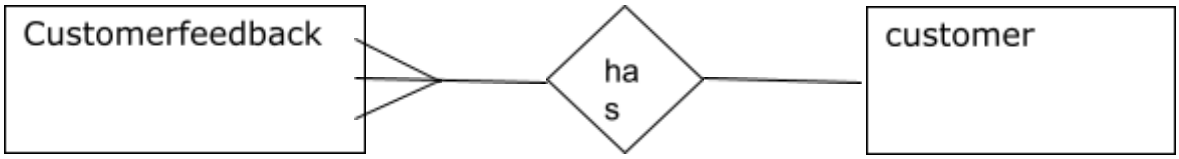
| CustomerFeedBack |
|---|
| CFB_Id(PK) |
| CustomerId(FK) |
| FeedBackId(FK) |
| **FeedBack** |
| FeedBackId(PK) |
| CustomerId |
| FeedBackDate |
| feedBackText |

| |
|---|
| Emailid |
| FeedBackVoiceFile |
| VoiceFileName |
| ComplaintRegistrationId |



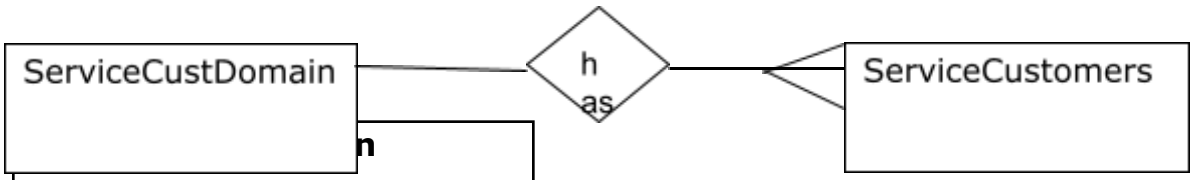

| **CustomerComplaintRegistration** |
|---|
| ComplaintRegistraionId(PK) |
| RegistrationDate |
| RegistrationTime |
| CustomerId |
| ServiceCustomerDomain |
| PhoneNo |
| TextOfCompklaint |
| EmpId |
| VoiceTextOfComplaint |
| TextFile |
| ComplaintStatus |
| ComplaintEscalatedStatus |
| CountOfEscalation |
| ComplaintResponseText |
| ComplainrResponseVoice |
| VoiceFileName |
| ComplaintSevearity |

| **Employee** |
|---|
| EmpId(PK) |
| Emp_FirstName |
| Emp_LastName |
| Emp_MiddleName |
| Emp_DOB |
| Emp_DOJ |
| Address |
| Email |
| PhoneNo |
| Emp_DesgId(FK) |
| Emp_DeptId(FK) |
| Emp_Photo |
| Emp_PhotoFileName |



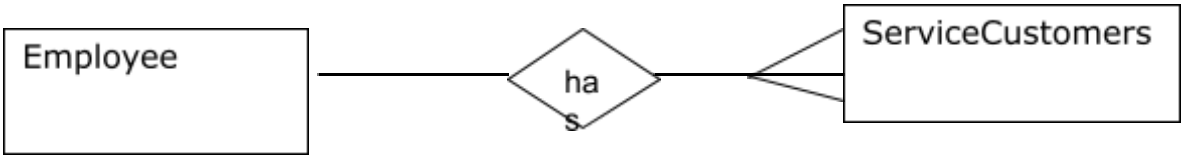

| **Customer** |
|---|
| CustomerId |
| CustomerName |
| CustomerDOB |
| CustomerPhoneNo |
| CustomerEmailId |
| CustomerDesc |
| CustomerDOR |
| CustomerAddress |

| **Customerfeedback** |
|---|
| FeedBackId |
| CustomerId |
| FeedBackDate |
| feedBackText |
| Emailid |
| FeedBackVoiceFile |
| VoiceFileName |
| ComplaintRegistrationId |



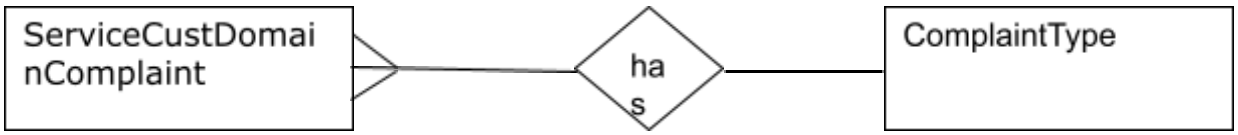

| **n** |
|---|
| ServiceCustDomainId |
| ServiceCustId |
| ServiceCustDomainMasterId |
| ServiceCustDomainPhoneNum |
| ServiceCustDomainInchargeId |
| ServiceCustDomainManagerId |
| ServiceCustDomainEmailId |
| ServiceCustDomainAddress |
| ServiceCustDomainDesc |

| **ServiceCustomers** |
|---|
| ServiceCustId |
| ServiceCustName |
| ServiceCustDOR |
| CustAddress |
| CustPhoneNum |
| CustEmail |
| serviceCustInchargeId |
| ServiceCustDesc |
| ServiceCustDomainId |



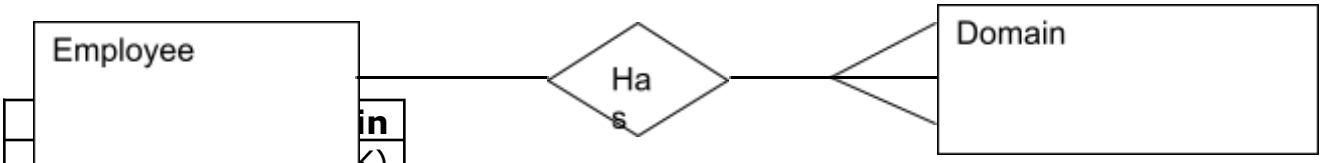

| **Employee** |
|---|
| EmpId(PK) |
| Emp_FirstName |
| Emp_LastName |
| Emp_MiddleName |
| Emp_DOB |
| Emp_DOJ |
| Address |
| Email |
| PhoneNo |
| Emp_DesgId |
| Emp_DeptId |
| Emp_Photo |
| Emp_PhotoFileName |
| **ServiceCustomers** |
| ServiceCustId(PK) |
| ServiceCustName |
| ServiceCustDOR |
| CustAddress |

| CustPhoneNum |
| --- |
| CustEmail |
| serviceCustInchargeId |
| ServiceCustDesc |
| EmpId |




| **ServiceCustDomainComplaint** |
| --- |
| ServiceCustDomainID |
| ComplaintTypeID |

| **ComplaintType** |
| --- |
| ComplaintTypeName |
| ComplaintTypeId |
| ComplaintTypeAbbrv |
| ComplaintTypeDesc |




| **in** |
| --- |
| K) |
| DomainName |
| DomainAbbrv |
| DomainDesc |
| DomainInchargeId(FK) |
| DepartmentId(FK) |
| EmpId |
| **Employee** |
| EmpId(PK) |
| Emp_FirstName |
| Emp_LastName |
| Emp_MiddleName |
| Emp_DOB |
| Emp_DOJ |
| Address |
| Email |
| PhoneNo |
| Emp_DesgId(FK) |
| Emp_DeptId(FK) |
| Emp_Photo |
| Emp_PhotoFileName |

## 5.5DB Tables:

### ComplaintEscalationTable:

| Sno | Coloumn name | Dtatype(size) | Constraints | References |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1 | ComplaintEsclationId | Int | Primarykey | |
| 2 | ComplaintEsclationDate | Varchar(50) | NotNull | |
| 3 | ComplaintEsclationTime | Varchar(50) | NotNull | |
| 4 | ComplaintRegistrationId | Int | Foreign Key | ComplaintRegistration |
| 5 | EmployeeId | Int | Foreign Key | Employee |
| 6 | ComplaintEsclationText | Varchar(50) | NotNull | |
| 7 | PhoneNo | Varchar(50) | NotNull | |
| 8 | VoiceTextFileOfCompliant | Varbinary(max) | NotNull | |
| 9 | VoiceTextFile | Varchar(50) | NotNull | |
| 10 | ComplaintResponseText | Varchar(50) | NotNull | |
| 11 | ComplaintResponseVoice | Varbinary(max) | NotNull | |
| 12 | VoiceFileName | Varchar(50) | NotNull | |

**Complaint Type:**

| Sno | Coloumn name | Dtatype(size) | Constraints | References |
|---|---|---|---|---|
| 1 | ComplaintTypeID | Int | Primarykey | |
| 2 | ComplaintTypeName | Varchar(50) | NotNull | |

| | | | | |
|---|---|---|---|---|
| 3 | ComplaintTypeAbbr | Varchar(50) | NotNull | |
| 4 | ComplaintTypeDesc | Varchar(500) | NotNull | |
| 5 | InchargeId | Int | ForeignKey | ServiceCustDomain Incharge |

**Customer Table:**

| Sno | Coloumnname | Datatype(size) | Constraints | References |
|---|---|---|---|---|
| 1 | CustomerId | Int | Primarykey | |
| 2 | CustomerName | Varchar(100) | NotNull | |
| 3 | CustomerDOB | Datetime | NotNull | |
| 4 | CustomerPhoneNo | Varchar(50) | NotNull | |
| 5 | CustomerEmailId | Varchar(50) | NotNull | |
| 6 | CustomerDesc | Varchar(50) | NotNull | |
| 7 | CustomerDOR | datetime | NotNull | |
| 8 | CustomerAddress | Varchar(5000) | NotNull | |

**Login Table:**

| Sno | ColoumnName | Datatype(size) | Constraints | References |
|---|---|---|---|---|
| 1 | UserId | Int | PrimaryKey | |
| 2 | UserName | Varchar(50) | NotNull | |
| 3 | Password | Varchar(50) | NotNull | |
| 4 | Emp_cus_Id | Int | ForeignKey | EmpCust |
| 5 | Role | Varchar(50) | NotNull | |

**CustomerComplaintRegistration:**

| Sno | ColoumnName | Datatype(size) | Constraints | References |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1 | ComplaintRegistraionId | Int | Primarykey | |
| 2 | RegistrationDate | Varcar(50) | Notnull | |
| 3 | RegistrationTime | varcarI(50) | NotNull | |
| 4 | CustomerId | Int | NotNull | |
| 5 | ServiceCustomerDomain | int | NotNull | |
| 6 | PhoneNo | Varchar(50) | NotNull | |
| 7 | TextOfCompklaint | Varchar(1000) | NotNull | |
| 8 | EmpId | int | ForeignKey | Employee |
| 9 | VoiceTextOfComplaint | Varbinary(max) | Notnull | |
| 10 | TextFile | Varchar(50) | NotNull | |
| 11 | ComplaintStatus | Varchar(50)1 | NotNull | |
| 12 | ComplaintEscalatedStatus | Varchar(50) | Notnull | |
| 13 | CountOfEscalation | Int | NotNull | |
| 14 | ComplaintResponseText | Varchar(1000) | NotNull | |
| 15 | ComplainrResponseVoice | Varbinary(max) | NotNull | |
| 16 | VoiceFileName | Varchar(50) | NotNull | |
| 17 | ComplaintSevearity | Varcar(50) | Notnull | |

**Customer feedback table:**

| **Sno** | **Coloumnname** | **Datatype(size)** | **Constraints** | **References** |
|---|---|---|---|---|
| 1 | FeedBackId | Int | primarykey | |
| 2 | CustomerId | Int | Foreign Key | Customer |
| 3 | FeedBackDate | Datetime | NotNull | |
| 4 | feedBackText | Varchar(50) | NotNull | |
| 5 | Emailid | Varchar(50) | NotNull | |
| 6 | FeedBackVoiceFile | Varbinary(max) | NotNull | |
| 7 | VoiceFileName | Varchar(50) | NotNull | |
| 8 | ComplaintRegistrationId | int | ForeignKey | ComplaintRegistration |

**Department table:**

| Sno | Coloumn name | Datatype(size) | Constraints | references |
|---|---|---|---|---|
| 1 | DepartmentId | Int | Primarykey | |
| 2 | DeptNmae | Varchar(100) | NotNull | |
| 3 | Abbrevation | Varchar(50) | Notnull | |
| 4 | DeptInChargeId | Int | NotNull | |
| 5 | Description | Varchar(1000) | Notnull | |

**Designation Table:**

| Sno | Coloumn name | Datatype(size) | Constraints | References |
|---|---|---|---|---|
| 1 | DesignationId | Int | Primarykey | |
| 2 | Desg_Name | Varchar(50) | NotNull | |
| 3 | Abbrevation | Varchar(50) | NotNull | |
| 4 | DesgInchargeId | Int | Notnull | |
| 5 | SuperinchargeId | Int | NotNull | |
| 6 | Description | Varchar(50) | NotNull | |
| 7 | | | | |

**DomainTable:**

| Sno | Column name | Datatype(size) | Constraints | references |
|---|---|---|---|---|
| 1 | DomainId | Int | Primarykey | |
| 2 | DomainName | Varchar(50) | NotNull | |
| 3 | DomainAbbrv | Varchar(50) | NotNull | |
| 4 | DomainDesc | Varchar(50) | NotNull | |
| 5 | DomainInchargeId | Int | ForeignKey | Domain |
| 6 | DepartmentId | int | ForeignKey | DeptDetails |

**FeedBackEvaluationTable:**

| Sno | Column Name | Data Type(Size) | Consraints | References |
|---|---|---|---|---|
| 1 | FeedBackEvaluationId | Int | Primarykey | |
| 2 | FeedBackEvaluationDate | Datetime | NotNull | |
| 3 | EmployeeInchargeId | Int | ForeignKey | ServiceCustDomain |
| 4 | FeedbackId | Int1 | ForeignKey | FeedbackDetails |
| 5 | FeedBackEvaluationText | Varchar(300) | NotNull | |

| | | | | |
|---|---|---|---|---|
| 6 | FeedBackEvaluationLink | Varbinary(max) | NotNull | |
| 7 | LinkFileName | Varchar(50) | NotNull | |

**EmployeeTable:**

| Sno | ColumnName | Datatype(size) | Constraints | References |
|---|---|---|---|---|
| 1 | EmpId | Int | PrimaryKey | |
| 2 | Emp_FirstName | Varchar(50) | NotNull | |
| 3 | Emp_LastName | Varchar(50) | NotNull | |
| 4 | Emp_MiddleName | Varchar(50) | NotNull | |
| 5 | Emp_DOB | Datetime | NotNull | |
| 6 | Emp_DOJ | Datetime | NotNull | |
| 7 | Address | Varchar(100) | NotNull | |
| 8 | Email | Varchar(50) | NotNull | |
| 9 | PhoneNo | Varchar(50) | NotNull | |
| 10 | Emp_DesgId | Int | ForeignKey | Employee |
| 11 | Emp_DeptId | Int | ForeignKey | Employee |
| 12 | Emp_Photo | Varchar(50) | NotNull | |
| 13 | Emp_PhotoFileName | Varchar(50) | NotNull | |

