INTRODUCTION

1.1. Project Overview

The Blood Donation Agent is to create an e-Information about the donor and organization that are related to donating the blood. Through this application any person who is interested in donating the blood can register himself in the same way if any organization wants to register itself with this site that can also register. Moreover if any general consumer wants to make request blood online he can also take the help of this site. Admin is the main authority who can do addition, deletion, and modification if required.

1.2. Project Description

This project is aimed to developing an online Blood Donation Information. The entire project has been developed keeping in view of the distributed client server computing technology, in mind.

The Blood Donation Agent is to create an e-Information about the donor and organization that are related to donating the blood. Through this application any person who is interested in donating the blood can register himself in the same way if any organization wants to register itself with this site that can also register. Moreover if any general consumer wants to make request blood online he can also take the help of this site.

Admin is the main authority who can do addition, deletion, and modification if required.

The project has been planned to be having the view of distributed architecture, with centralized storage of the database. The application for the storage of the data has been planned. Using the constructs of MS-SQL Server and all the user interfaces have been designed using the ASP.Net technologies.

The database connectivity is planned using the “SQL Connection” methodology. The standards of security and data protective mechanism have been given a big choice for proper usage.

The application takes care of different modules and their associated reports, which are produced as per the applicable strategies and standards that are put forwarded by the administrative staff.

The entire project has been developed keeping in view of the distributed client server computing technology, in mind. The specification has been normalized up to 3NF to eliminate all the anomalies that may arise due to the database transaction that are executed by the general users and the organizational administration. The user interfaces are browser specific to give distributed accessibility for the overall system. The internal database has been selected as MS-SQL server 2000.

The basic constructs of table spaces, clusters and indexes have been exploited to provide higher consistency and reliability for the data storage. The MS-SQL server 2000 was a choice as it provides the constructs of high-level reliability and security. The total front end was dominated using the ASP.Net technologies. At all proper levels high care was taken to check that the system manages the data consistency with proper business rules or validations.

The database connectivity was planned using the latest "SQL Connection" technology provided by Microsoft Corporation. The authentication and authorization was crosschecked at all the relevant stages. The user level accessibility has been restricted into two zones namely.

Problem Definition

2.1 Existing System

- Cannot Upload and Download the latest updates.
- No use of Web Services and Remoting.
- Risk of mismanagement and of data when the project is under development.
- Less Security.
- No proper coordination between different Applications and Users.
- Fewer Users – Friendly

Disadvantages

1. User friendliness is provided in the application with various controls.
2. The system makes the overall project management much easier and flexible.
3. Readily upload the latest updates, allows user to download the alerts by clicking the URL.
4. There is no risk of data mismanagement at any level while the project development is under process.
5. It provides high level of security with different level of authentication.

2.2. Proposed System

To debug the existing system, remove procedures those cause data redundancy, make navigational sequence proper. To provide information about audits on different level and also to reflect the current work status depending on organization/auditor or date. To build strong password mechanism.

Advantages:

- User friendliness I provided in the application with various controls.
- The system makes the overall project management much easier and flexible.
- Readily upload the latest updates ,allows user to download the alerts by clicking the url.
- It provides high level of security with different level of authentication.

Feasibility Study

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.

System Analysis

4.1. Software Requirement Specification (SRS)

The software, Site Explorer is designed for management of web sites from a remote location.

INTRODUCTION

Purpose: The main purpose for preparing this document is to give a general insight into the analysis and requirements of the existing system or situation and for determining the operating characteristics of the system.

Scope: This Document plays a vital role in the development life cycle (SDLC) and it describes the complete requirement of the system. It is meant for use by the developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

DEVELOPERS RESPONSIBILITIES OVERVIEW:

The developer is responsible for:

- Developing the system, which meets the SRS and solving all the requirements of the system?
- Demonstrating the system and installing the system at client's location after the acceptance testing is successful.
- Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.
- Conducting any user training that might be needed for using the system.
- Maintaining the system for a period of one year after installation.

The modules involved are:

1. **Administration:**

In this module the Administrator has the privileges to add all the Blood Groups, Blood Type, Organization, Type, Country, State, City, and Location. He can search all the info about the Organization, Donor.

User Account:

- AccountID
- Username
- Password
- UserDesc
- HintQuestion
- Answer
- RoleID
- Active

Functionality

- Association User Account with UserRole.
- Association User Account with Organisation.
- Association User Account with personal Details.
- Association User Account with Employee deatails.
- Association User Account with BloodDonation Details.

Alerts:

- All fields are mandatory
- Select user role
- Select role id
- Select role name

UserRole:

- RoleID
- RoleName
- RoleDesc
- Active

Functionality:

- Association user role with user Account

Alerts:

- Select Role Id
- Select role name

BDA State:

- StateID
- StateName
- StateCode
- StateDesc
- CountryID
- Active

Functionality:

- Association state with city
- Assocition state with Address

Alerts:

- Select State id
- Select state name

Country:

- CountryID
- CountryName
- CountryDesc

- CountryCode
- Active

Functionality:

- Association state with country
- Associtionstate with Address

Alerts:

- Select countryId
- Select countryname
- Select country code

BDA City:

- CityID
- CityName
- CityDesc
- CityCode
- StateID
- Active

Functionality:

- Association Location with city
- Assocition Address with city.

Alerts:

- Select cityId
- Select cityNane
- Select state code

BDALocation:

- LocationID
- LocationName
- LocationDesc
- LocationCode

- CityID
- Pin code
- Active

Functionality:

- Association Location with Address.

Alerts:

- Select LocationId
- Select Location Name
- Select Pincode.

BloodGroup:

- BloodGroupID
- BloodGroup
- Description
- Active

Functionality:

- Association Blood group with Personal details.

Alerts:

- Select BloodGroupID
- Select BloodGroupID

Blood Type:

- BloodTypeID
- TypeName
- TypeDesc
- Active

Functionality:

- Association Blood type with Personal details.

Alerts:

- Select BloodGroupID
- Select TypeName

Personal Details:

- UserAccountID
- FirstName
- MiddleName
- LastName
- Email
- DOB
- Weight
- Gender
- ImageURL
- BloodGroupID
- BloodType
- BloodType
- AddressID
- ContactNo_Office
- ContactNo_Residence
- MobileNo
- Active

Functionality:

- Association personal detaials with preferd location Day Time Details.

Alerts:

- Select user account id
- Select Email id
- Select date of birth

Call Center:

In this module all the employee who has been appointed by Admin will come. Admin will add all the information of employee and assign user name and password to them. By using that user

name and password they will enter to their login and can search for all the donor, and about all the blood request which have been made by either consumer, donor or any organization. Call center people will assign donor to related request.

Employee Detail:

- EmpId
- Name
- Address
- Phone
- Email
- Active

Functionality:

- Association Employee Details type with user Accounts.

Alerts:

- Select Emp Id
- Select email id

Donor:

Donor is that person who is interested in donating their blood so they can register themselves through this website. If any requirement comes then they will be contacted and they can donate their blood. Along with it they can search for the various organization locations wise and can also make request for blood if needed

Donation Frequencies:

- Frequency ID
- Frequency
- Description

Functionality:

- Association Donor Frequencies with Blood donation preferences.

Alerts:

- Select Frequency Id

Donor Preferred Organization:

- User Account D
- Organization ID
- Active

Functionality:

- Association Donor preferred organization with personal details.

Alerts:

- Select user account id
- Select organization id.

Organization:

In this module if any organization wants to register itself then it can do it. It can also search for donor location wise and if needed then it can also make request for blood

Organization:

- OrgID
- OrgName
- OrgType
- Email
- OrgAddrID
- OrgImageURL
- OrgDescription
- ContactNo
- MobileNo
- Active

- Comment

Functionality:

- Association organization type with Organization type.

Alerts:

- Select OrgId
- Select Email

Organization Type:

- TypeID
- TypeName
- Type Description

Functionality:

- Association organization type with Organization.

Alerts:

- Select Type Id
- Select Type Name

HARDWARE REQUIREMENTS:

- PIV 2.8 GHz Processor and Above

- RAM 512MB and Above
- HDD 20 GB Hard Disk Space and Above

SOFTWARE REQUIREMENTS:

- WINDOWS OS (XP / 2000 / 200 Server / 2003 Server)
- Visual Studio .Net 2005 Enterprise Edition
- Internet Information Server 5.0 (IIS)
- Visual Studio .Net Framework (Minimal for Deployment)
- SQL Server 2000 Enterprise Edition

System Design

5.1. Data Flow Diagrams (DFD)

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.

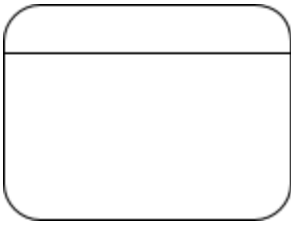
A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows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

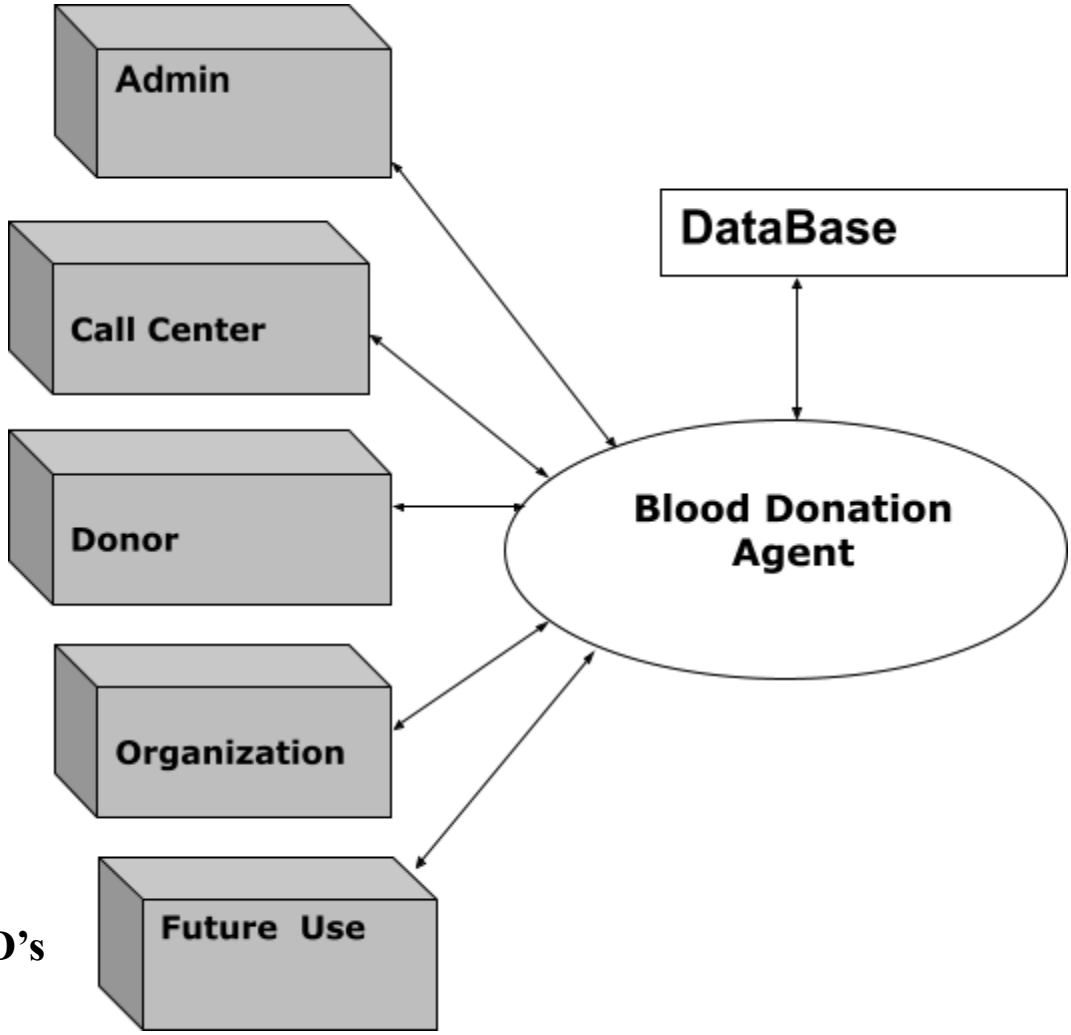
The origin and /or destination of data.

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

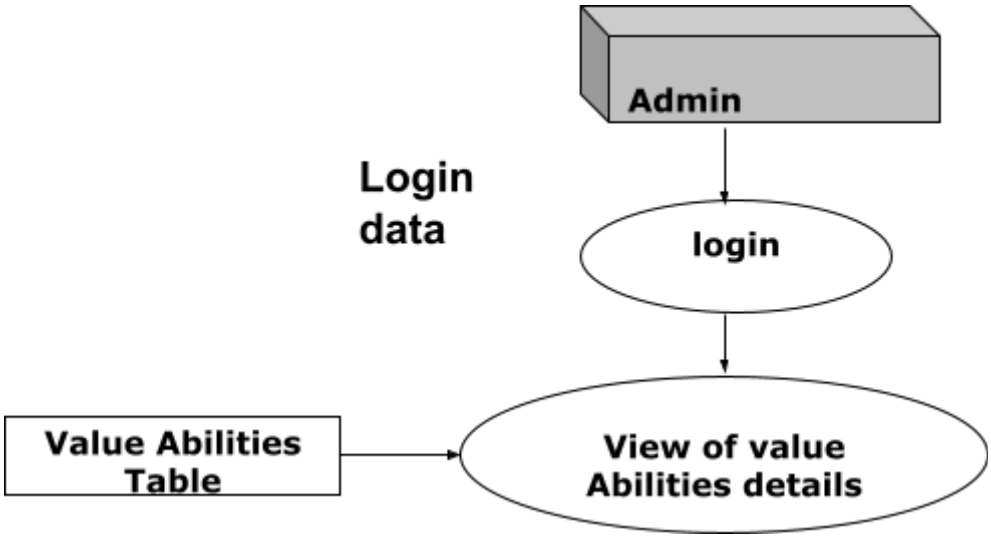
1) A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The later is usually indicated however by two separate arrows since these happen at different type.
2) A join in DFD means that exactly the 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.