**Report Table**

| Sno | Column Name | Data Type(Size) | Consraints | References |
|---|---|---|---|---|
| 1 | ReportId | Int | Primarykey | |
| 2 | ReportDate | Varchar(50) | NotNull | |
| 3 | ReportTime | Varchar(50) | NotNull | |
| 4 | ReportGenerationEmpId | Int1 | ForeignKey | ReportGenerationEmp |
| 5 | ReportFileToSave | Varchar(1000) | NotNull | |

**Service Customer Domain Table**

| Sno | Column Name | Data Type(Size) | Consraints | References |
|---|---|---|---|---|
| 1 | ServiceCustomerDomainId | Int | Primarykey | |
| 2 | ServiceCustomerID | int | NotNull | |
| 3 | ServiceCustomerDomainMasterId | Int | ForeignKey | ServiceCustDomain |
| 4 | ServiceCustomerDomainPhone | Varchar(50) | ForeignKey | ServiceCustDomain |
| 5 | ServiceCustomerDomainInchargeId | int | NotNull | |
| 6 | ServiceCustomerDomainManagerName | Varbinary(100) | AllowNull | |
| 7 | ServiceCustomerDomainEmail | Varchar(50) | AllowNull | |
| 8 | ServiceCustomerDomainAddress | Varchar(500) | AllowNull | |
| 9 | ServiceCustomerDomainDesc | Varchar(2000) | AllowNull | |

**Service Customer Domain Complaint Table**

| Sno | Column Name | Data Type(Size) | Consraints | References |
|---|---|---|---|---|
| 1 | ServiceCustomerDomainId | int | Primarykey | |
| 2 | ComplaintTypeId | int | ForiegnKey | ComplaintType |

**Service Customer Domain Customers Table**

| Sno | Column Name | Data Type(Size) | Consraints | References |
|---|---|---|---|---|
| 1 | ServiceCustomerDomainId | int | Primarykey | |
| 2 | CustomerId | int | ForiegnKey | CustomerDetails |

**Service Customer Domain Employee In charge Table**

| Sno | Column Name | Data Type(Size) | Consraints | References |
|---|---|---|---|---|
| 1 | ServiceCustomerDomainId | int | Primarykey | |
| 2 | ServiceCustomerDomainInchargeId | int | ForiegnKey | Domain |
| 3 | EmpId | int | ForiegnKey | Employee |

**Service Customer Domain Phone Table**

| Sno | Column Name | Data Type(Size) | Consraints | References |
|---|---|---|---|---|
| 1 | ServiceCustomerDomainId | int | AllowNulls | |
| 2 | PhoneNo1 | Varchar(50) | AllowNulls | |
| 3 | PhoneNo2 | Varchar(50) | AllowNulls | |
| 4 | PhoneNo3 | VArchar(50) | AllowNulls | |

**Service Customers Table**

| Sno | Column Name | Data Type(Size) | Consraints | References |
|---|---|---|---|---|
| 1 | ServiceCustomerId | Int | Primarykey | |
| 2 | ServiceCustomerName | Varbinary(50) | AllowNull | |
| 3 | ServiceCustomerDOR | DateTime | AllowMulls | |
| 4 | CustomerAddress | Varchar(50) | AllowNulls | |
| 5 | CustomerPhoneNum | Varbinary(50) | AllowNull | |
| 6 | CustomerEmail | Varbinary(50) | AllowNull | |
| 7 | ServiceCustomerInchargeId | int | NotNull | |
| 8 | ServiceCustomerDomainDesc | Varchar(2000) | AllowNull | |

# CHAPTER 6

# 6.Implementation of Project
## Description of Technology Used in Project.

### 6.1. INTRODUCTION TO .NET Framework

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that guarantees safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has two main components: the common language runtime and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework. You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and Remoting, while also enforcing strict type safety and other forms

of code accuracy that ensure security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code. The class library, the other main component of the .NET Framework, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.

The .NET Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features. The .NET Framework not only provides several runtime hosts, but also supports the development of third-party runtime hosts.

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable Web Forms applications and XML Web services, both of which are discussed later in this topic.

Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Windows Forms controls in HTML documents. Hosting the runtime in this way makes managed mobile code (similar to Microsoft® ActiveX® controls) possible, but with significant improvements that only managed code can offer, such as semi-trusted execution and secure isolated file storage.

The following illustration shows the relationship of the common language runtime and the class library to your applications and to the overall system. The illustration also shows how managed code operates within a larger architecture.

## FEATURES OF THE COMMON LANGUAGE RUNTIME

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are intrinsic to the managed code that runs on the common language runtime.

With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions, even if it is being used in the same active application.

The runtime enforces code access security. For example, users can trust that an executable embedded in a Web page can play an animation on screen or sing a song, but cannot access their personal data, file system, or network. The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally featuring rich.

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers

Generate managed code that conforms to the CTS. This means that managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

In addition, the managed environment of the runtime eliminates many common software issues. For example, the runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used. This automatic memory management resolves the two most common application errors, memory leaks and invalid memory references.

The runtime also accelerates developer productivity. For example, programmers can write applications in their development language of choice, yet take full advantage of the runtime, the class library, and components written in other languages by other developers. Any compiler vendor who chooses to target the runtime can do so. Language compilers that target the .NET Framework make the features of the .NET Framework available to existing code written in that language, greatly easing the migration process for existing applications.

While the runtime is designed for the software of the future, it also supports software of today and yesterday. Interoperability between managed and unmanaged code enables developers to continue to use necessary COM components and DLLs.

The runtime is designed to enhance performance. Although the common language runtime provides many standard runtime services, managed code is never interpreted. A feature called just-in-time (JIT) compiling enables all managed code to run in the native machine language of the system on which it

is executing. Meanwhile, the memory manager removes the possibilities of fragmented memory and increases memory locality-of-reference to further increase performance.

Finally, the runtime can be hosted by high-performance, server-side applications, such as Microsoft® SQL Server™ and Internet Information Services (IIS). This infrastructure enables you to use managed code to write your business logic, while still enjoying the superior performance of the industry's best enterprise servers that support runtime hosting.

## .NET FRAMEWORK CLASS LIBRARY

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components can integrate seamlessly with classes in the .NET Framework.

For example, the .NET Framework collection classes implement a set of interfaces that you can use to develop your own collection classes. Your collection classes will blend seamlessly with the classes in the .NET Framework.

As you would expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use the .NET Framework to develop the following types of applications and services:

- Console applications.
- Scripted or hosted applications.
- Windows GUI applications (Windows Forms).
- ASP.NET applications.
- XML Web services.
- Windows services.

For example, the Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form application, you can use the Web Forms classes.

## CLIENT APPLICATION DEVELOPMENT

Client applications are the closest to a traditional style of application in Windows-based programming. These are the types of applications that display windows or forms on the desktop, enabling a user to perform a task. Client applications include applications such as word processors and spreadsheets, as well as custom business applications such as data-entry tools, reporting tools, and so on. Client applications usually employ windows, menus, buttons, and other GUI elements, and they likely access local resources such as the file system and peripherals such as printers.

Another kind of client application is the traditional ActiveX control (now replaced by the managed Windows Forms control) deployed over the Internet as a Web page. This application is much like other client applications: it is executed natively, has access to local resources, and includes graphical elements.

In the past, developers created such applications using C/C++ in conjunction with the Microsoft Foundation Classes (MFC) or with a rapid application development (RAD) environment such as Microsoft® Visual Basic®. The .NET Framework incorporates aspects of these existing products into a single, consistent development environment that drastically simplifies the development of client applications.

The Windows Forms classes contained in the .NET Framework are designed to be used for GUI development. You can easily create command windows, buttons, menus, toolbars, and other screen elements with the flexibility necessary to accommodate shifting business needs.

For example, the .NET Framework provides simple properties to adjust visual attributes associated with forms. In some cases the underlying operating system does not support changing these attributes directly, and in these cases the .NET Framework automatically recreates the forms. This is one of many ways in which the .NET Framework integrates the developer interface, making coding simpler and more consistent.

Unlike ActiveX controls, Windows Forms controls have semi-trusted access to a user's computer. This means that binary or natively executing code can access some of the resources on the user's system (such as GUI elements and limited file access) without being able to access or compromise other resources. Because of code access security, many applications that once needed to be installed on a user's system can now be safely deployed through the Web. Your applications can implement the features of a local application while being deployed like a Web page.

## 6.2 ASP.NET

### Server Application Development

Server-side applications in the managed world are implemented through runtime hosts. Unmanaged applications host the common language runtime, which allows your custom managed code to control the behavior of the server. This model provides you with all the features of the common language runtime and class library while gaining the performance and scalability of the host server.

The following illustration shows a basic network schema with managed code running in different server environments. Servers such as IIS and SQL Server can perform standard operations while your application logic executes through the managed code.

### SERVER-SIDE MANAGED CODE

ASP.NET is the hosting environment that enables developers to use the .NET Framework to target Web-based applications. However, ASP.NET is more than just a runtime host; it is a complete architecture for developing Web sites and Internet-distributed objects using managed code. Both Web Forms and XML Web services use IIS and ASP.NET as the publishing mechanism for applications, and both have a collection of supporting classes in the .NET Framework.

XML Web services, an important evolution in Web-based technology, are distributed, server-side application components similar to common Web sites. However, unlike Web-based applications, XML Web services components have no UI and are not targeted for browsers such as Internet Explorer and Netscape Navigator. Instead, XML Web services consist of reusable software components designed to be consumed by other applications, such as traditional client applications, Web-based applications, or even other XML Web services. As a result, XML Web services technology is rapidly moving application development and deployment into the highly distributed environment of the Internet.

If you have used earlier versions of ASP technology, you will immediately notice the improvements that ASP.NET and Web Forms offers. For example, you can develop Web Forms pages in any language that supports the .NET Framework. In addition, your code no longer needs to share the same file with your HTTP text (although it can continue to do so if you prefer). Web Forms pages execute in native machine language because, like any other managed application, they take full advantage of the runtime. In contrast, unmanaged ASP pages are always scripted and interpreted. ASP.NET pages are

faster, more functional, and easier to develop than unmanaged ASP pages because they interact with the runtime like any managed application.

The .NET Framework also provides a collection of classes and tools to aid in development and consumption of XML Web services applications. XML Web services are built on standards such as SOAP (a remote procedure-call protocol), XML (an extensible data format), and WSDL ( the Web Services Description Language). The .NET Framework is built on these standards to promote interoperability with non-Microsoft solutions.

For example, the Web Services Description Language tool included with the .NET Framework SDK can query an XML Web service published on the Web, parse its WSDL description, and produce C# or Visual Basic source code that your application can use to become a client of the XML Web service. The source code can create classes derived from classes in the class library that handle all the underlying communication using SOAP and XML parsing. Although you can use the class library to consume XML Web services directly, the Web Services Description Language tool and the other tools contained in the SDK facilitate your development efforts with the .NET Framework.

If you develop and publish your own XML Web service, the .NET Framework provides a set of classes that conform to all the underlying communication standards, such as SOAP, WSDL, and XML. Using those classes enables you to focus on the logic of your service, without concerning yourself with the communications infrastructure required by distributed software development.

Finally, like Web Forms pages in the managed environment, your XML Web service will run with the speed of native machine language using the scalable communication of IIS.

## ACTIVE SERVER PAGES.NET

ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. ASP.NET offers several important advantages over previous Web development models:

- **Enhanced Performance.** ASP.NET is compiled common language runtime code running on the server. Unlike its interpreted predecessors, ASP.NET can take advantage of early binding, just-in-time compilation, native optimization, and caching services right out of the box. This amounts to dramatically better performance before you ever write a line of code.

- **World-Class Tool Support.** The ASP.NET framework is complemented by a rich toolbox and designer in the Visual Studio integrated development environment. WYSIWYG editing, drag-and-drop server controls, and automatic deployment are just a few of the features this powerful tool provides.
- **Power and Flexibility.** Because ASP.NET is based on the common language runtime, the power and flexibility of that entire platform is available to Web application developers. The .NET Framework class library, Messaging, and Data Access solutions are all seamlessly accessible from the Web. ASP.NET is also language-independent, so you can choose the language that best applies to your application or partition your application across many languages. Further, common language runtime interoperability guarantees that your existing investment in COM-based development is preserved when migrating to ASP.NET.
- **Simplicity.** ASP.NET makes it easy to perform common tasks, from simple form submission and client authentication to deployment and site configuration. For example, the ASP.NET page framework allows you to build user interfaces that cleanly separate application logic from presentation code and to handle events in a simple, Visual Basic - like forms processing model. Additionally, the common language runtime simplifies development, with managed code services such as automatic reference counting and garbage collection.
- **Manageability.** ASP.NET employs a text-based, hierarchical configuration system, which simplifies applying settings to your server environment and Web applications. Because configuration information is stored as plain text, new settings may be applied without the aid of local administration tools. This "zero local administration" philosophy extends to deploying ASP.NET Framework applications as well. An ASP.NET Framework application is deployed to a server simply by copying the necessary files to the server. No server restart is required, even to deploy or replace running compiled code.
- **Scalability and Availability.** ASP.NET has been designed with scalability in mind, with features specifically tailored to improve performance in clustered and multiprocessor environments. Further, processes are closely monitored and managed by the ASP.NET runtime, so that if one misbehaves (leaks, deadlocks), a new process can be created in its place, which helps keep your application constantly available to handle requests.
- **Customizability and Extensibility.** ASP.NET delivers a well-factored architecture that allows developers to "plug-in" their code at the appropriate level. In fact, it is possible to extend or replace

any subcomponent of the ASP.NET runtime with your own custom-written component. Implementing custom authentication or state services has never been easier.

- **Security.** With built in Windows authentication and per-application configuration, you can be assured that your applications are secure.

## LANGUAGE SUPPORT

The Microsoft .NET Platform currently offers built-in support for three languages: C#, Visual Basic, and JScript.

## WHAT IS ASP.NET WEB FORMS?

The ASP.NET Web Forms page framework is a scalable common language runtime programming model that can be used on the server to dynamically generate Web pages.

Intended as a logical evolution of ASP (ASP.NET provides syntax compatibility with existing pages), the ASP.NET Web Forms framework has been specifically designed to address a number of key deficiencies in the previous model. In particular, it provides:

- The ability to create and use reusable UI controls that can encapsulate common functionality and thus reduce the amount of code that a page developer has to write.
- The ability for developers to cleanly structure their page logic in an orderly fashion (not "spaghetti code").
- The ability for development tools to provide strong WYSIWYG design support for pages (existing ASP code is opaque to tools).

ASP.NET Web Forms pages are text files with an .aspx file name extension. They can be deployed throughout an IIS virtual root directory tree. When a browser client requests .aspx resources, the ASP.NET runtime parses and compiles the target file into a .NET Framework class. This class can then be used to dynamically process incoming requests. (Note that the .aspx file is compiled only the first time it is accessed; the compiled type instance is then reused across multiple requests).