Context Diagram

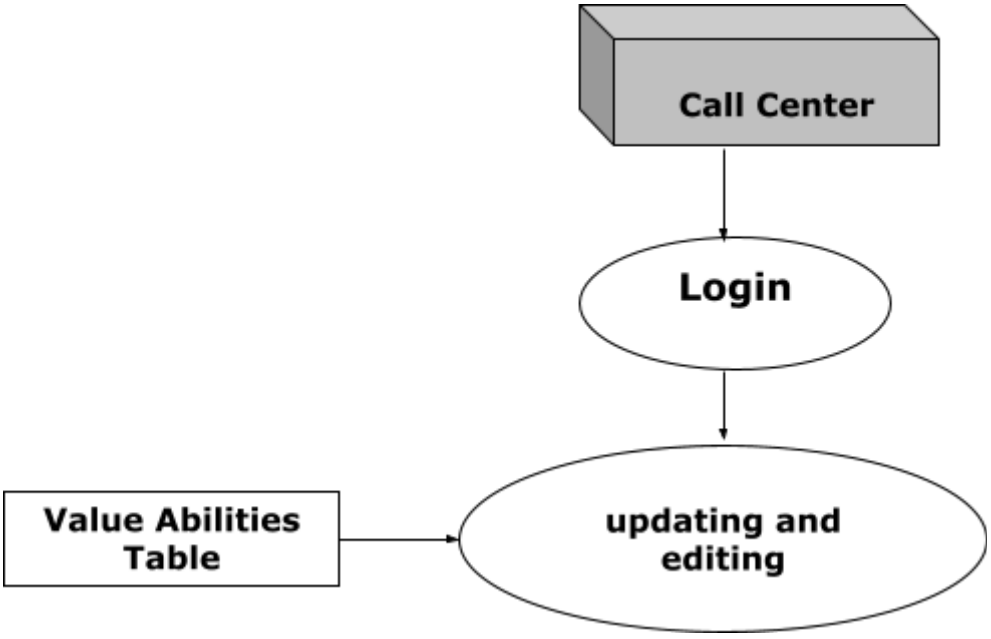


1st Level DFD's

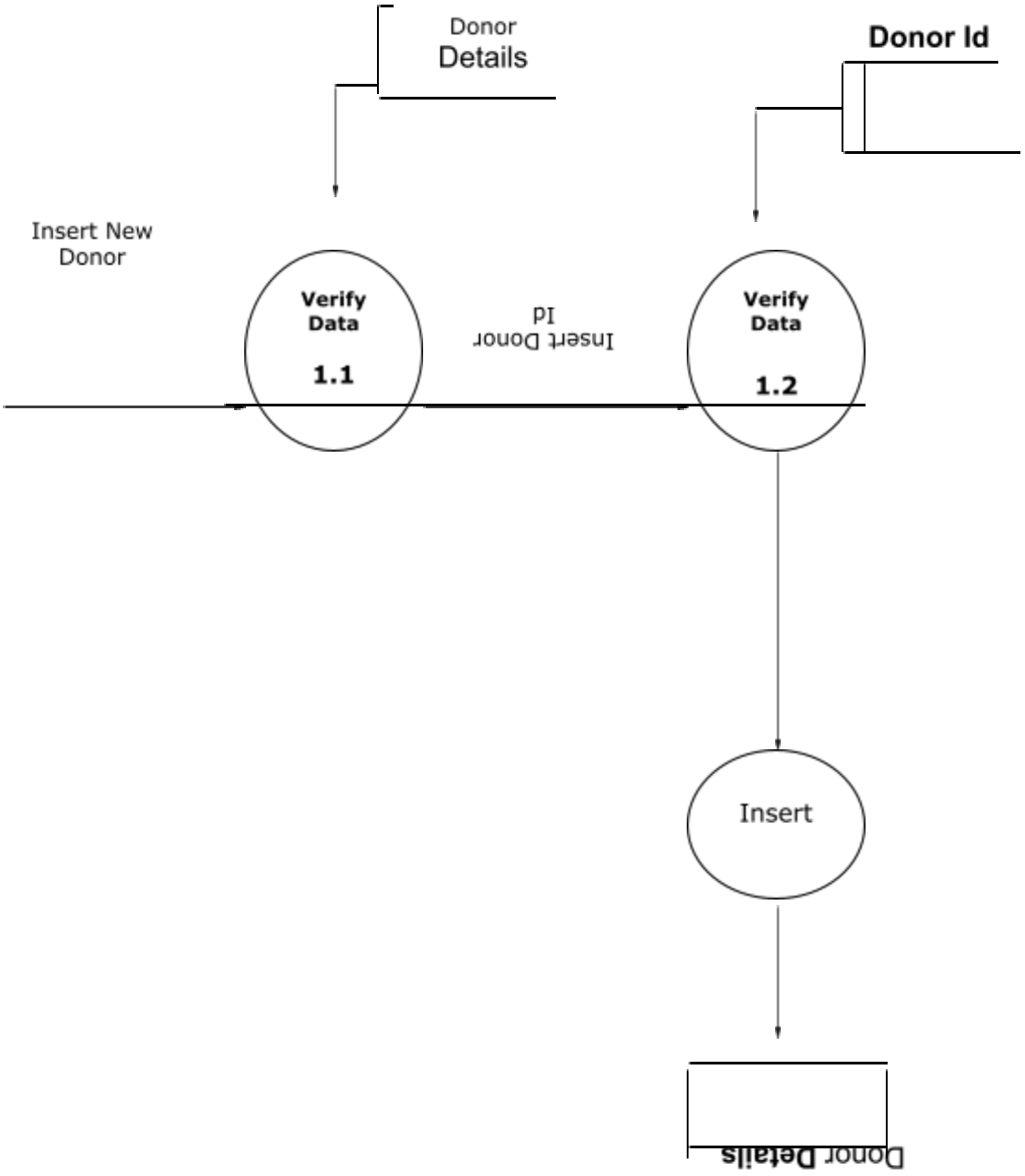
Level 1 DFD: For Admin Module



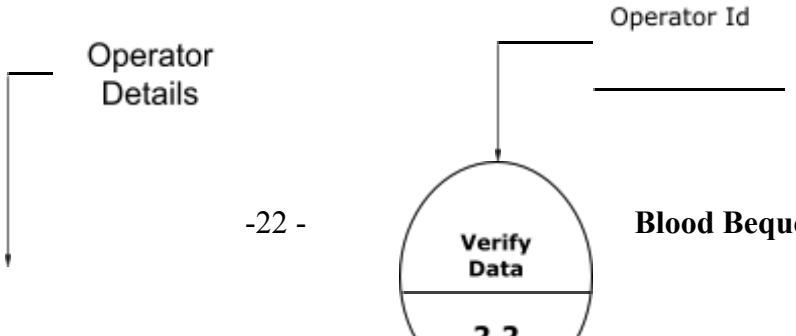
Level 1 DFD: For Users Module

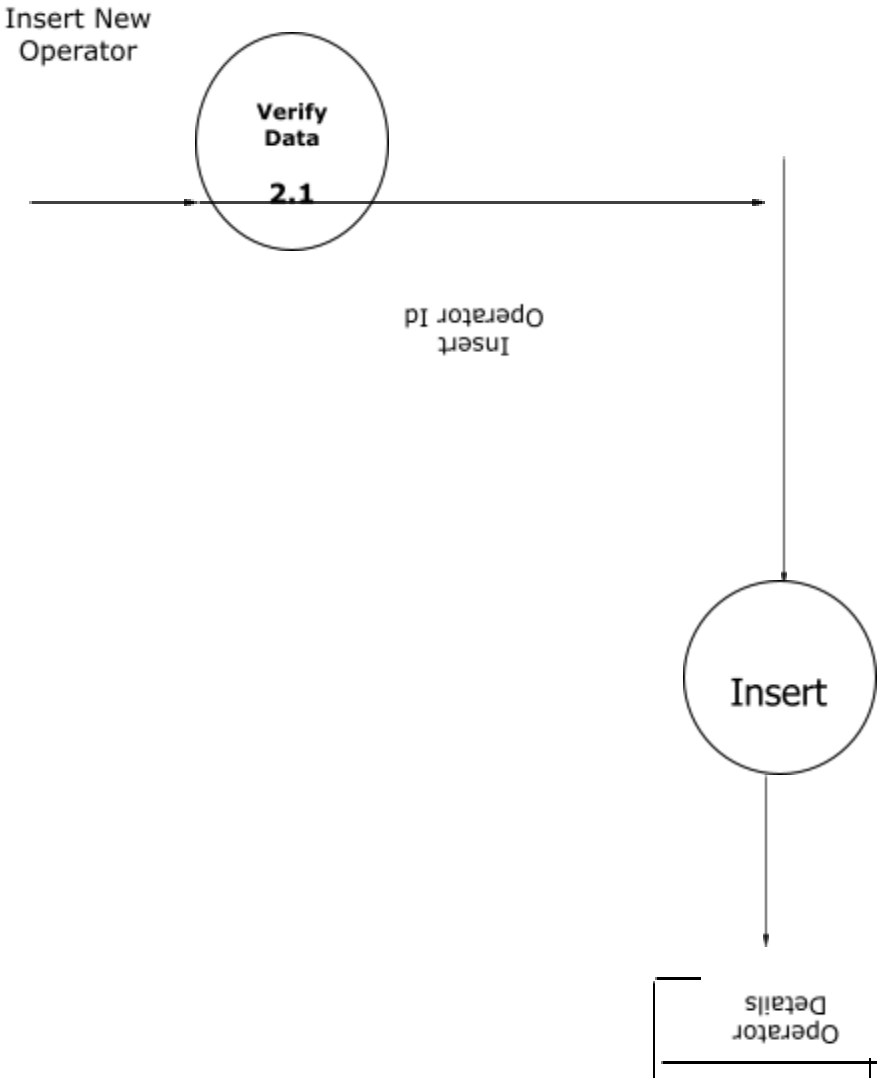


DFD For Donor Creation

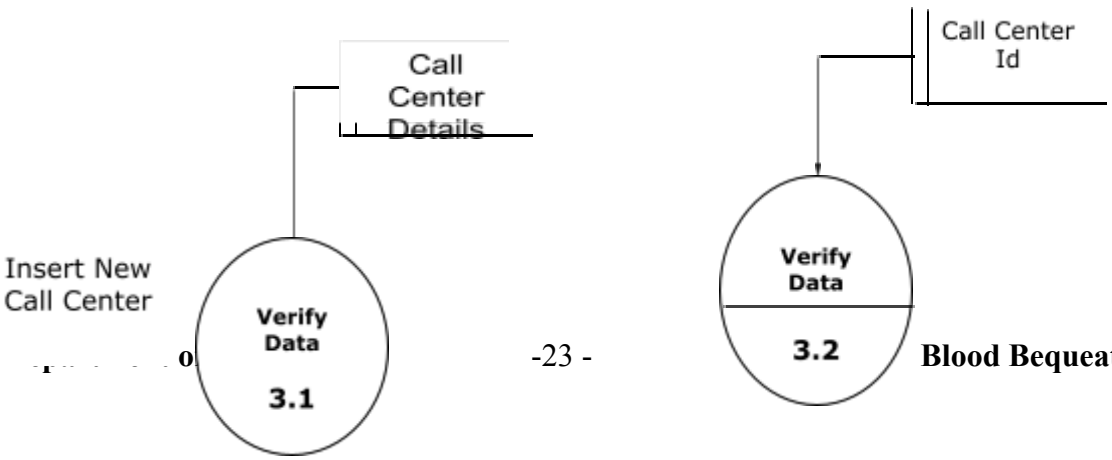


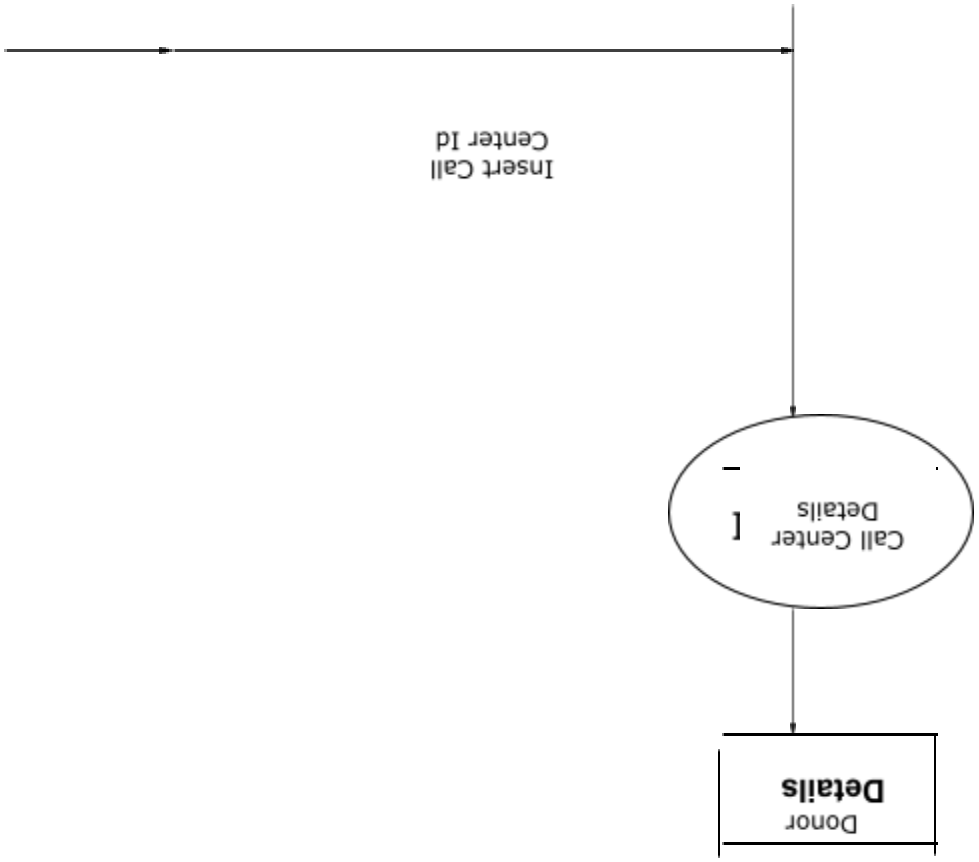
DFD For Operator Creation



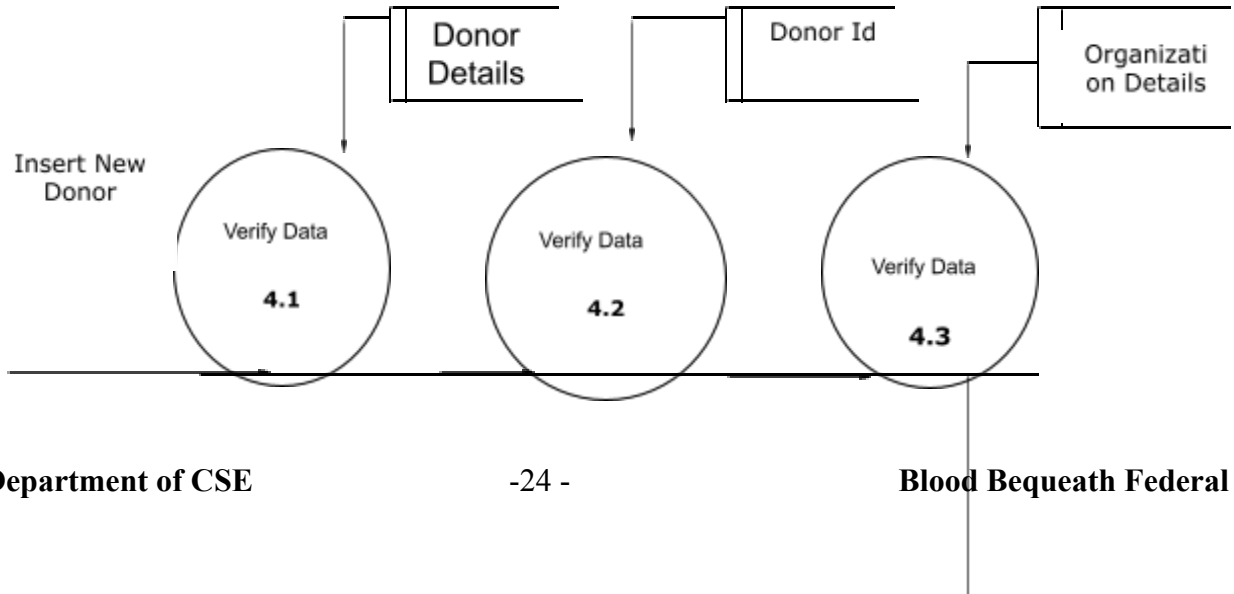


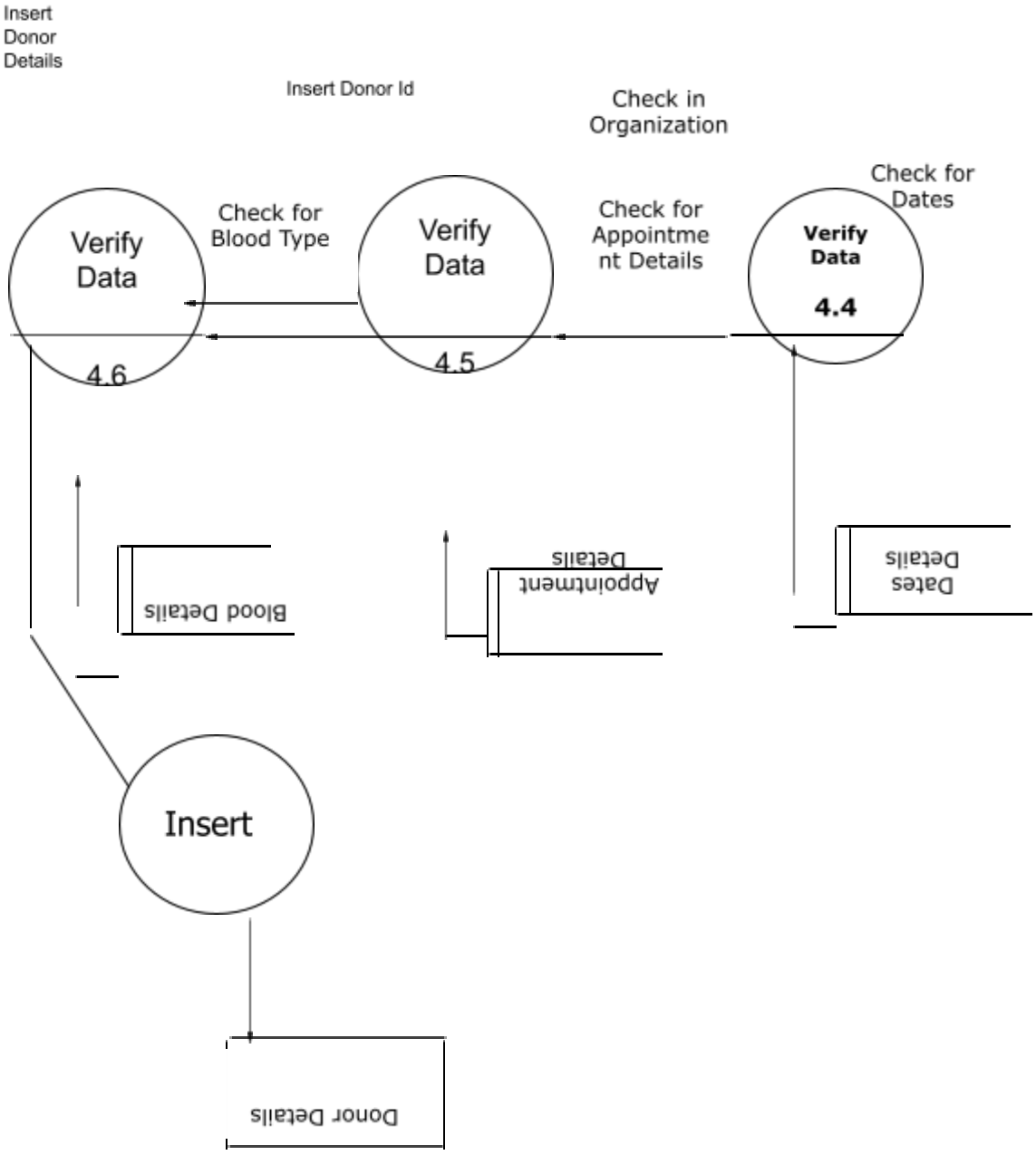
DFD For Call Center Creation





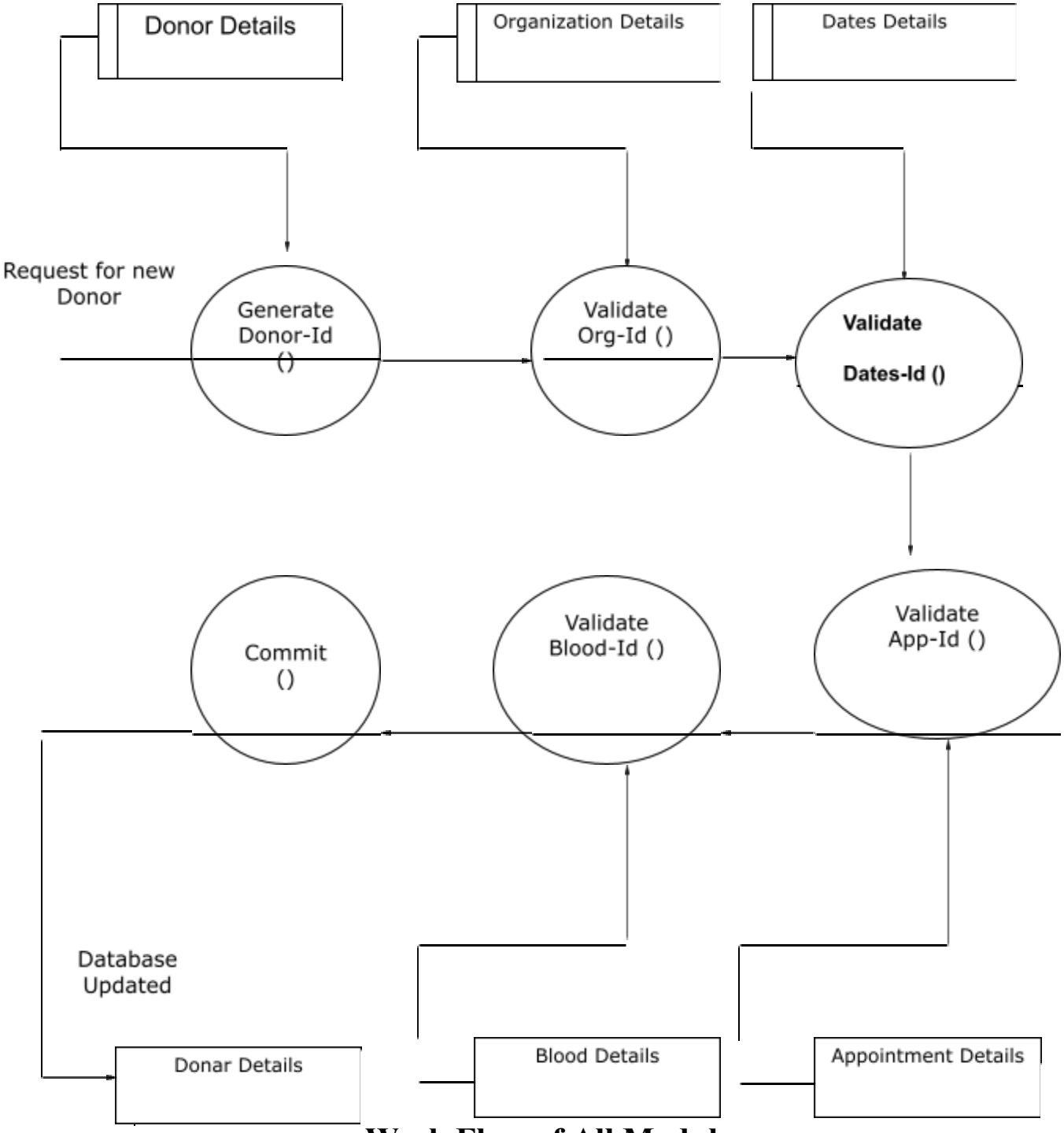
DFD for Donor Creation



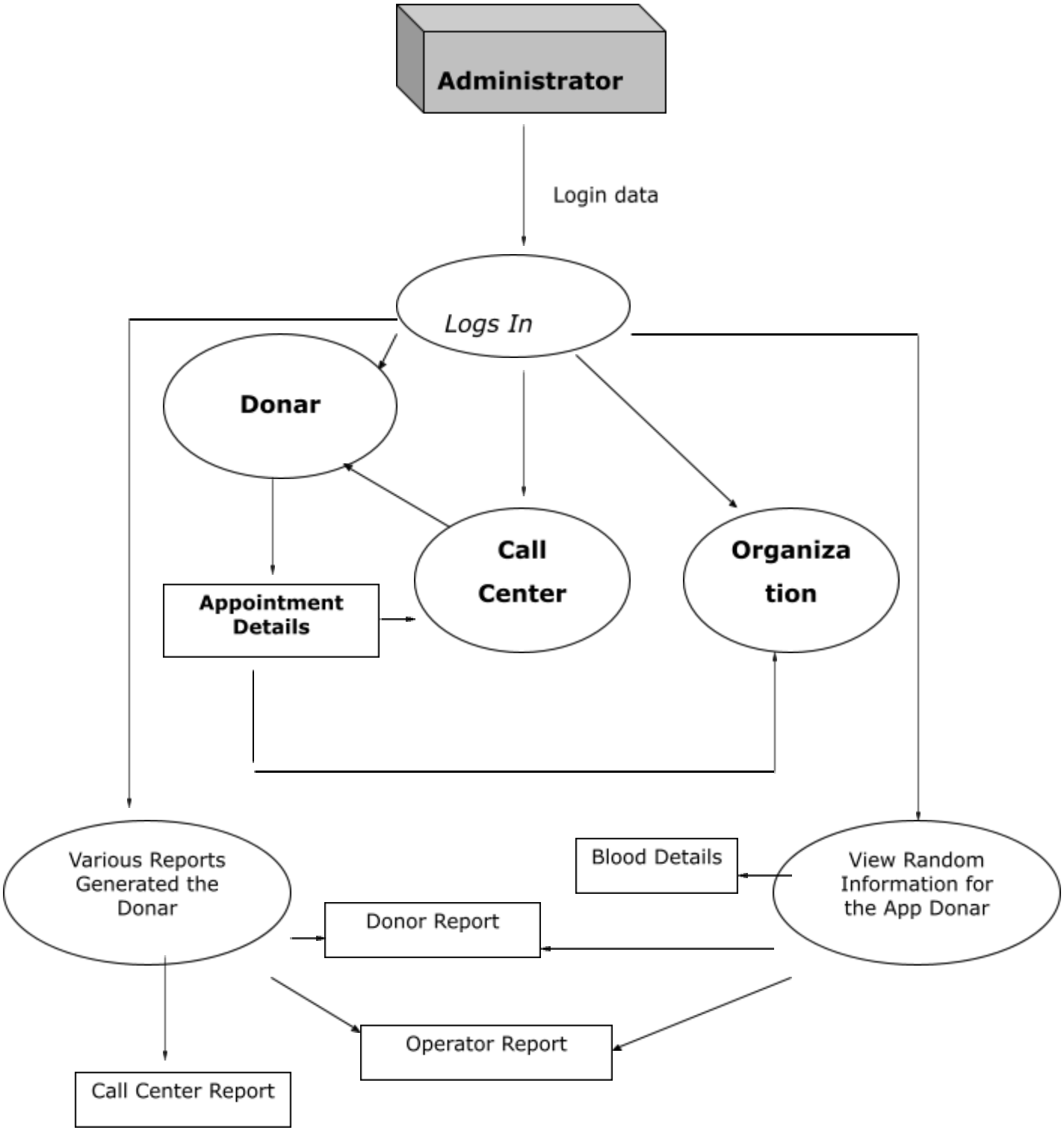

Insert
Donor
Details
Insert Donor Id
Check in
Organization
Check for
Dates
Verify
Data
Check for
Blood Type
Verify
Data
Check for
Appointme
nt Details
Verify
Data
4.4
4.6
4.5
Blood Details
Appointment
Details
Dates
Details
Insert
Donor Details

3rd Level DFD'S

DFD For Donor Creation



Work Flow of All Modules



5.2. UML Diagrams

Use Case Diagram:

- The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.