An ASP.NET page can be created simply by taking an existing HTML file and changing its file name extension to .aspx (no modification of code is required). For example, the following sample demonstrates a simple HTML page that collects a user's name and category preference and then performs a form postback to the originating page when a button is clicked:

ASP.NET provides syntax compatibility with existing ASP pages. This includes support for <% %> code render blocks that can be intermixed with HTML content within an .aspx file. These code blocks execute in a top-down manner at page render time.

## CODE-BEHIND WEB FORMS

ASP.NET supports two methods of authoring dynamic pages. The first is the method shown in the preceding samples, where the page code is physically declared within the originating .aspx file. An alternative approach--known as the code-behind method--enables the page code to be more cleanly separated from the HTML content into an entirely separate file.

## INTRODUCTION TO ASP.NET SERVER CONTROLS

In addition to (or instead of) using <% %> code blocks to program dynamic content, ASP.NET page developers can use ASP.NET server controls to program Web pages. Server controls are declared within an .aspx file using custom tags or intrinsic HTML tags that contain a **runat="server"** attributes value. Intrinsic HTML tags are handled by one of the controls in the **System.Web.UI.HtmlControls** namespace. Any tag that doesn't explicitly map to one of the controls is assigned the type of **System.Web.UI.HtmlControls.HtmlGenericControl**.

Server controls automatically maintain any client-entered values between round trips to the server. This control state is not stored on the server (it is instead stored within an **<input type="hidden">** form field that is round-tripped between requests). Note also that no client-side script is required.

In addition to supporting standard HTML input controls, ASP.NET enables developers to utilize richer custom controls on their pages. For example, the following sample demonstrates how the **<asp:adrotator>** control can be used to dynamically display rotating ads on a page.

1. ASP.NET Web Forms provide an easy and powerful way to build dynamic Web UI.
2. ASP.NET Web Forms pages can target any browser client (there are no script library or cookie requirements).
3. ASP.NET Web Forms pages provide syntax compatibility with existing ASP pages.
4. ASP.NET server controls provide an easy way to encapsulate common functionality.
5. ASP.NET ships with 45 built-in server controls. Developers can also use controls built by third parties.
6. ASP.NET server controls can automatically project both uplevel and downlevel HTML.

7. ASP.NET templates provide an easy way to customize the look and feel of list server controls.
8. ASP.NET validation controls provide an easy way to do declarative client or server data validation.

## 6.3.C#.NET

### ADO.NET OVERVIEW

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.
ADO.NET uses some ADO objects, such as the **Connection** and **Command** objects, and also introduces new objects. Key new ADO.NET objects include the **DataSet**, **DataReader**, and **DataAdapter**.

The important distinction between this evolved stage of ADO.NET and previous data architectures is that there exists an object -- the **DataSet** -- that is separate and distinct from any data stores. Because of that, the **DataSet** functions as a standalone entity. You can think of the DataSet as an always disconnected recordset that knows nothing about the source or destination of the data it contains. Inside a **DataSet**, much like in a database, there are tables, columns, relationships, constraints, views, and so forth.

A **DataAdapter** is the object that connects to the database to fill the **DataSet**. Then, it connects back to the database to update the data there, based on operations performed while the **DataSet** held the data. In the past, data processing has been primarily connection-based. Now, in an effort to make multi-tiered apps more efficient, data processing is turning to a message-based approach that revolves around chunks of information. At the center of this approach is the **DataAdapter**, which provides a bridge to retrieve and save data between a **DataSet** and its source data store. It accomplishes this by means of requests to the appropriate SQL commands made against the data store.

The XML-based **DataSet** object provides a consistent programming model that works with all models of data storage: flat, relational, and hierarchical. It does this by having no 'knowledge' of the source of its data, and by representing the data that it holds as collections and data types. No matter what

the source of the data within the **DataSet** is, it is manipulated through the same set of standard APIs exposed through the **DataSet** and its subordinate objects.

While the **DataSet** has no knowledge of the source of its data, the managed provider has detailed and specific information. The role of the managed provider is to connect, fill, and persist the **DataSet** to and from data stores. The OLE DB and SQL Server .NET Data Providers (System.Data.OleDb and System.Data.SqlClient) that are part of the .Net Framework provide four basic objects: the **Command**, **Connection**, **DataReader** and **DataAdapter**. In the remaining sections of this document, we'll walk through each part of the **DataSet** and the OLE DB/SQL Server .NET Data Providers explaining what they are, and how to program against them.

The following sections will introduce you to some objects that have evolved, and some that are new. These objects are:

- **Connections**. For connection to and managing transactions against a database.
- **Commands**. For issuing SQL commands against a database.
- **DataReaders**. For reading a forward-only stream of data records from a SQL Server data source.
- **DataSets**. For storing, Remoting and programming against flat data, XML data and relational data.
- **DataAdapters**. For pushing data into a **DataSet**, and reconciling data against a database.

When dealing with connections to a database, there are two different options: SQL Server .NET Data Provider (System.Data.SqlClient) and OLE DB .NET Data Provider (System.Data.OleDb). In these samples we will use the SQL Server .NET Data Provider. These are written to talk directly to Microsoft SQL Server. The OLE DB .NET Data Provider is used to talk to any OLE DB provider (as it uses OLE DB underneath).

## Connections:

Connections are used to 'talk to' databases, and are represented by provider-specific classes such as **SqlConnection**. Commands travel over connections and resultsets are returned in the form of streams which can be read by a **DataReader** object, or pushed into a **DataSet** object.

## Commands:

Commands contain the information that is submitted to a database, and are represented by provider-specific classes such as **SqlCommand**. A command can be a stored procedure call, an UPDATE statement, or a statement that returns results. You can also use input and output parameters, and return

values as part of your command syntax. The example below shows how to issue an INSERT statement against the **Northwind** database.

### DataReaders:

The **DataReader** object is somewhat synonymous with a read-only/forward-only cursor over data. The **DataReader** API supports flat as well as hierarchical data. A **DataReader** object is returned after executing a command against a database. The format of the returned **DataReader** object is different from a recordset. For example, you might use the **DataReader** to show the results of a search list in a web page.

## DATASETS AND DATAADAPTERS:

### DataSets

The **DataSet** object is similar to the ADO **Recordset** object, but more powerful, and with one other important distinction: the **DataSet** is always disconnected. The **DataSet** object represents a cache of data, with database-like structures such as tables, columns, relationships, and constraints. However, though a **DataSet** can and does behave much like a database, it is important to remember that **DataSet** objects do not interact directly with databases, or other source data. This allows the developer to work with a programming model that is always consistent, regardless of where the source data resides. Data coming from a database, an XML file, from code, or user input can all be placed into **DataSet** objects. Then, as changes are made to the **DataSet** they can be tracked and verified before updating the source data. The **GetChanges** method of the **DataSet** object actually creates a second **DatSet** that contains only the changes to the data. This **DataSet** is then used by a **DataAdapter** (or other objects) to update the original data source.

The **DataSet** has many XML characteristics, including the ability to produce and consume XML data and XML schemas. XML schemas can be used to describe schemas interchanged via WebServices. In fact, a **DataSet** with a schema can actually be compiled for type safety and statement completion.

## DATAADAPTERS (OLEDB/SQL)

The **DataAdapter** object works as a bridge between the **DataSet** and the source data. Using the provider-specific **SqlDataAdapter** (along with its associated **SqlCommand** and **SqlConnection**) can increase overall performance when working with a Microsoft SQL Server databases. For other OLE

DB-supported databases, you would use the **OleDbDataAdapter** object and its associated **OleDbCommand** and **OleDbConnection** objects.

The **DataAdapter** object uses commands to update the data source after changes have been made to the **DataSet**. Using the **Fill** method of the **DataAdapter** calls the SELECT command; using the **Update** method calls the INSERT, UPDATE or DELETE command for each changed row. You can explicitly set these commands in order to control the statements used at runtime to resolve changes, including the use of stored procedures. For ad-hoc scenarios, a **CommandBuilder** object can generate these at run-time based upon a select statement. However, this run-time generation requires an extra round-trip to the server in order to gather required metadata, so explicitly providing the INSERT, UPDATE, and DELETE commands at design time will result in better run-time performance.

1. ADO.NET is the next evolution of ADO for the .Net Framework.
2. ADO.NET was created with n-Tier, statelessness and XML in the forefront. Two new objects, the **DataSet** and **DataAdapter**, are provided for these scenarios.
3. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.
4. There is a lot more information about ADO.NET in the documentation.
5. Remember, you can execute a command directly against the database in order to do inserts, updates, and deletes. You don't need to first put data into a **DataSet** in order to insert, update, or delete it.
6. Also, you can use a **DataSet** to bind to the data, move through the data, and navigate data relationships

## 6.3 SQL SERVER

A database management, or DBMS, gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, SQL Server and SQL Server. These systems allow users to create, update and extract information from their database.

A database is a structured collection of data. Data refers to the characteristics of people, things and events. SQL Server stores each data item in its own fields. In SQL Server, the fields relating to a particular person, thing or event are bundled together to form a single complete unit of data, called a record (it can also be referred to as raw or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name.

During an SQL Server Database design project, the analysis of your business needs identifies all the fields or attributes of interest. If your business needs change over time, you define any additional fields or change the definition of existing fields.

## SQL SERVER TABLES

SQL Server stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

## PRIMARY KEY

Every table in SQL Server has a field or a combination of fields that uniquely identifies each record in the table. The Unique identifier is called the Primary Key, or simply the Key. The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identify, locate and refer to one particular record in the database.

## RELATIONAL DATABASE

Sometimes all the information of interest to a business operation can be stored in one table. SQL Server makes it very easy to link the data in multiple tables. Matching an employee to the department in which they work is one example. This is what makes SQL Server a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the table and enables you to define relationships between the tables.

## FOREIGN KEY

When a field is one table matches the primary key of another field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values match those of the primary key of another table.

## REFERENTIAL INTEGRITY

Not only does SQL Server allow you to link multiple tables, it also maintains consistency between them. Ensuring that the data among related tables is correctly matched is referred to as maintaining referential integrity.

## DATA ABSTRACTION

A major purpose of a database system is to provide users with an abstract view of the data. This system hides certain details of how the data is stored and maintained. Data abstraction is divided into three levels.

**Physical level**: This is the lowest level of abstraction at which one describes how the data are actually stored.

**Conceptual Level**: At this level of database abstraction all the attributed and what data are actually stored is described and entries and relationship among them.

**View level**: This is the highest level of abstraction at which one describes only part of the database.

## ADVANTAGES OF RDBMS

- Redundancy can be avoided
- Inconsistency can be eliminated
- Data can be Shared
- Standards can be enforced
- Security restrictions ca be applied
- Integrity can be maintained
- Conflicting requirements can be balanced
- Data independence can be achieved.

## DISADVANTAGES OF DBMS

A significant disadvantage of the DBMS system is cost. In addition to the cost of purchasing of developing the software, the hardware has to be upgraded to allow for the extensive programs and the workspace required for their execution and storage. While centralization reduces duplication, the lack of duplication requires that the database be adequately backed up so that in case of failure the data can be recovered.

## FEATURES OF SQL SERVER (RDBMS)

SQL SERVER is one of the leading database management systems (DBMS) because it is the only Database that meets the uncompromising requirements of today's most demanding information systems. From complex decision support systems (DSS) to the most rigorous online transaction processing (OLTP) application, even application that require simultaneous DSS and OLTP access to the same critical data, SQL Server leads the industry in both performance and capability

SQL SERVER is a truly portable, distributed, and open DBMS that delivers unmatched performance, continuous operation and support for every database.
SQL SERVER RDBMS is high performance fault tolerant DBMS which is specially designed for online transactions processing and for handling large database application.

SQL SERVER with transactions processing option offers two features which contribute to very high level of transaction processing throughput, which are

- The row level lock manager

## ENTERPRISE WIDE DATA SHARING

The unrivaled portability and connectivity of the SQL SERVER DBMS enables all the systems in the organization to be linked into a singular, integrated computing resource.

## PORTABILITY

SQL SERVER is fully portable to more than 80 distinct hardware and operating systems platforms, including UNIX, MSDOS, OS/2, Macintosh and dozens of proprietary platforms. This portability gives complete freedom to choose the database sever platform that meets the system requirements.

## OPEN SYSTEMS

SQL SERVER offers a leading implementation of industry –standard SQL. SQL Server's open architecture integrates SQL SERVER and non –SQL SERVER DBMS with industries most comprehensive collection of tools, application, and third party software products SQL Server's Open architecture provides transparent access to data from other relational database and even non-relational database.

## DISTRIBUTED DATA SHARING

SQL Server's networking and distributed database capabilities to access data stored on remote server with the same ease as if the information was stored on a single local computer. A single SQL statement can access data at multiple sites. You can store data where system requirements such as performance, security or availability dictate.

## UNMATCHED PERFORMANCE

The most advanced architecture in the industry allows the SQL SERVER DBMS to deliver unmatched performance.

## SOPHISTICATED CONCURRENCY CONTROL

Real World applications demand access to critical data. With most database Systems application becomes "contention bound" – which performance is limited not by the CPU power or by disk I/O, but user waiting on one another for data access . SQL Server employs full, unrestricted row-level locking and contention free queries to minimize and in many cases entirely eliminates contention wait times.

## NO I/O BOTTLENECKS

SQL Server's fast commit groups commit and deferred write technologies dramatically reduce disk I/O bottlenecks. While some database write whole data block to disk at commit time, SQL Server commits transactions with at most sequential log file on disk at commit time, On high throughput systems, one sequential writes typically group commit multiple transactions. Data read by the transaction remains as shared memory so that other transactions may access that data without reading it again from disk. Since fast commits write all data necessary to the recovery to the log file, modified blocks are written back to the database independently of the transaction commit, when written from memory to disk.

# CHAPTER 7

# 7.SYSTEM TESTING AND IMPLEMENTATION

## 7.1. INTRODUCTION

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

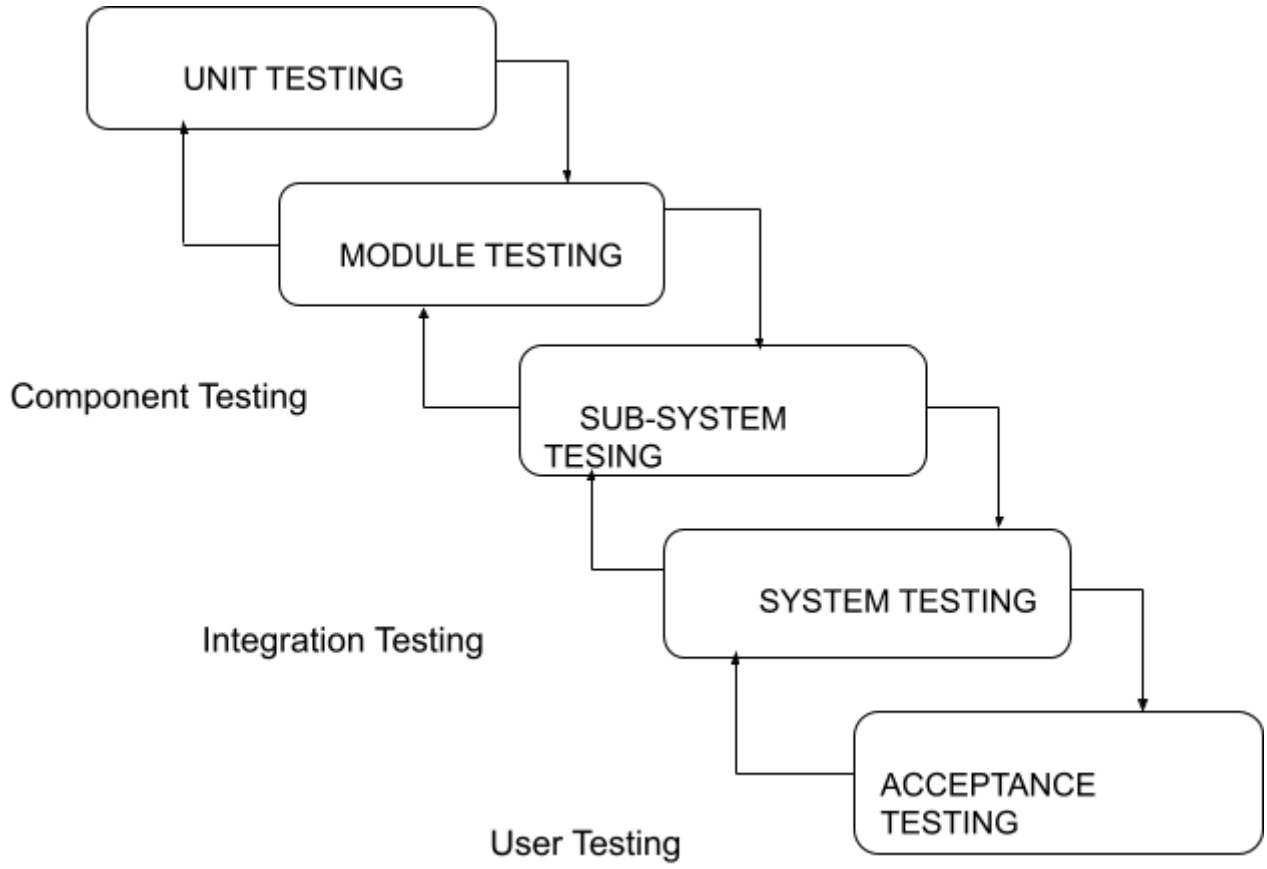
A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

## 7.2. SOFTWARE TESTING

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress by moving outward along the spiral to integration testing, where the focus is on the

design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.




## 7.3. Unit Testing

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

### 1. WHITE BOX TESTING

This type of testing ensures that

- All independent paths have been exercised at least once
- All logical decisions have been exercised on their true and false sides
- All loops are executed at their boundaries and within their operational bounds
- All internal data structures have been exercised to assure their validity.

To follow the concept of white box testing we have tested each form .we have created independently to verify that Data flow is correct, All conditions are exercised to check their validity, All loops are executed on their boundaries.

## 2. BASIC PATH TESTING

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:

Use the design of the code and draw correspondent flow graph.
Determine the Cyclomatic complexity of resultant flow graph, using formula:
V(G)=E-N+2 or
V(G)=P+1 or
V(G)=Number Of Regions
Where V(G) is Cyclomatic complexity,
E is the number of edges,
N is the number of flow graph nodes,
P is the number of predicate nodes.
Determine the basis of set of linearly independent paths.

## 3. CONDITIONAL TESTING

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

## 4. DATA FLOW TESTING

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The *definition-use chain* method was used in this type of testing. These were particularly useful in nested statements.

## 5. LOOP TESTING

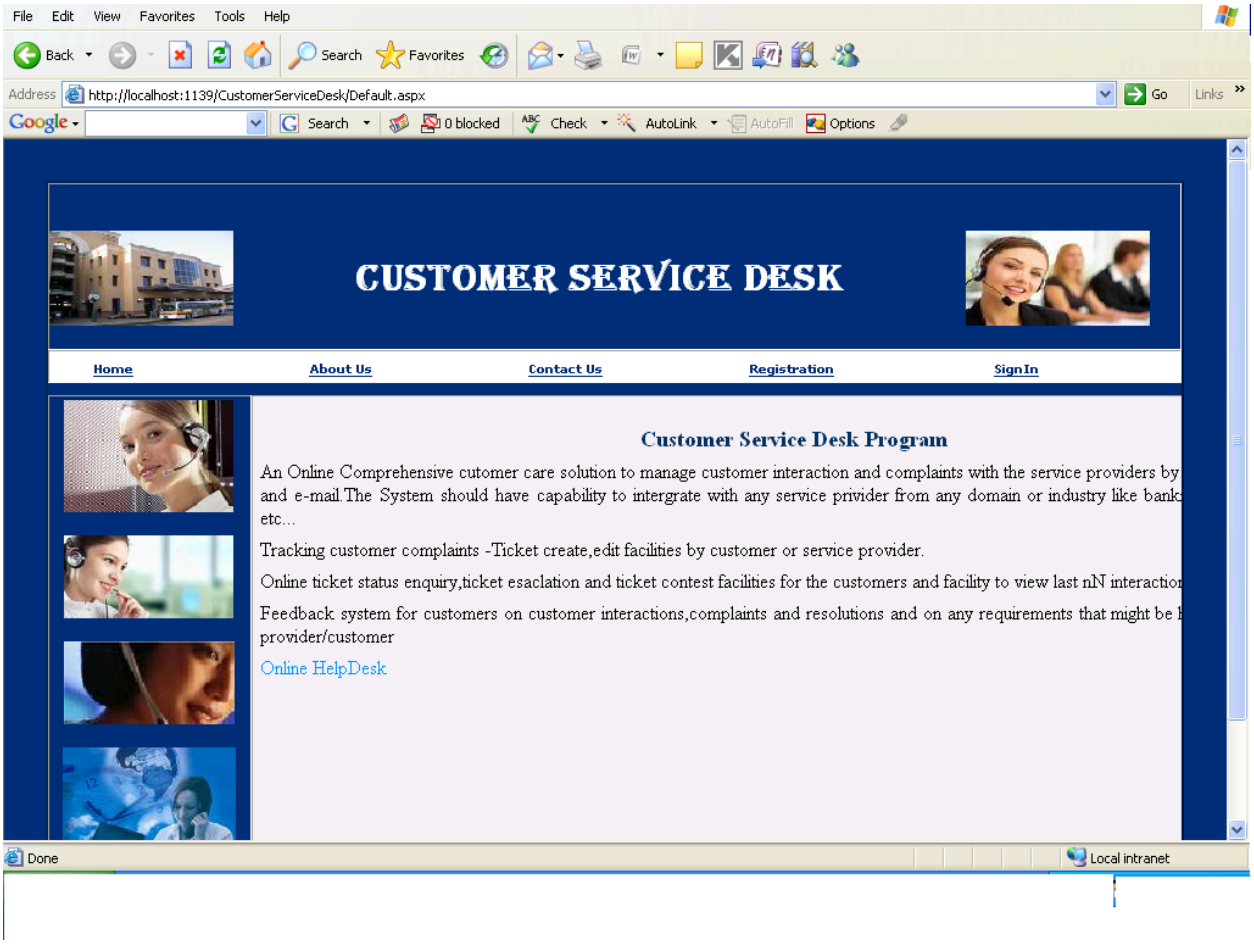
In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

- All the loops were tested at their limits, just above them and just below them.
- All the loops were skipped at least once.
- For nested loops test the inner most loop first and then work outwards.
- For concatenated loops the values of dependent loops were set with the help of connected loop.
- Unstructured loops were resolved into nested loops or concatenated loops and tested as above.

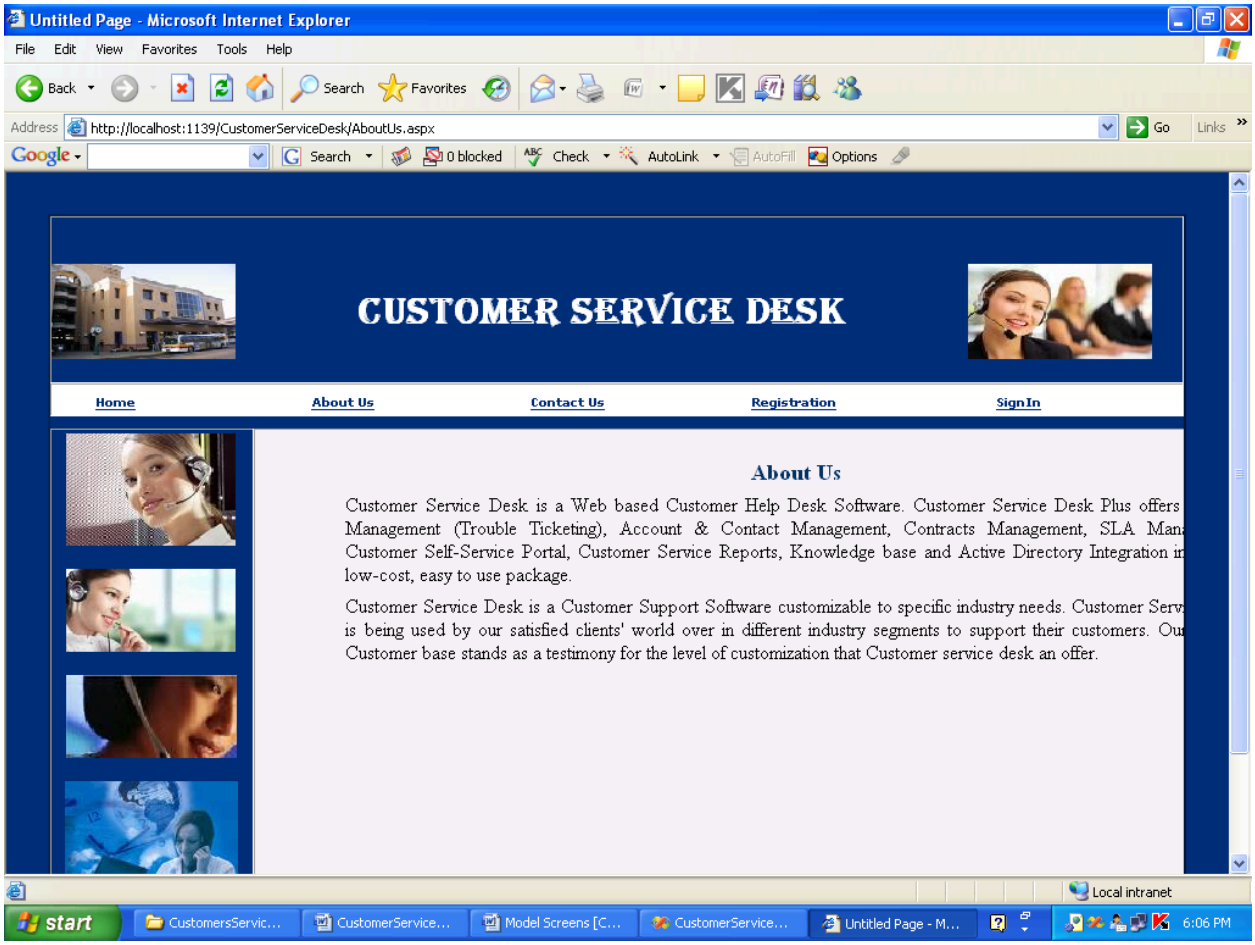
Each unit has been separately tested by the development team itself and all the input have been validated.

# CHAPTER 8

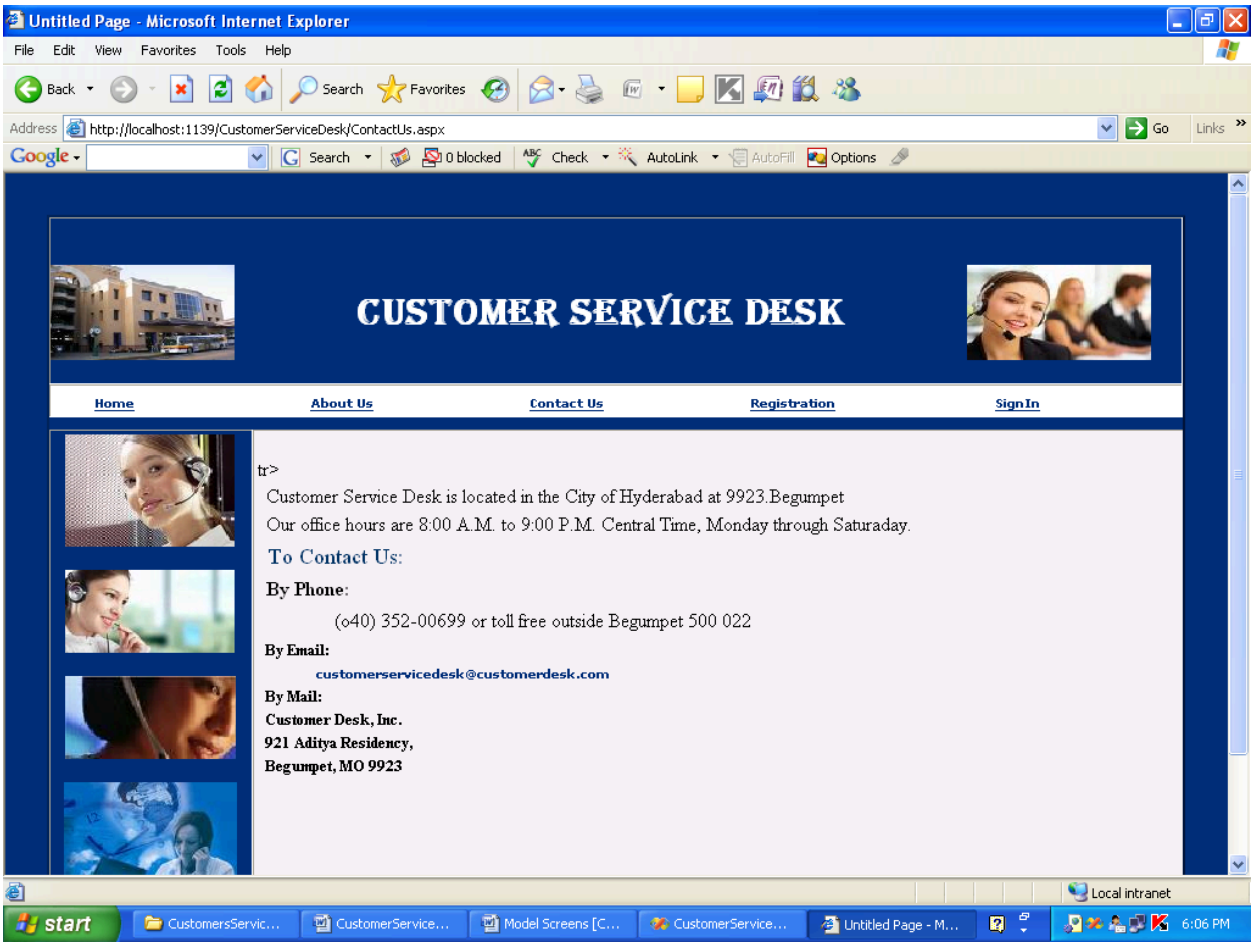
**WELCOME FOR PROJECT:**


File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:1139/CustomerServiceDesk/Default.aspx
Go Links
Google Search 0 blocked Check AutoLink AutoFill Options
CUSTOMER SERVICE DESK
Home
About Us
Contact Us
Registration
SignIn
Customer Service Desk Program
An Online Comprehensive cutomer care solution to manage customer interaction and complaints with the service providers by
and e-mail.The System should have capability to intergrate with any service privider from any domain or industry like bank
etc...
Tracking customer complaints -Ticket create,edit facilities by customer or service provider.
Online ticket status enquiry,ticket esaclation and ticket contest facilities for the customers and facility to view last nN interactio
Feedback system for customers on customer interactions,complaints and resolutions and on any requirements that might be
provider/customer
Online HelpDesk
Done
Local intranet