- A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.
- User Model View
    i. This view represents the system from the users perspective.
    ii. The analysis representation describes a usage scenario from the end-users perspective.

Structural model view

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

Behavioral model view

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

Implementation Model View

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

Environmental Model View

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

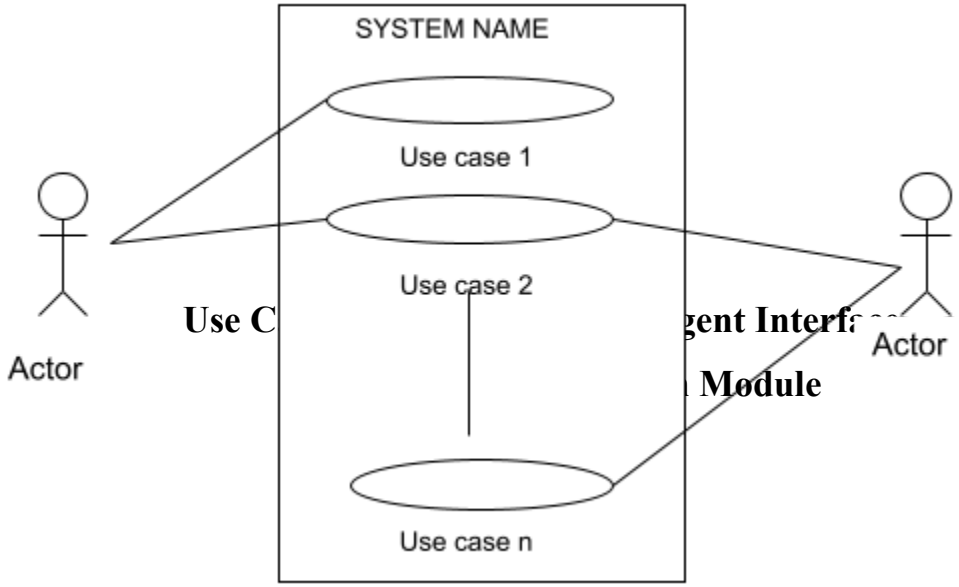
UML is specifically constructed through two different domains they are

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

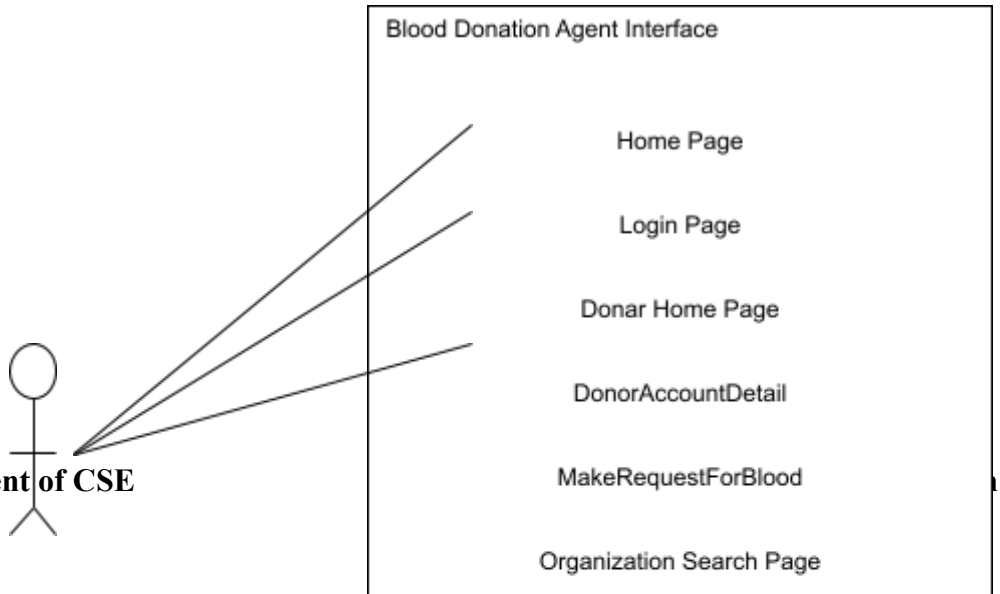
Use case Diagrams represent the functionality of the system from a user's point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

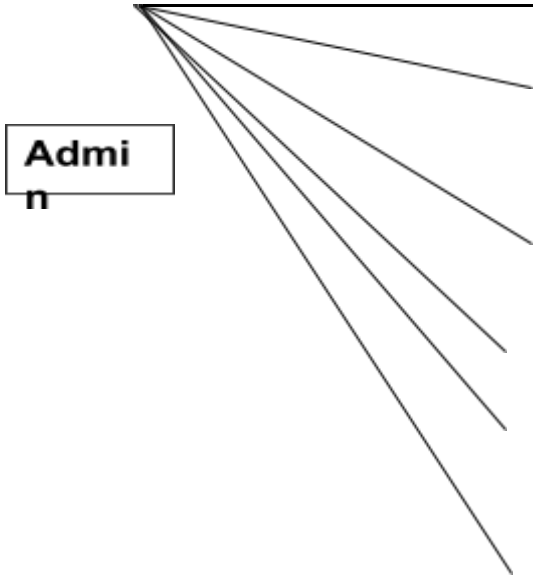
Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

Use case Model

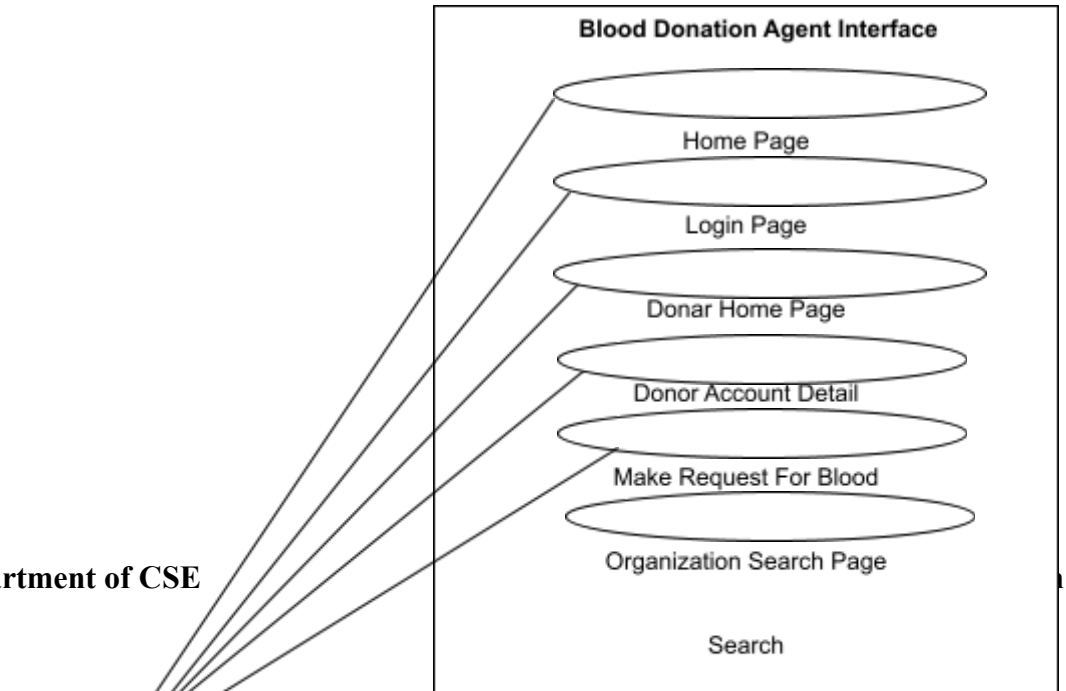


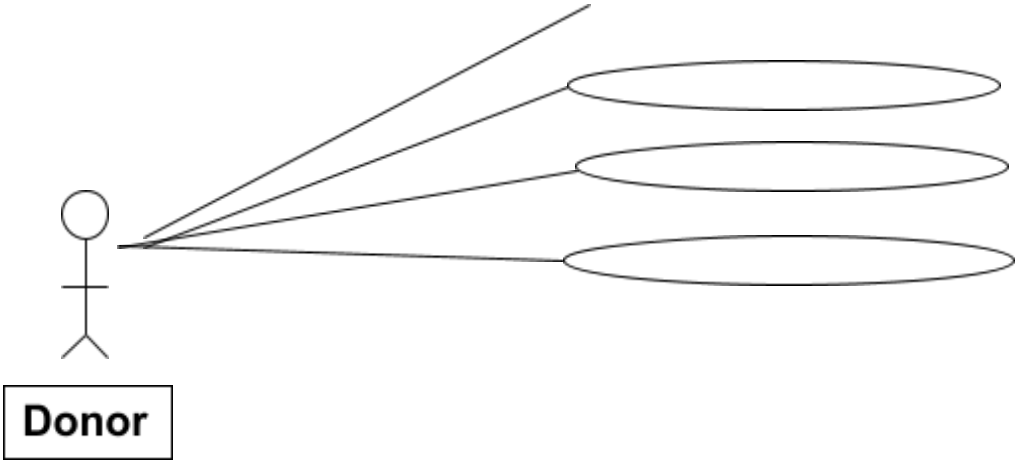
Use C **gent Interfa** **Module**



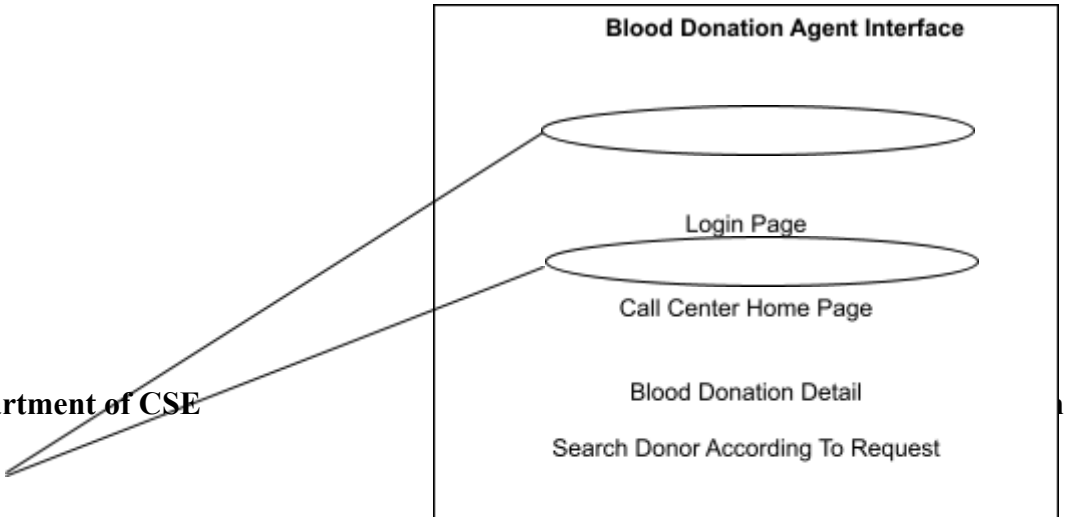


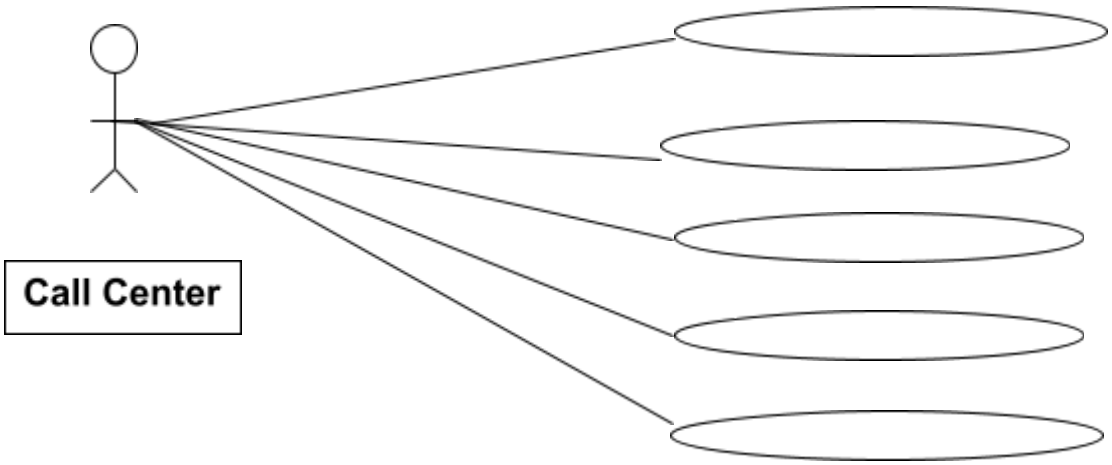
Use case For Donor Module



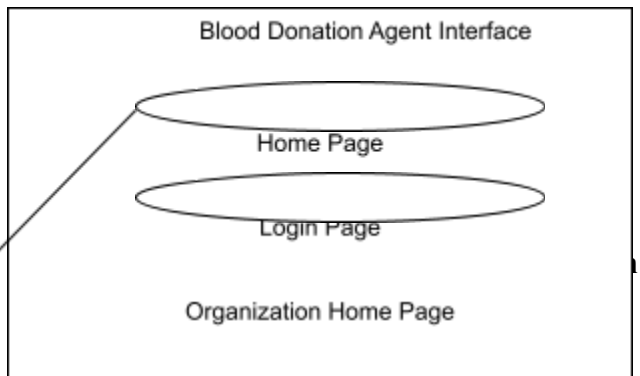


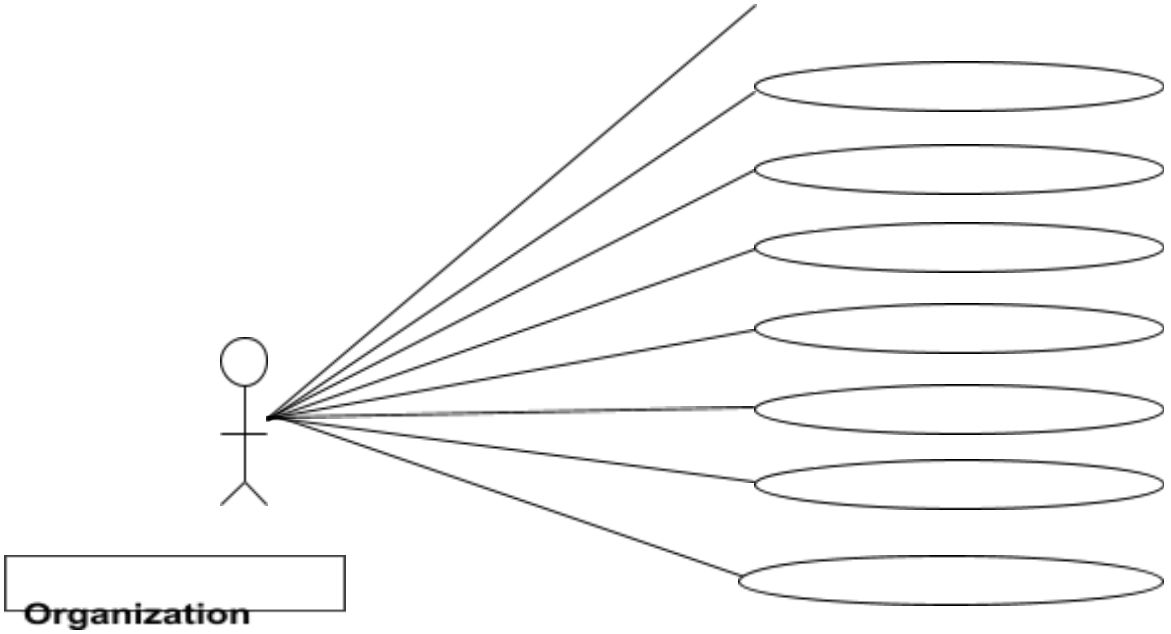
Use case For Call Center Module





Use case For Organization Module





Use case name	Login
Participating actors	Admin, Donor, Call Center, Organization
Flow of events	The Actor will give the user name and password to the system. The system will verify the authentication.
Entry Condition	The actor will enter the system by using username and password
Exit condition	If un authenticated should be exited
Quality Requirements	Password must satisfy the complexity requirements.

Use case name	Admin Registration
Participating actors	Admin
Flow of events	The Admin will submit all the details and place in the application.
Entry Condition	Must satisfy all the norms given by the Blood Donation Agent interface site.
Exit condition	Successful or Un successful completion of creation of account.
Quality Requirements	All fields are mandatory.

Use case name	Donor Registration
Participating actors	Donor
Flow of events	The Donor must enter all his personal details.
Entry Condition	View Home page
Exit condition	Registered Donor should be successfully logged out. Error Message should be displayed on Un successful creation.
Quality Requirements	Best Error Handling techniques. Check on Mandatory fields.

Use case name	Call Center Registration
Participating actors	Call Center
Flow of events	The Call Center must enter all his personal details.
Entry Condition	View Home page

Exit condition	Registered Call Center should be successfully logged out. Error Message should be displayed on Un successful creation.
Quality Requirements	Best Error Handling techniques. Check on Mandatory fields.

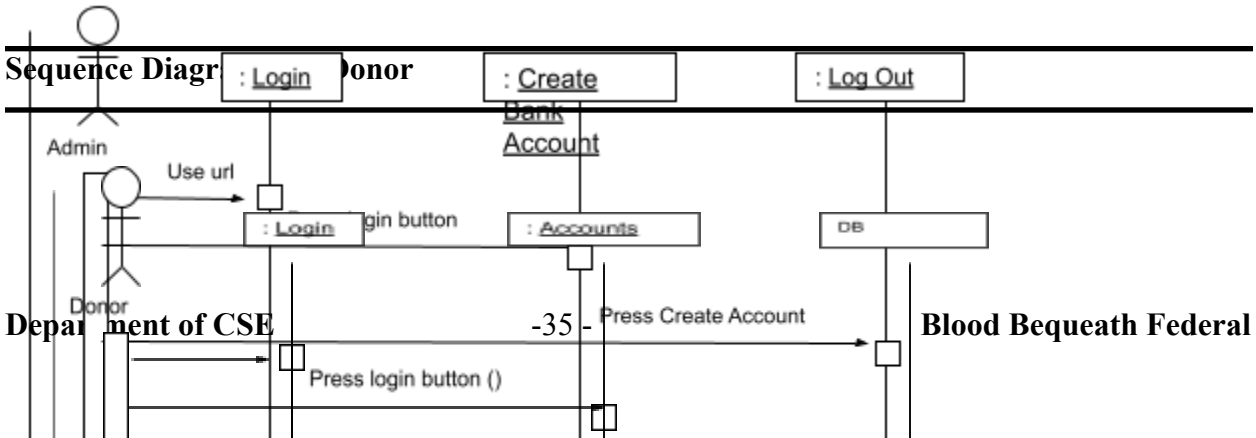
Use case name	Organization Registration
Participating actors	Organization
Flow of events	The Organization must enter all his personal details.
Entry Condition	View Home page
Exit condition	Registered Organization should be successfully logged out. Error Message should be displayed on Un successful creation.
Quality Requirements	Best Error Handling techniques. Check on Mandatory fields.