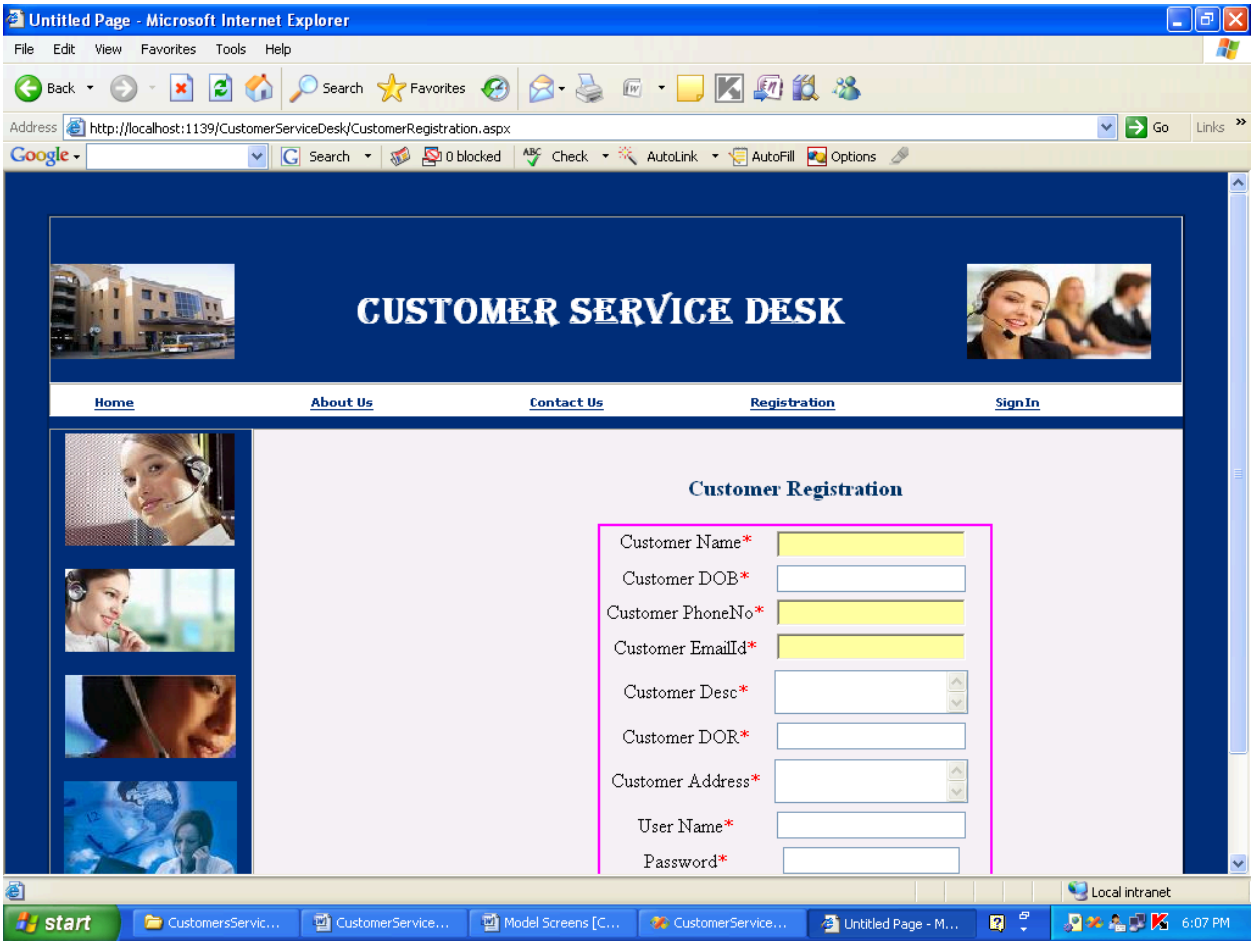
## About us:


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:1139/CustomerServiceDesk/AboutUs.aspx
Go Links
Google Search 0 blocked Check AutoLink AutoFill Options
CUSTOMER SERVICE DESK
Home
About Us
Contact Us
Registration
SignIn
About Us
Customer Service Desk is a Web based Customer Help Desk Software. Customer Service Desk Plus offers
Management (Trouble Ticketing), Account & Contact Management, Contracts Management, SLA Mana
Customer Self-Service Portal, Customer Service Reports, Knowledge base and Active Directory Integration in
low-cost, easy to use package.
Customer Service Desk is a Customer Support Software customizable to specific industry needs. Customer Servi
is being used by our satisfied clients' world over in different industry segments to support their customers. Our
Customer base stands as a testimony for the level of customization that Customer service desk an offer.
Local intranet
start
CustomersServic...
CustomerService...
Model Screens [C...
CustomerService...
Untitled Page - M...
6:06 PM

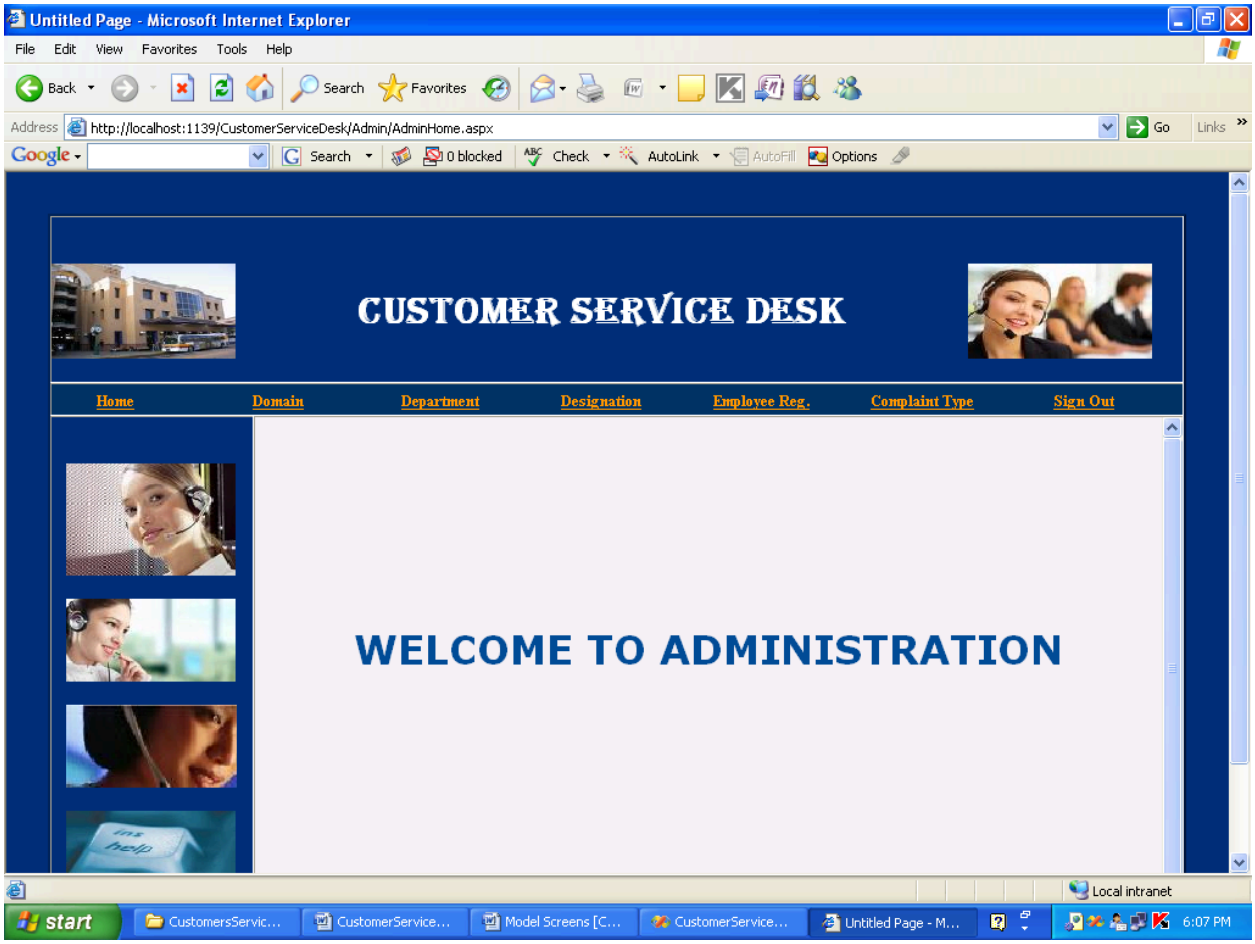
## CONTACTUS:


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:1139/CustomerServiceDesk/ContactUs.aspx
Go Links
Google Search 0 blocked Check AutoLink AutoFill Options
CUSTOMER SERVICE DESK
Home
About Us
Contact Us
Registration
SignIn
tr>
Customer Service Desk is located in the City of Hyderabad at 9923.Begumpet
Our office hours are 8:00 A.M. to 9:00 P.M. Central Time, Monday through Saturaday.
To Contact Us:
By Phone:
(o40) 352-00699 or toll free outside Begumpet 500 022
By Email:
customerservicedesk@customerdesk.com
By Mail:
Customer Desk, Inc.
921 Aditya Residency,
Begumpet, MO 9923
Local intranet
start
CustomersServic...
CustomerService...
Model Screens [C...
CustomerService...
Untitled Page - M...
6:06 PM

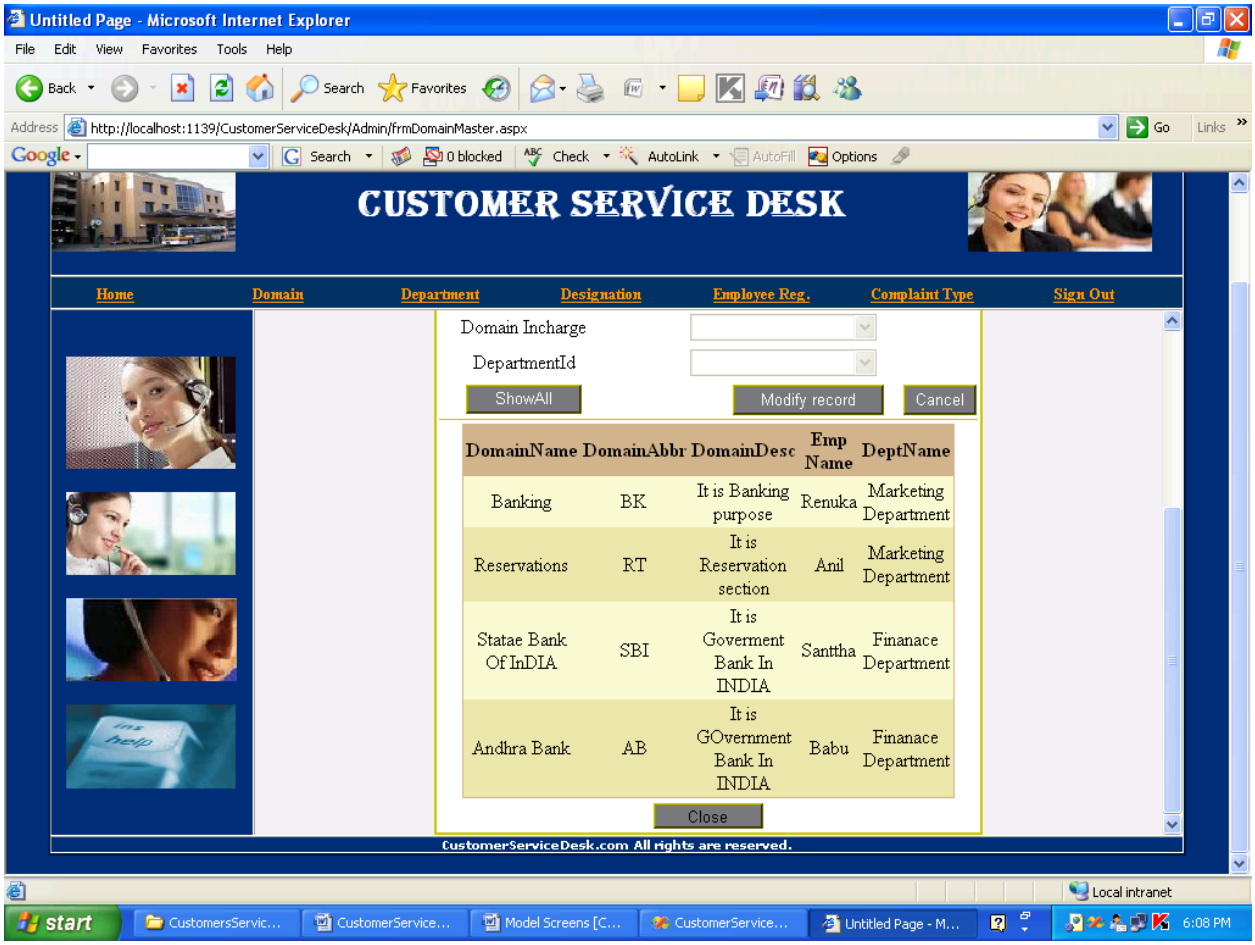
## Customer Registration:



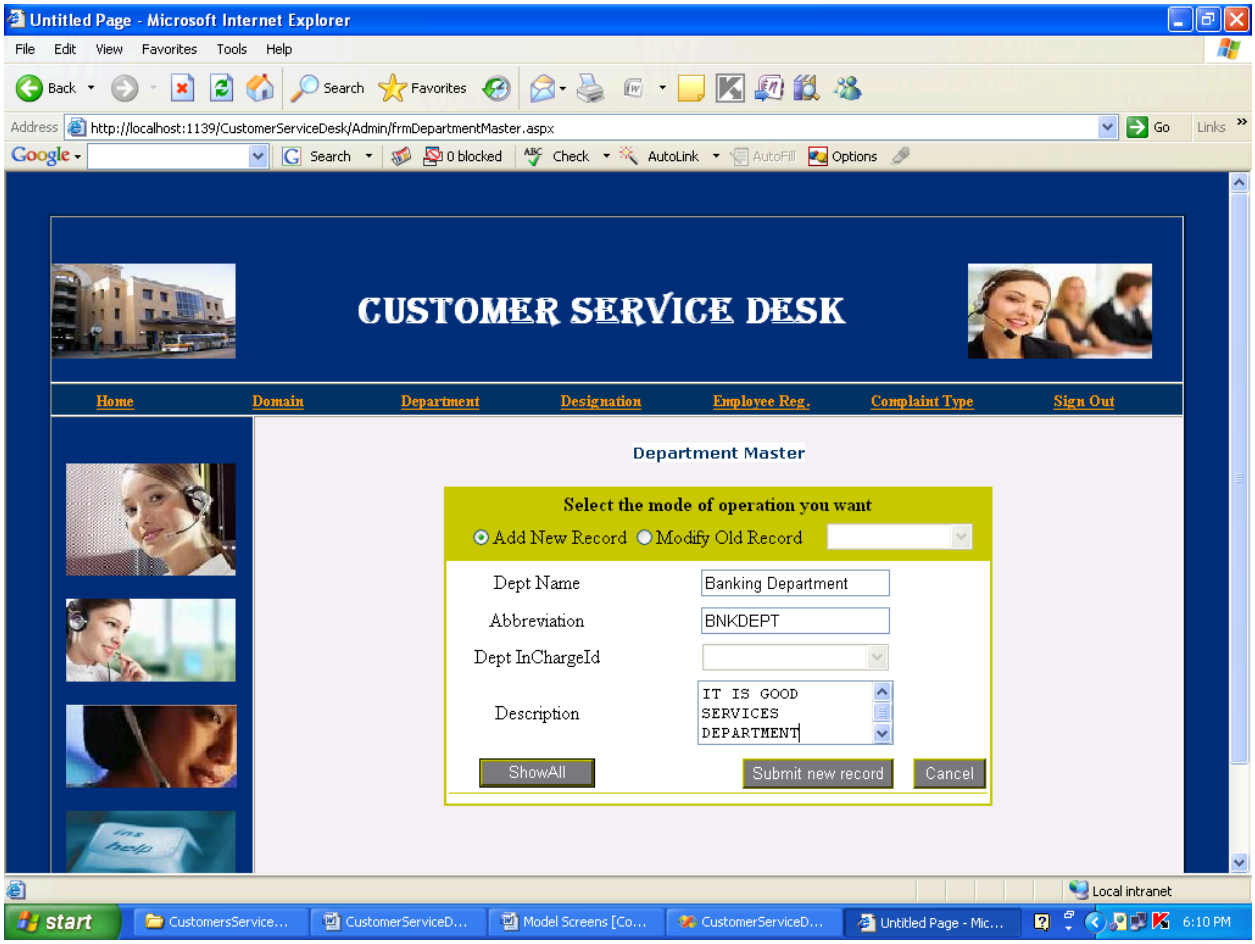
**ADMIN HOME PAGE:**


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:1139/CustomerServiceDesk/Admin/AdminHome.aspx
Go Links
Google Search 0 blocked Check AutoLink AutoFill Options
CUSTOMER SERVICE DESK
Home
Domain
Department
Designation
Employee Reg.
Complaint Type
Sign Out
WELCOME TO ADMINISTRATION
Local intranet
start
CustomersServic...
CustomerService...
Model Screens [C...
CustomerService...
Untitled Page - M...
6:07 PM

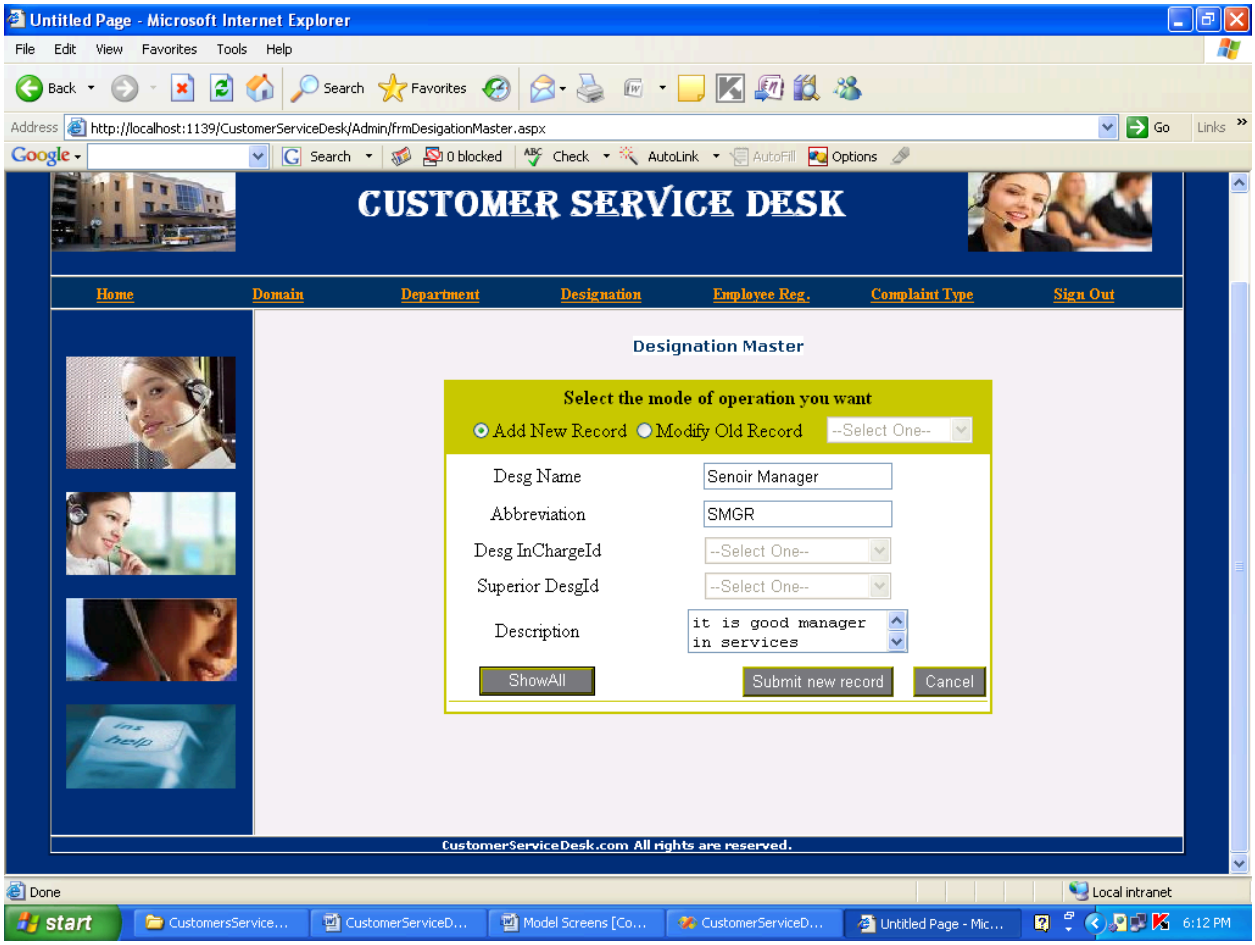
**DOMAIN PAGE:**



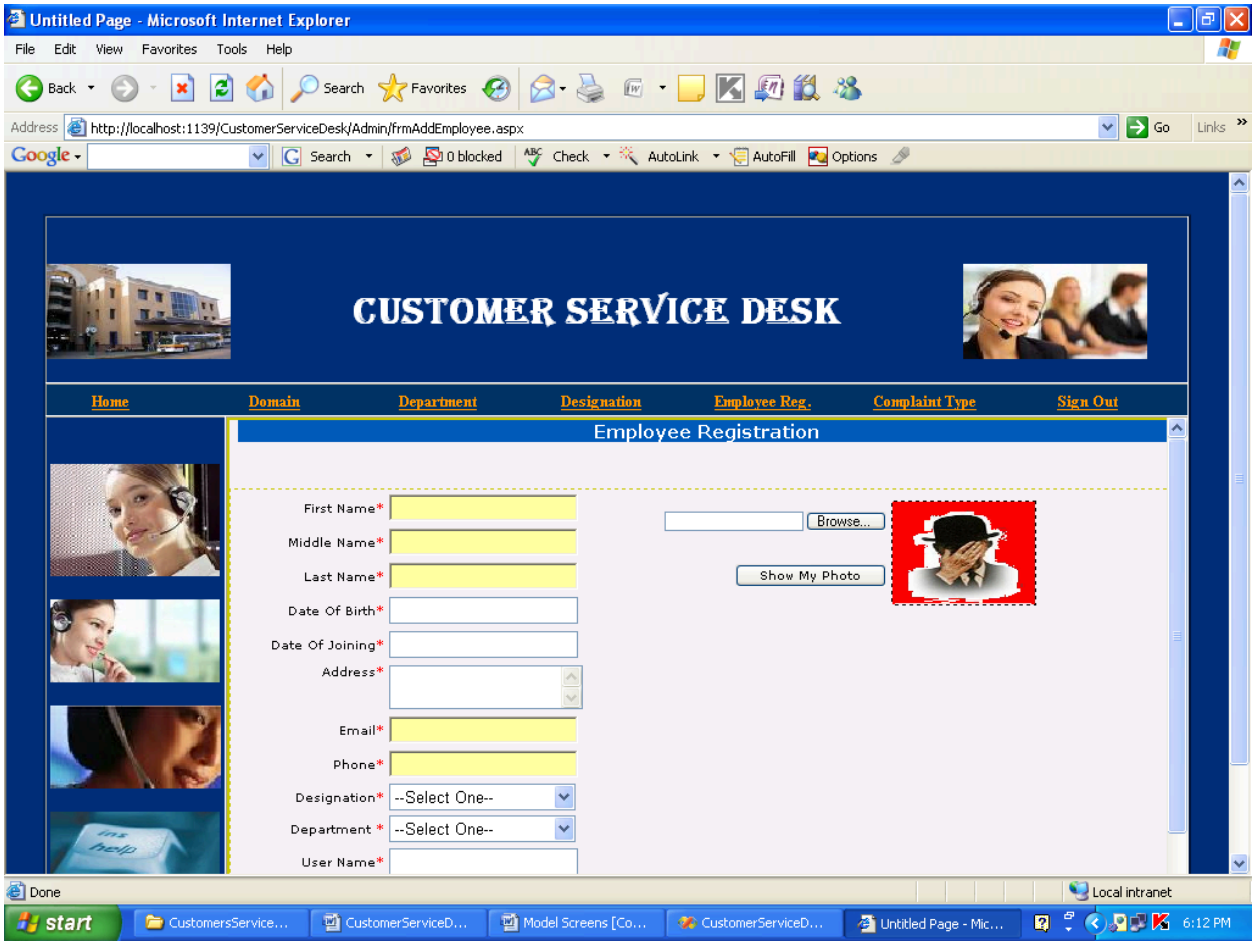

## DEPARTMENT PAGE:


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:1139/CustomerServiceDesk/Admin/frmDepartmentMaster.aspx
Go Links
Google Search 0 blocked Check AutoLink AutoFill Options
CUSTOMER SERVICE DESK
Home Domain Department Designation Employee Reg. Complaint Type Sign Out
Department Master
Select the mode of operation you want
Add New Record Modify Old Record
Dept Name
Banking Department
Abbreviation
BNKDEPT
Dept InChargeId
Description
IT IS GOOD
SERVICES
DEPARTMENT
ShowAll
Submit new record
Cancel
Local intranet
start
CustomersService...
CustomerServiceD...
Model Screens [Co...
CustomerServiceD...
Untitled Page - Mic...
6:10 PM

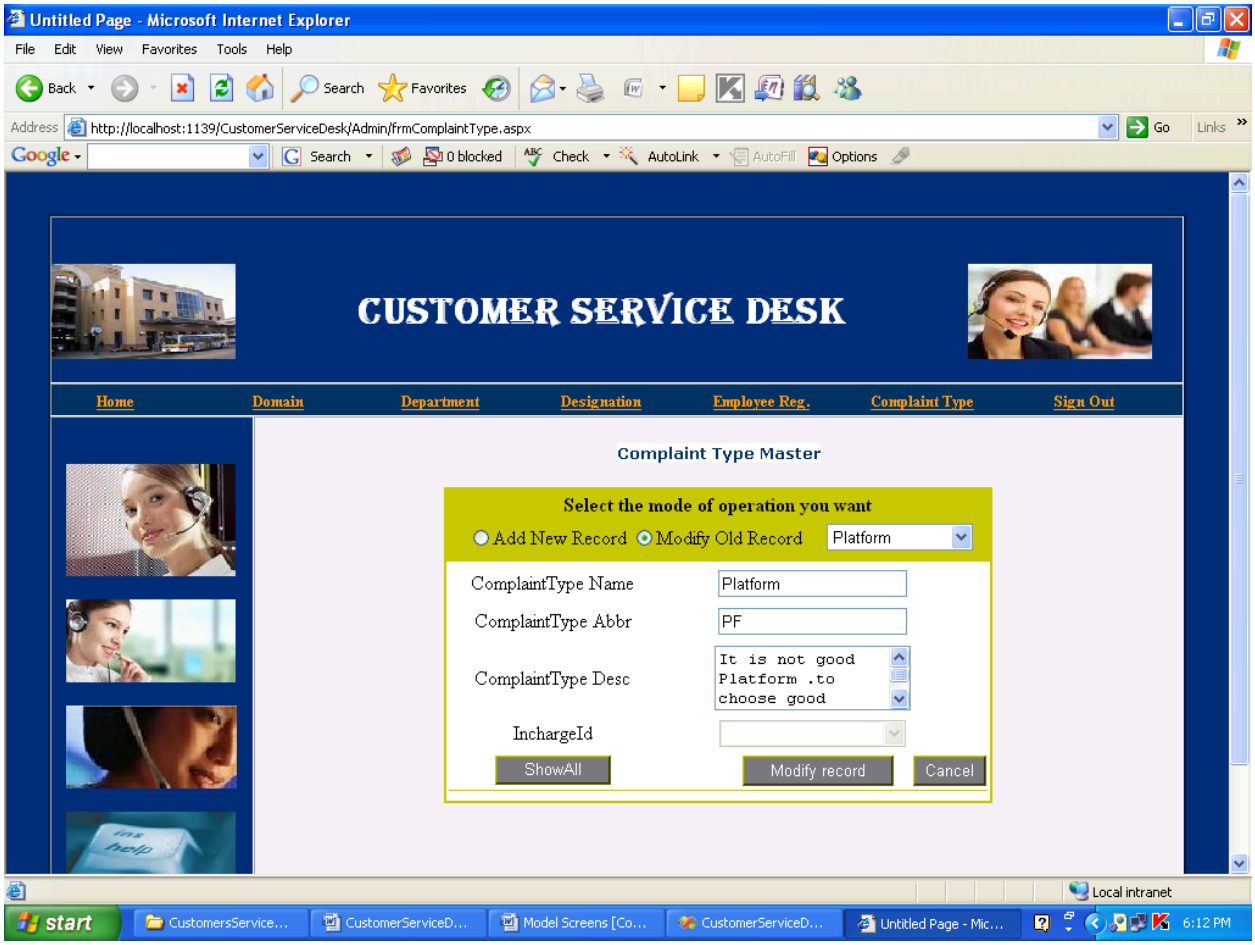
## DESIGNATION MASTER PAGE:



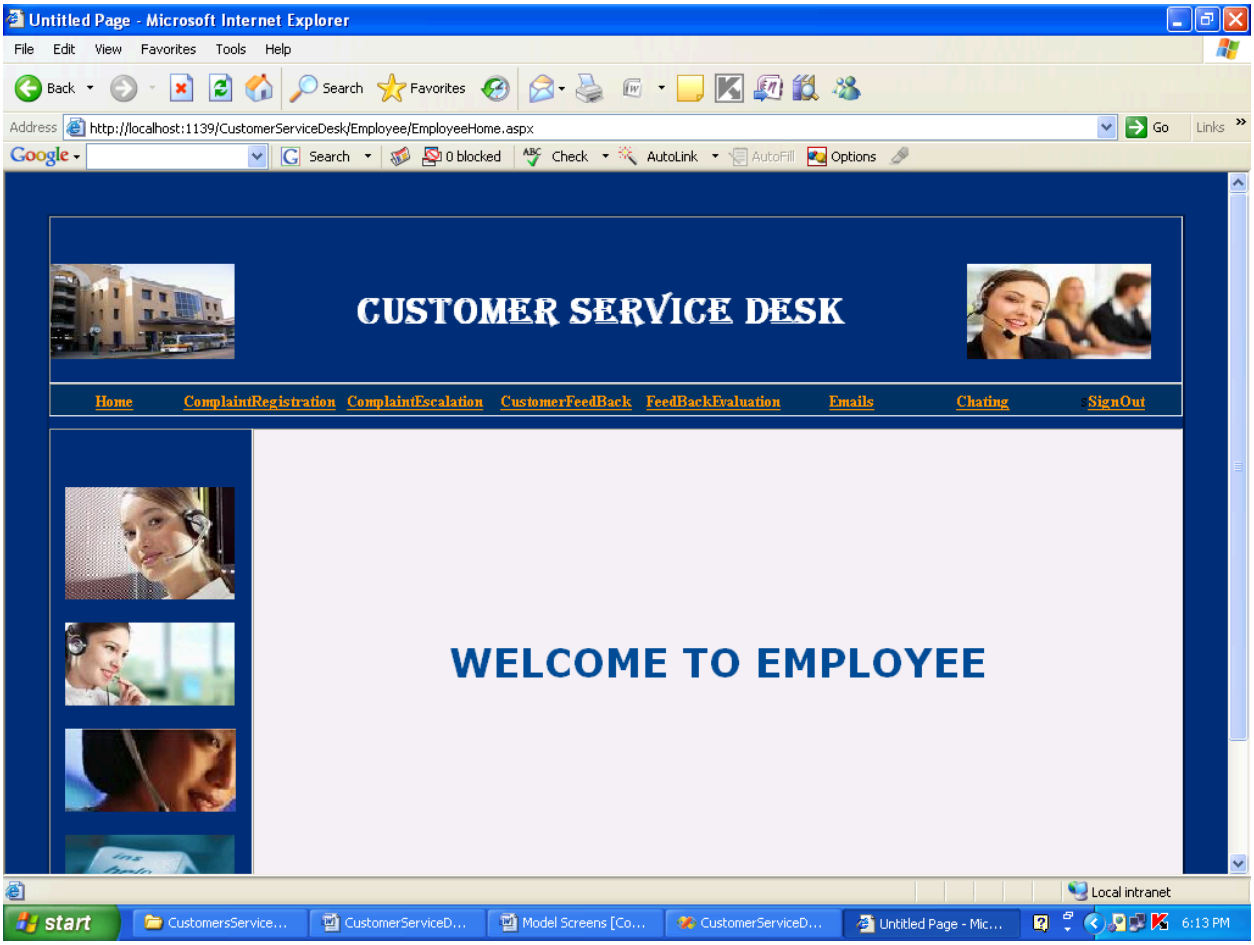
# EMPLOYEE REGISTRATION:



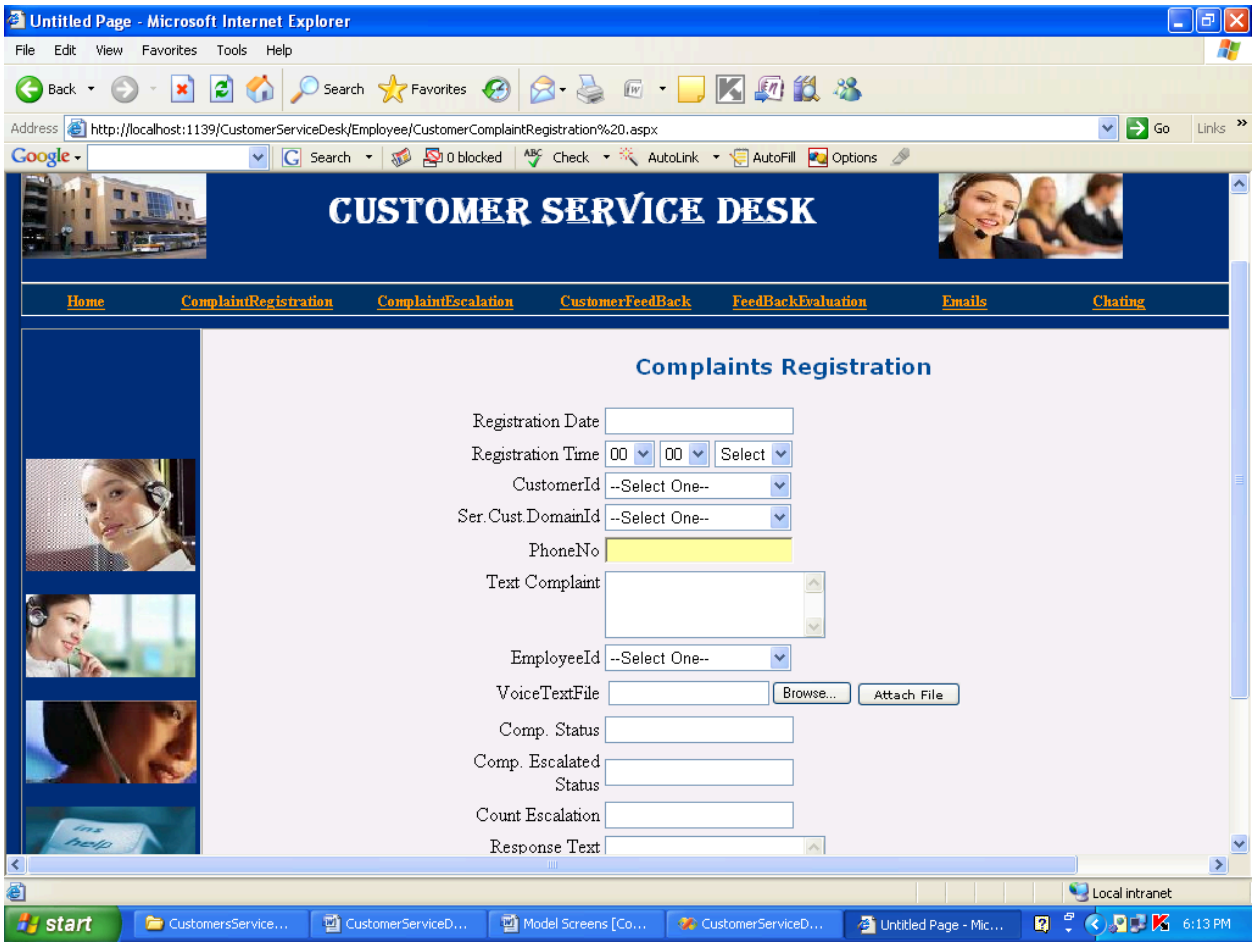
## COMPLAINT TYPE MASTER:



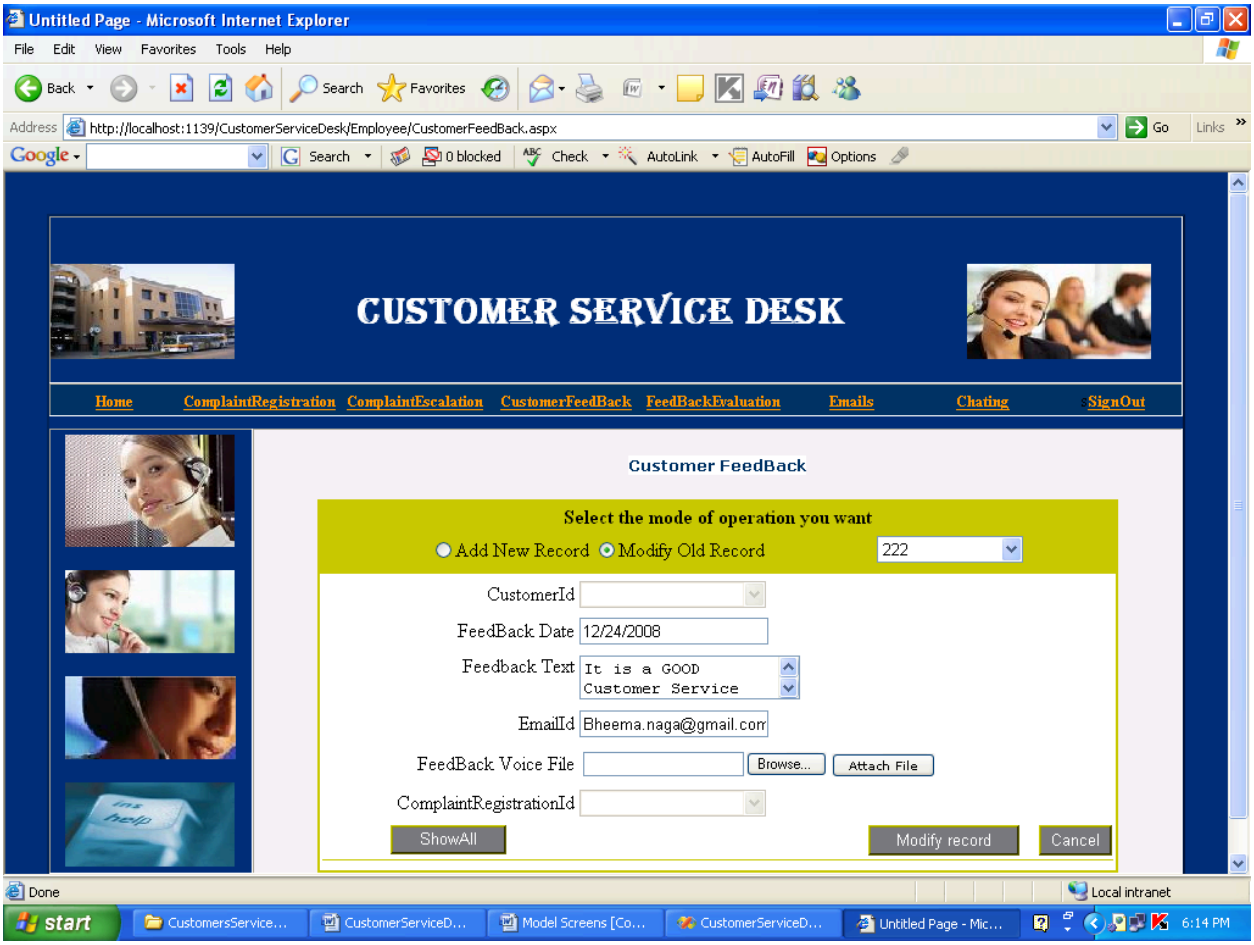
## Employee Home page:



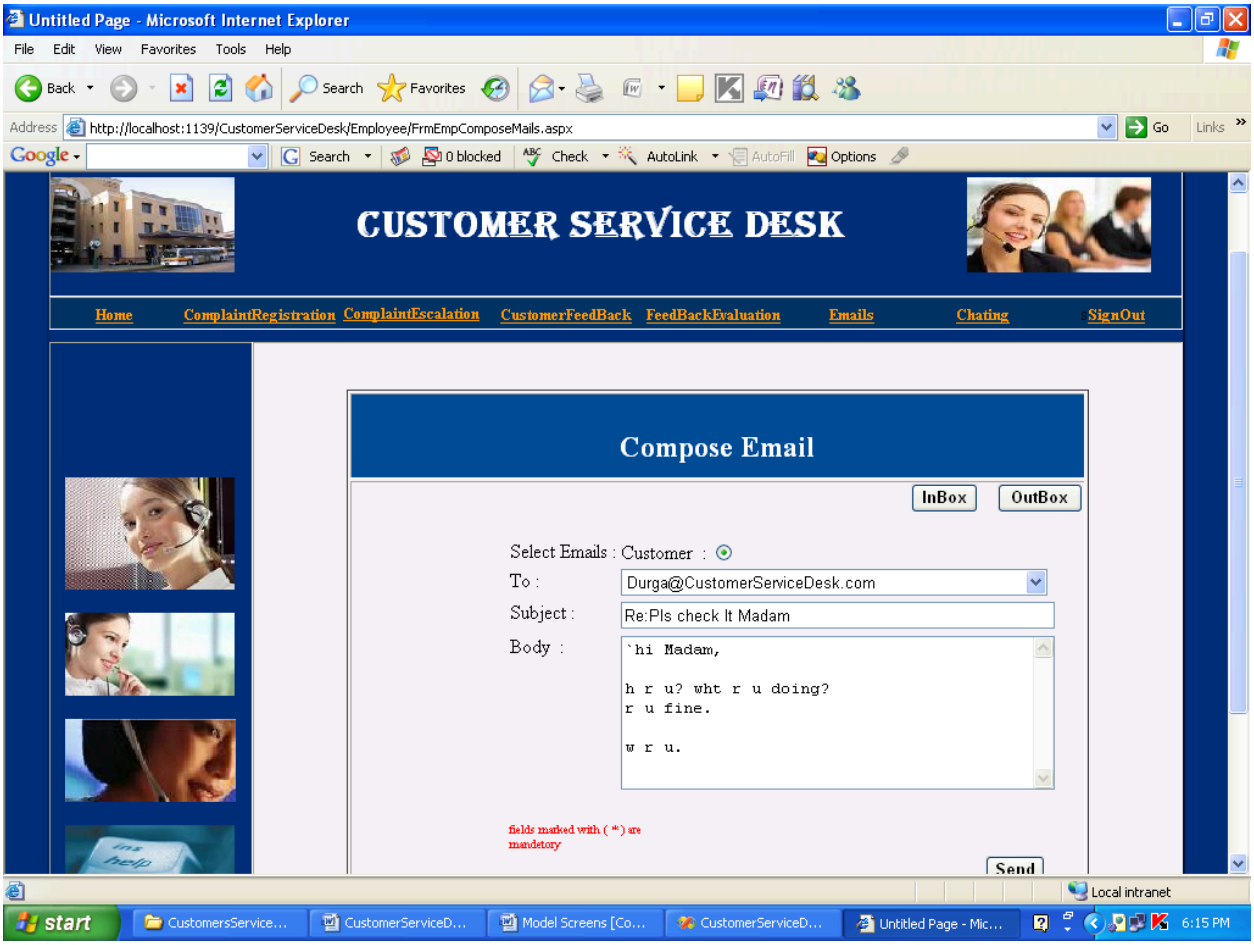
## Employee Complaint Regisrtation:



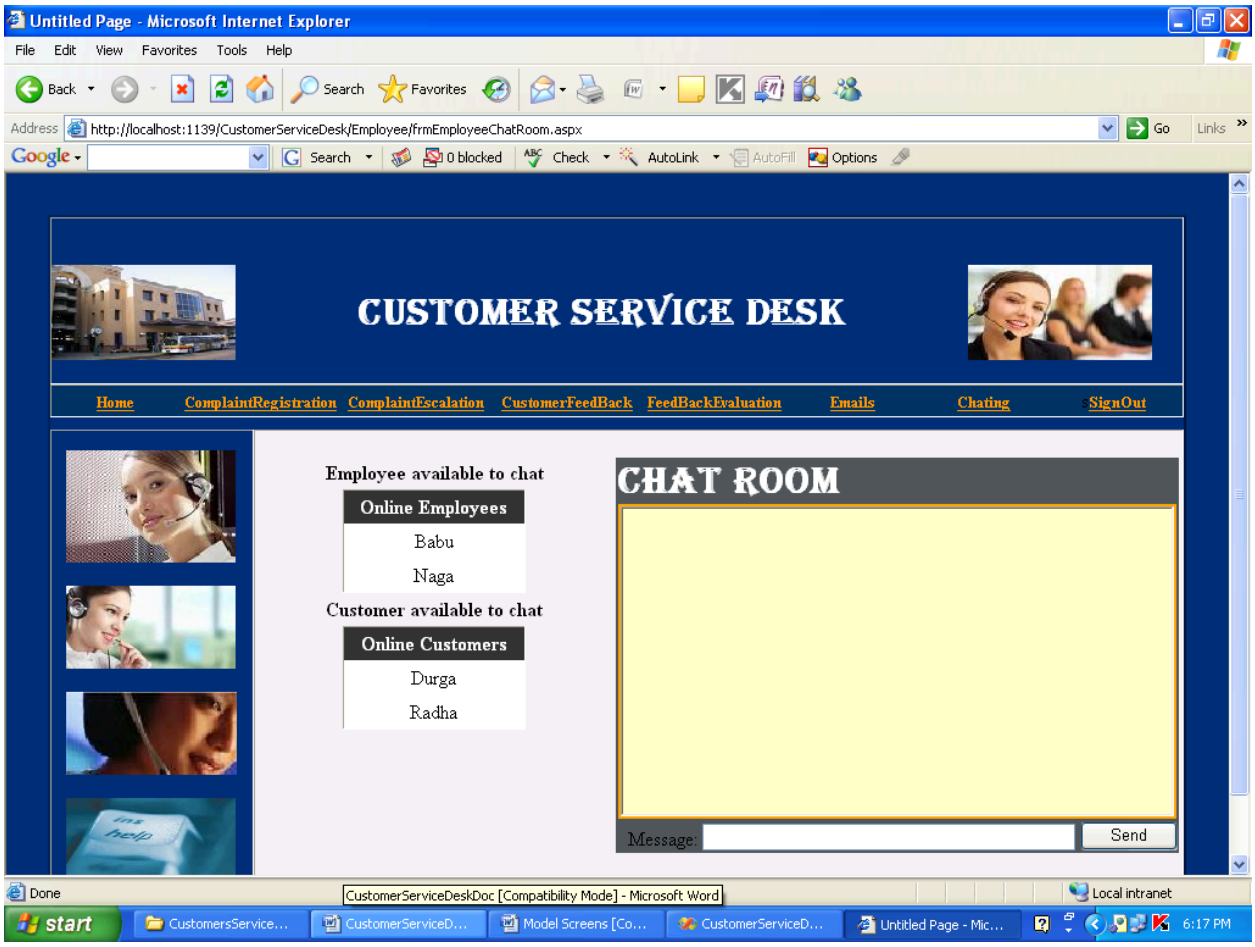
## Customer Feedback:



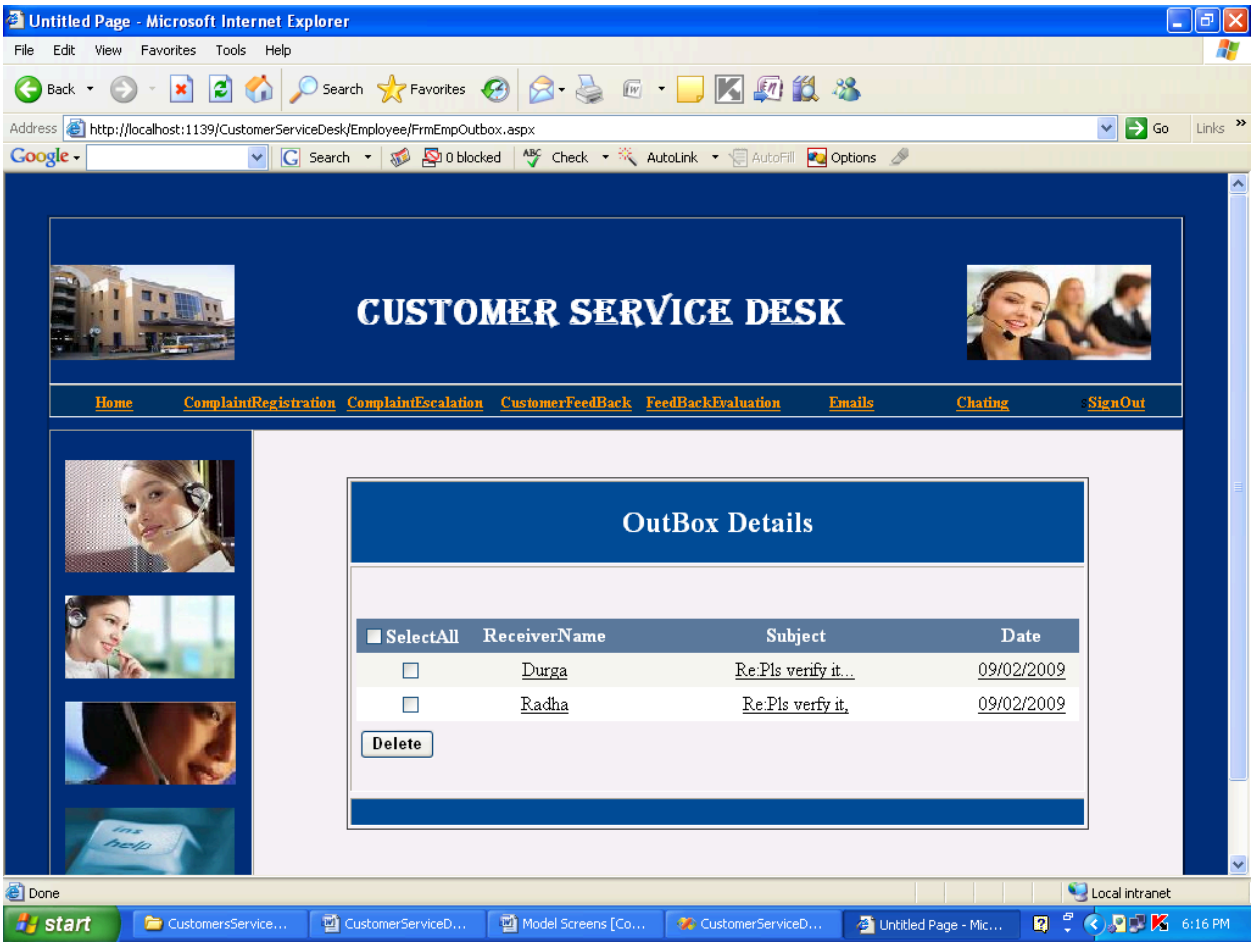
## Composing mail by Employee:



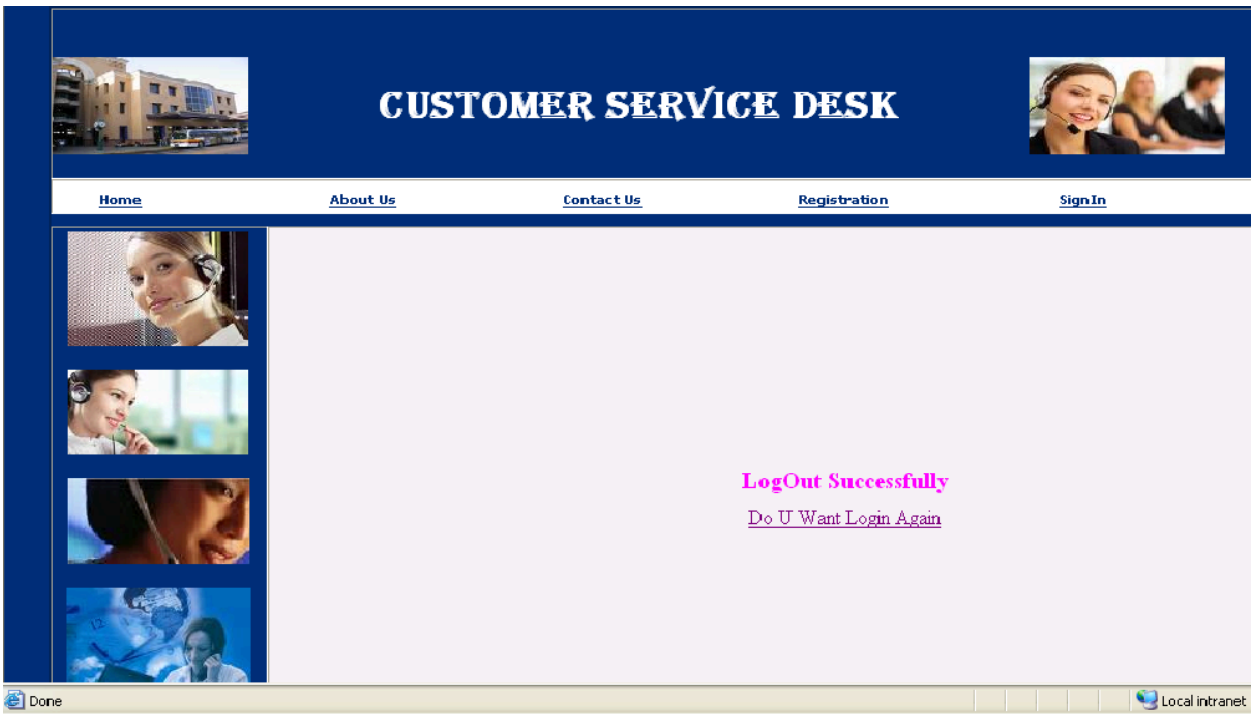
## Employee chat room:



## Viewing outbox mails:



## Employee Logout Page:



# CHAPTER 9

# System Security

## 9.1. Introduction

The protection of computer based resources that includes hardware, software, data, procedures and people against unauthorized use or natural

Disaster is known as System Security.

System Security can be divided into four related issues:

- Security
- Integrity
- Privacy
- Confidentiality

**SYSTEM SECURITY** refers to the technical innovations and procedures applied to the hardware and operation systems to protect against deliberate or accidental damage from a defined threat.

**DATA SECURITY** is the protection of data from loss, disclosure, modification and destruction.

**SYSTEM INTEGRITY** refers to the power functioning of hardware and programs, appropriate physical security and safety against external threats such as eavesdropping and wiretapping.

**PRIVACY** defines the rights of the user or organizations to determine what information they are willing to share with or accept from others and how the organization can be protected against unwelcome, unfair or excessive dissemination of information about it.

**CONFIDENTIALITY** is a special status given to sensitive information in a database to minimize the possible invasion of privacy. It is an attribute of information that characterizes its need for protection.

## 9.2. SECURITY IN SOFTWARE

System security refers to various validations on data in form of checks and controls to avoid the system from failing. It is always important to ensure that only valid data is entered and only valid operations are performed on the system. The system employees two types of checks and controls:

## CLIENT SIDE VALIDATION

Various client side validations are used to ensure on the client side that only valid data is entered. Client side validation saves server time and load to handle invalid data. Some checks imposed are:

- VBScript in used to ensure those required fields are filled with suitable data only. Maximum lengths of the fields of the forms are appropriately defined.
- Forms cannot be submitted without filling up the mandatory data so that manual mistakes of submitting empty fields that are mandatory can be sorted out at the client side to save the server time and load.
- Tab-indexes are set according to the need and taking into account the ease of user while working with the system.

## SERVER SIDE VALIDATION

Some checks cannot be applied at client side. Server side checks are necessary to save the system from failing and intimating the user that some invalid operation has been performed or the performed operation is restricted. Some of the server side checks imposed is:

- Server side constraint has been imposed to check for the validity of primary key and foreign key. A primary key value cannot be duplicated. Any attempt to duplicate the primary value results into a message intimating the user about those values through the forms using foreign key can be updated only of the existing foreign key values.
- User is intimating through appropriate messages about the successful operations or exceptions occurring at server side.
- Various Access Control Mechanisms have been built so that one user may not agitate upon another. Access permissions to various types of users are controlled according to the organizational structure. Only permitted users can log on to the system and can have access according to their category. User-name, passwords and permissions are controlled o the server side.
- Using server side validation, constraints on several restricted operations are imposed.

# Chapter 10

# CONCLUSION

# CONCLUSION

- **It is a web-enabled project.**
- **With this project the details about the product will be given to the customers in detail with in a short span of time.**
- **Queries regarding the product or the services will also be clarified.**
- **It provides more knowledge about the various technologies.**

# Chapter 11

# FUTURE IMPROVEMENT

# Chapter 12

# BIBLIOGRAPHY

- **FOR .NET INSTALLATION**

  www.support.mircosoft.com
- **FOR DEPLOYMENT AND PACKING ON SERVER**

  www.developer.com

  www.15seconds.com
- **FOR SQL**

  www.msdn.microsoft.com
- **FOR ASP.NET**

  www.msdn.microsoft.com/net/quickstart/aspplus/default.com

  www.asp.net

  www.fmexpense.com/quickstart/aspplus/default.com

  www.asptoday.com

  www.aspfree.com

  www.4guysfromrolla.com/index.aspx