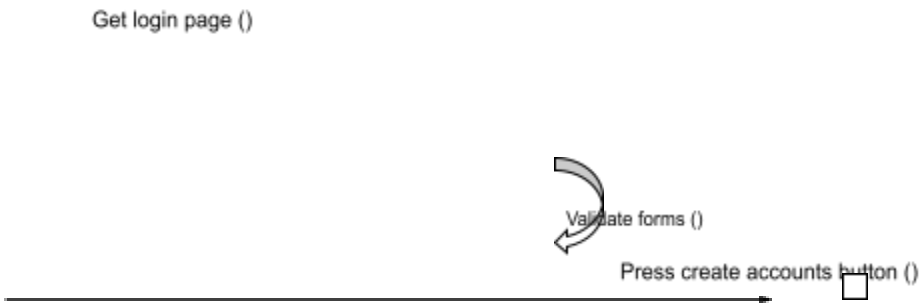
SEQUENCE DIAGRAMS

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically.

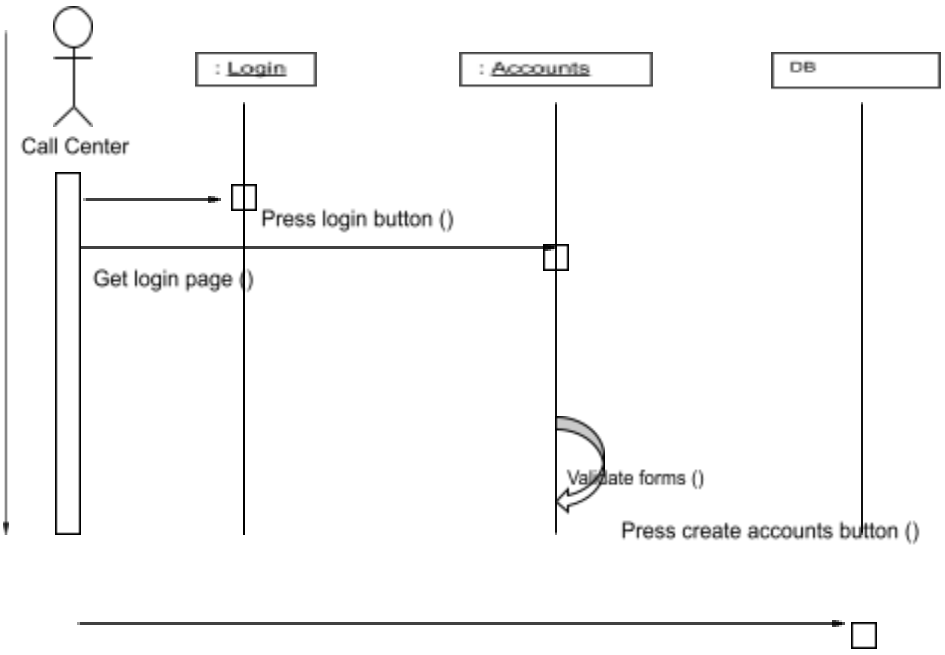
Sequence Diagram for Admin

Sequence Diagr onor

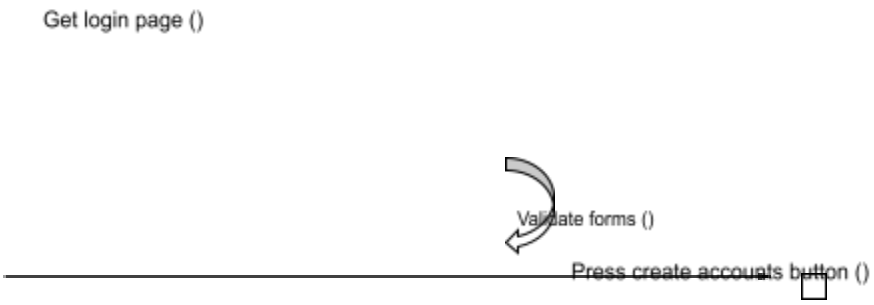




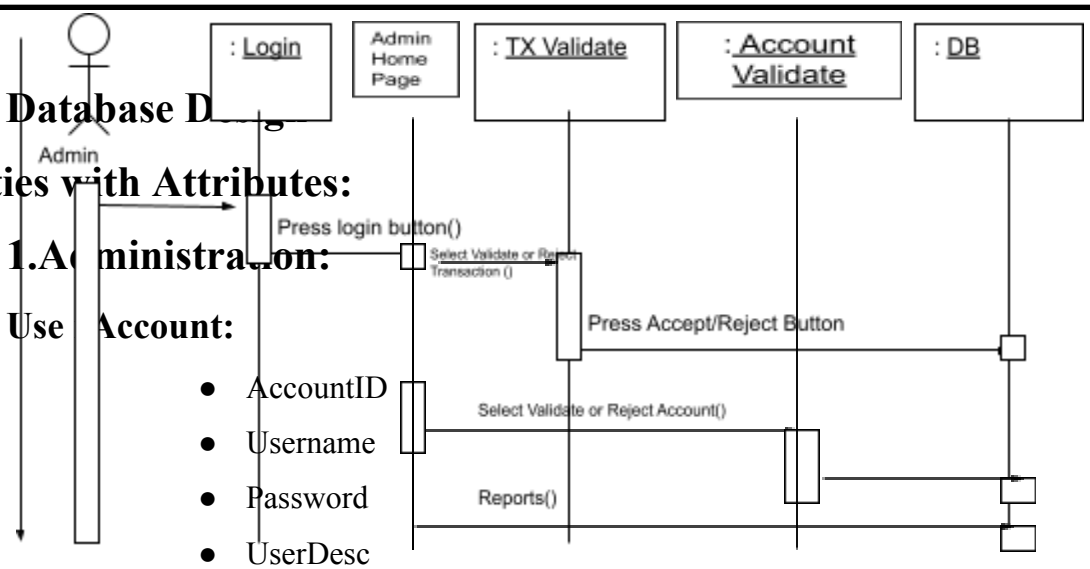
Sequence Diagram for callcenter



Sequence Diagram for organization



Sequence Diagram 5



5.4. Database D...

Entities with Attributes:

1.Administration:

User Account:

- AccountID
- Username
- Password
- UserDesc
- HintQuestion
- Answer
- RoleID
- Active

UserRole:

- RoleID
- RoleName
- RoleDesc
- Active

BDAState:

- StateID
- StateName
- StateCode
- StateDesc
- CountryID
- Active

Country:

- CountryID
- CountryName
- CountryDesc
- CountryCode

BDACity

- CityID
- CityName
- CityDesc
- CityCode
- StateID
- Active

:

BDALocation:

- LocationID
- LocationName

- LocationDesc
- LocationCode
- CityID
- Pin code
- Active

BloodGroup:

- BloodGroupID
- BloodGroup
- Description
- Active

BloodType:

- BloodTypeID
- TypeName
- TypeDesc
- Active

PersonalDetails:

- UserAccountID
- FirstName
- MiddleName
- LastName
- Email
- DOB
- Weight
- Gender
- ImageURL
- BloodGroupID
- BloodType
- BloodType
- AddressID
- ContactNo_Office

- ContactNo_Residence
- MobileNo
- Active

EmployeeDetail:

- EmpId
- Name
- Address
- Phone
- Email
- Active

DonationFrequencies:

- FrequencyID
- Frequency
- Description
- Active

Donor Preferred Organization:

- UserAcountID
- OrganisationID
- Active

Organisation:

- OrgID
- OrgName
- OrgType

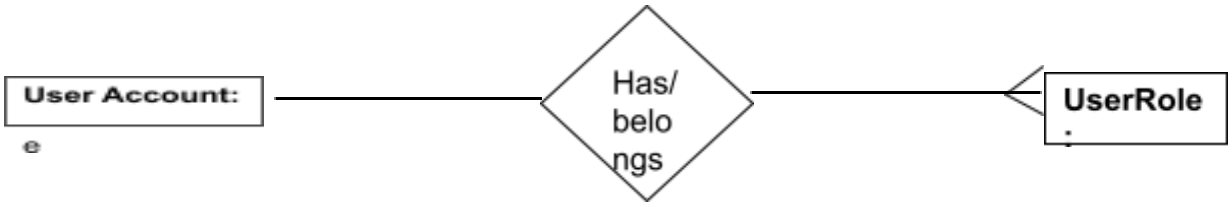
- Email
- OrgAddrID
- OrgImageURL
- OrgDescription
- ContactNo
- MobileNo
- Active
- Comment

OrganisationType:

- TypeID
- TypeName
- TypeDescription
- OrgImage
- Active

ER Diagrams

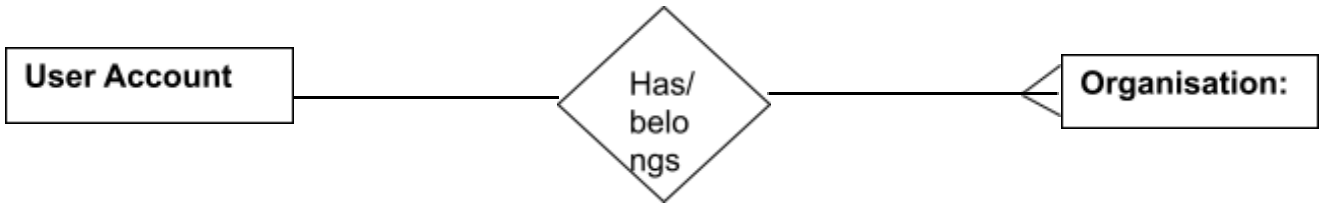
ER diagram for User Account and Users



UserRole:
RoleID
RoleName
RoleDesc
Active
UserAccount:
AccountID
Username
Password
UserDesc
HintQuestion
Answer

RoleID

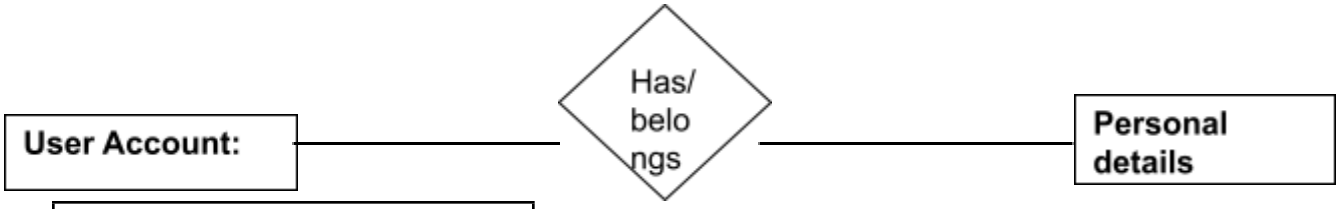
ER diagram for User Account and Organisation



UserAccount:
AccountID
Username
Password

UserDesc
Hint Question
Answer
RoleID
Organisation:
OrgID
OrgName
OrgType
Email
OrgAddrID
OrgImageURL
OrgDescription
ContactNo
MobileNo
Active
Comment

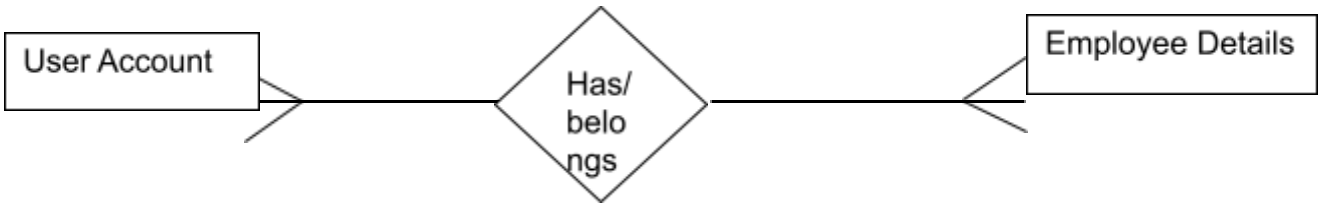
ER diagram for Users Account and Personal Details



Personal Details:
First Name
Email
DOB
Weight
Gender
ImageURL
BloodGroupID
BloodType
BloodType
AddressID

ContactNo_Office
ContactNo_Residence
MobileNo
Active
MiddleName
UserAccount:
AccountID
Username
Password
UserDesc
HintQuestion
Answer
RoleID

ER diagram for User Account and Employee Details



Employee Details:
Emp Id
Name
Address
Phone
Email
Active
User Account:

AccountID
Username
Password
User Desc
Hint Question
Answer
Role ID

ER diagram for User Role and Blood User account

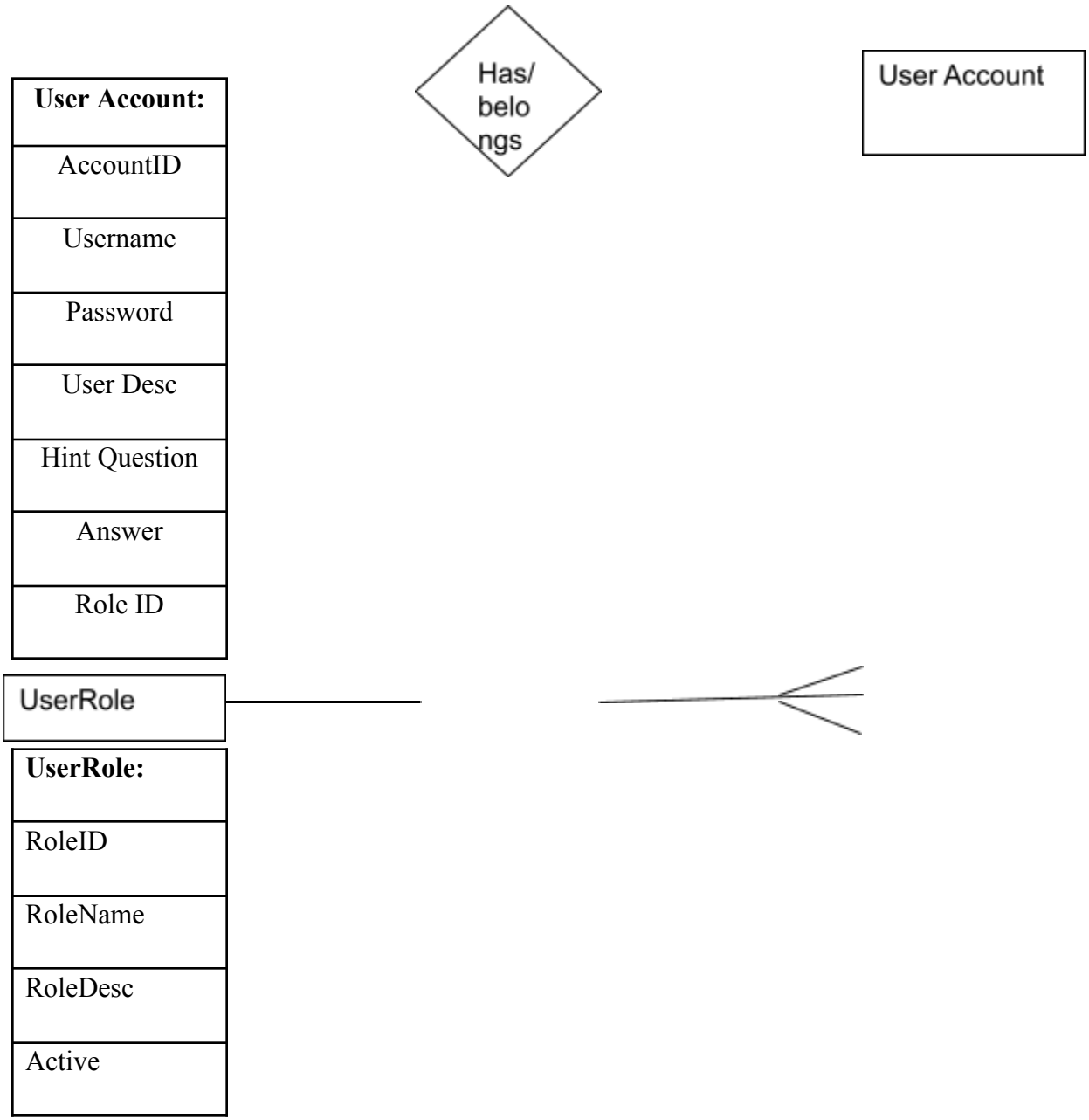
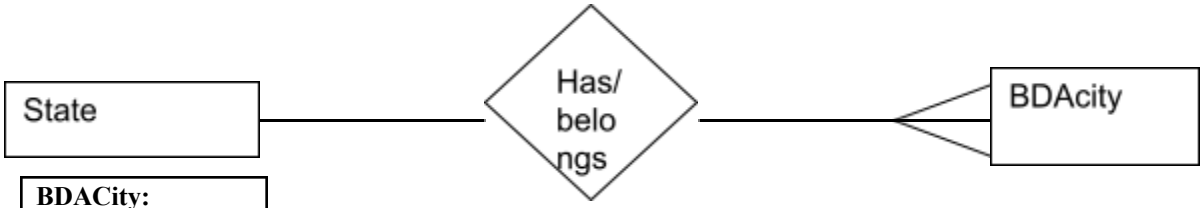

Has/
belo
ngs
User Account
User Account:
AccountID
Username
Password
User Desc
Hint Question
Answer
Role ID
UserRole
UserRole:
RoleID
RoleName
RoleDesc
Active

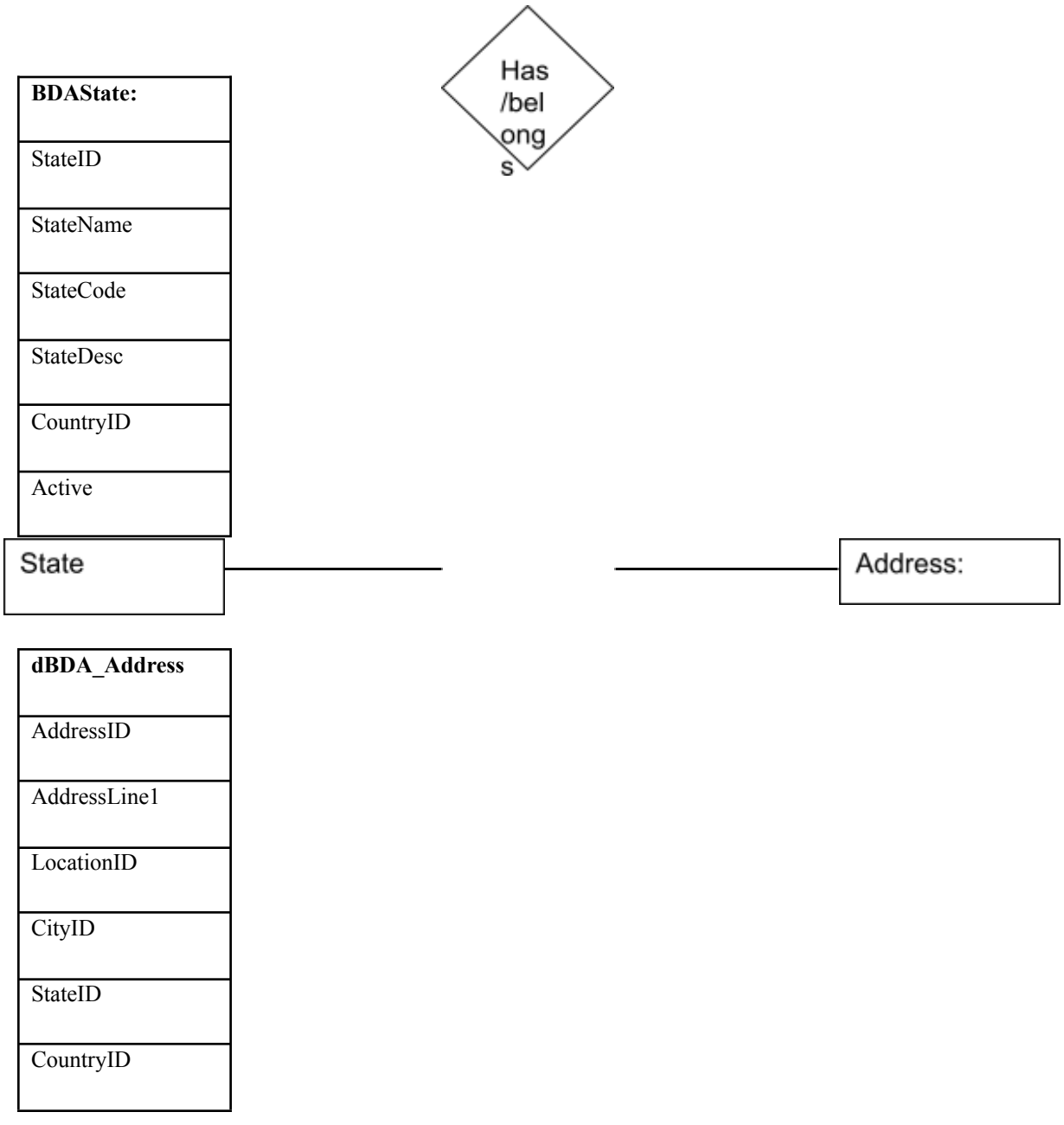
diagram for State and BDA City



BDACity:
CityID
CityName
CityDesc
CityCode
StateID
Active

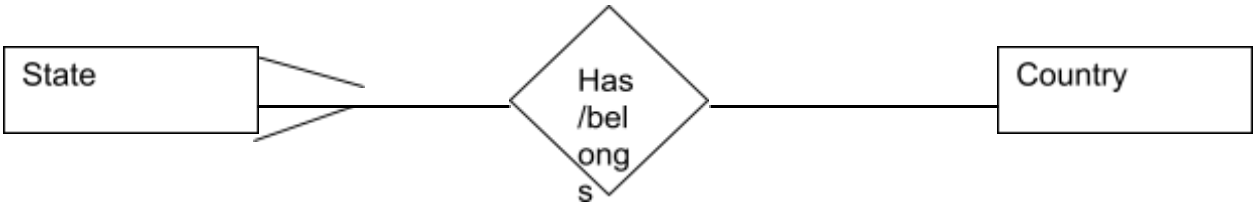
BDAState:
StateID
StateName
StateCode
StateDesc
CountryID
Active

ER diagram for State and Address



CountryID
Zipcode
Active

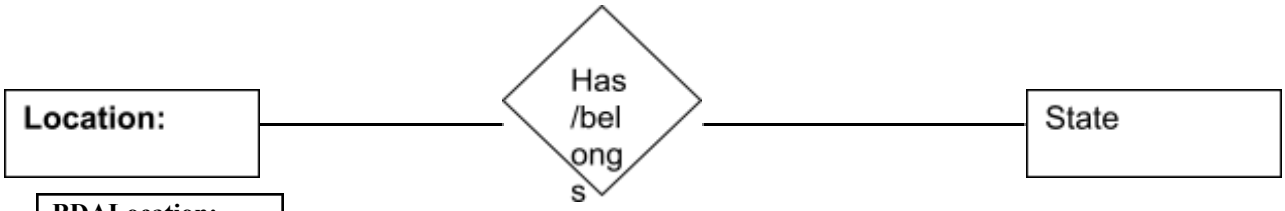
ER diagram for State and Country:



BDAState:
StateID
StateName
StateCode
StateDesc
CountryID
Active

Country:
CountryID
CountryName
CountryDesc
CountryCode
Active

ER diagram for Location and City

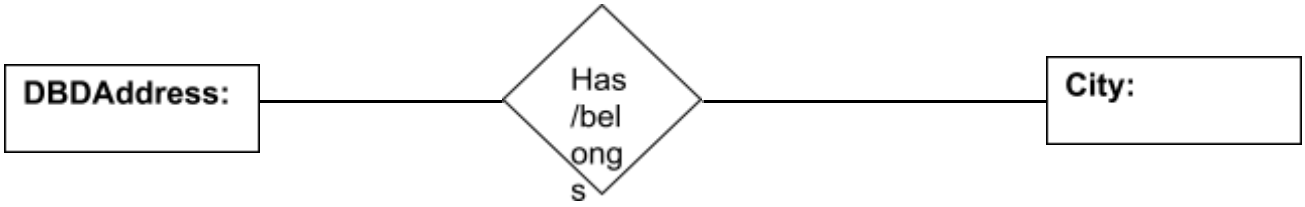


BDALocation:
LocationID
LocationName
LocationDesc
LocationCode
CityID
Pin code

Active

BDAState:
StateID
StateName
StateCode
StateDesc
CountryID
Active

ER diagram for Location and City



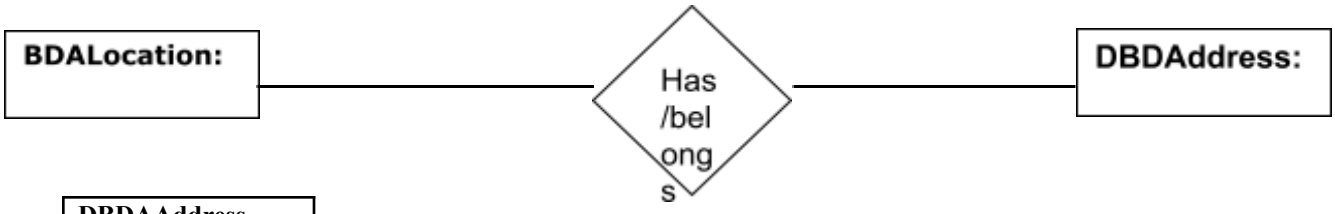
BDACity:
CityID
CityName
CityDesc
CityCode
StateID
Active

dBDA_Address
AddressID
AddressLine1
LocationID

Blood Bequeath Federal

CityID
StateID
CountryID
CountryID
Zipcode
Active

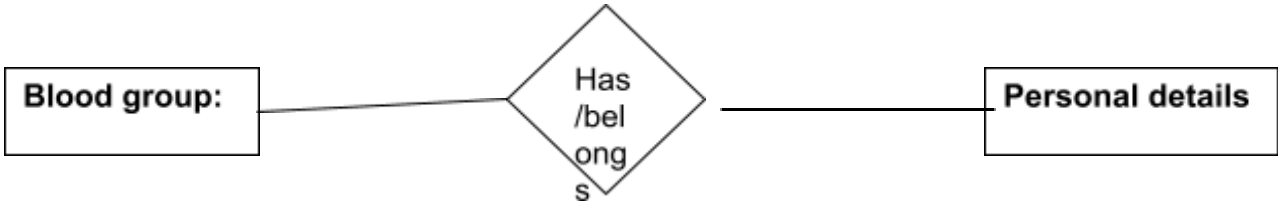
ER diagram for Location and Address



DBDAAddress
AddressID
AddressLine1
LocationID
CityID
StateID
CountryID
CountryID
Zipcode
Active
BDALocation:
LocationID
LocationName
LocationDesc
LocationCode
CityID

Pin code
Active

ER diagram for Blood group Personal details

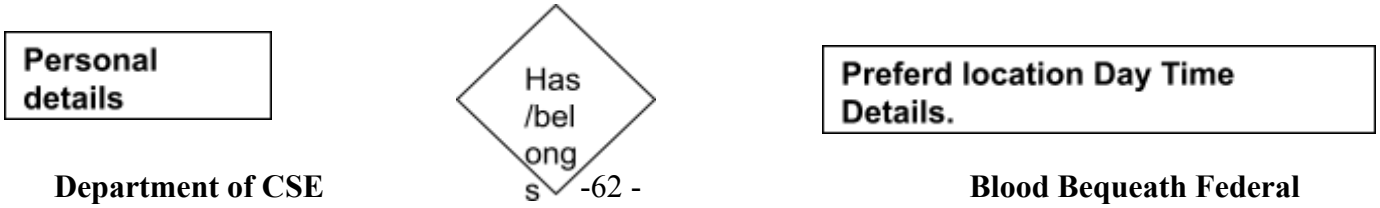


BloodGroup:
BloodGroupID
BloodGroup
Description

Active
Personal details:
UserAccountID
FirstName
MiddleName
LastName
Email
DOB
Weight
Gender
ImageURL
BloodGroupID
BloodType
AddressID
ContactNo_Office
ContactNo_Residence
MobileNo

Active

ER diagram for Personal details and Blood Donation preferences

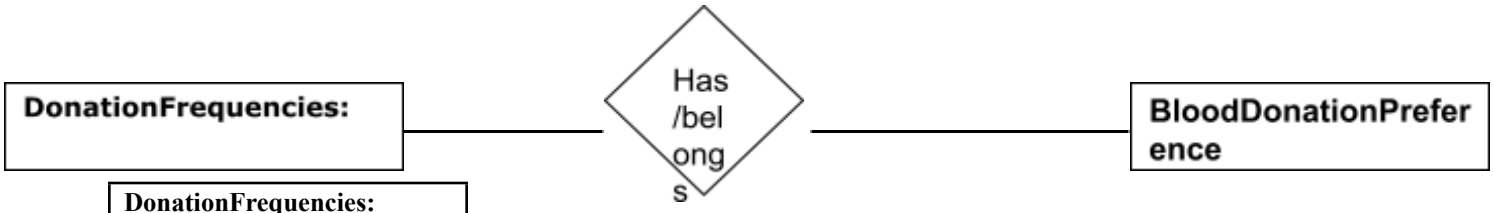


Personal details:
UserAccountID
FirstName
MiddleName
LastName
Email
DOB
Weight
Gender
ImageURL
BloodGroupID
BloodType
AddressID
ContactNo_Office
ContactNo_Residence
MobileNo
Active

PreferredDonationDayTime
DonorPreferenceID

WeekDay
TimeFrom
TimeFrom
TimeUpto
UserAccountID
Comment
Active

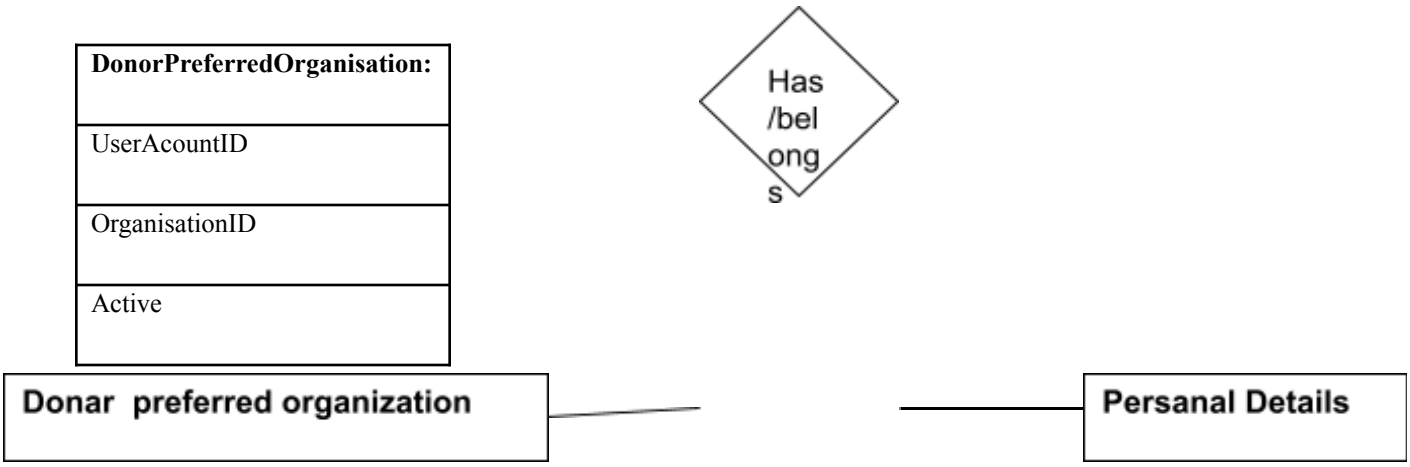
ER diagram for Donor Frequencies and Blood Donation Preferences



DonationFrequencies:
FrequencyID
Frequency
Description
Active

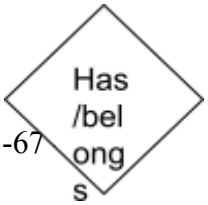
BloodDonationPreference:
PreferenceID
UserAccountID
DonationFrequencyID
WantToDonateWhitecells
Active

ER diagram for Donor preferred organization and personal details



Personal details:
UserAccountID
FirstName
MiddleName
LastName
Email
DOB
Weight
Gender
ImageURL
BloodGroupID
BloodType
AddressID
ContactNo_Office
ContactNo_Residence
MobileNo
Active

ER diagram for Organization type and Organization .



OrganisationType:
TypeID
TypeName
TypeDescription
OrgImage
Active
Organisation:
OrgID
OrgName
OrgType
Email
OrgAddrID
OrgImageURL
OrgDescription
ContactNo
MobileNo
Active
Comment

5.5. Database Tables

1.Entities

- BDA_BloodDonationDetails
- BDA_BloodDonationPreference
- BDA_BloodGroup
- BDA_BloodRequest
- BDA_BloodType
- BDA_City
- BDA_Country

- BDA_DonationFrequencies
- BDA_DonorPreferredOrganisation
- BDA_EmployeeDetail
- BDA_FAQ
- BDA_Location
- BDA_Organisation
- BDA_OrganisationType
- BDA_PersonalDetails
- BDA_PreferredDonationDayTime
- BDA_State
- BDA_UserAccount
- BDA_UserRole

2.Entities with Attributes

1. BDA_Address
    - AddressID
    - AddressLine1
    - LocationID
    - CityID
    - StateID
    - CountryID
    - Zipcode
    - Active

2. BDA_BloodDonationDetails
    - ID
    - DonorID
    - DonationDate
    - TakerID
    - OrgID
    - Quantity
    - Comment
    - Active
3. BDA_BloodDonationPreference
    - PreferenceID
    - UserAccountID
    - DonationFrequencyID
    - WantToDonateWhitecells
    - Active
4. BDA_BloodRequest
    - RequestID
    - DonorId
    - OrgId
    - Name
    - Email
    - Phone
    - Country
    - State
    - City
    - Location
    - BloodRequireAddress
    - BloodType
    - BloodGroup
    - AppDate
    - ReqDate

- Status
- RequestType

5. BDA_BloodGroup
    - BloodGroupID
    - BloodGroup
    - Description
    - Active
6. BDA_BloodType
    - BloodTypeID
    - TypeName
    - TypeDesc
    - Active
7. BDA_City
    - CityID
    - CityName
    - CityDesc
    - CityCode
    - StateID
    - Active
8. BDA_Country
    - CountryID
    - CountryName
    - CountryDesc
    - CountryCode
    - Active

9. BDA_DonationFrequencies
    - FrequencyID
    - Frequency
    - Description

- Active

10. BDA_DonorPreferredOrganisation
    - UserAcountID
    - OrganisationID
    - Active
11. BDA_EmployeeDetail
    - EmpId
    - Name
    - Address
    - Phone
    - Email
    - Active
12. BDA_FAQ
    - FaqID
    - Question
    - Answer
    - Active
13. BDA_Location
    - LocationID
    - LocationName
    - LocationDesc
    - LocationCode
    - CityID
    - Pincode
    - Active

14. BDA_Organisation
    - OrgID
    - OrgName
    - OrgType
    - Email

- OrgAddrID
- OrgImageURL
- OrgDescription
- ContactNo
- MobileNo
- Active
- Comment

15. BDA_OrganisationType
    - TypeID
    - TypeName
    - TypeDescription
    - OrgImage
    - Active
16. BDA_PersonalDetails
    - UserAccountID
    - FirstName
    - MiddleName
    - LastName
    - Email
    - DOB
    - Weight
    - Gender
    - ImageURL
    - BloodGroupID
    - BloodType
    - AddressID
    - ContactNo_Office
    - ContactNo_Residence
    - MobileNo
    - Active
17. BDA_PreferredDonationDayTime

- DonorPreferenceID
- WeekDay
- TimeFrom
- TimeUpto
- UserAccountID
- Comment
- Active

18. BDA_State
    - StateID
    - StateName
    - StateCode
    - StateDesc
    - CountryID
    - Active
19. BDA_UserAccount
    - AccountID
    - Username
    - Password
    - UserDesc
    - HintQuestion
    - Answer
    - RoleID
    - Active

2.Data Dictionary

BDA_ Address

Sno	Column name	Data type	Constraint	reference
1	AddressID	Bigint	Primary Key	

2	AddressLine1	varchar(512)	Not null	
3	LocationID	Bigint	Foreign key	BDA_Location
4	CityID	Bigint	Foreign key	BDA_City
5	StateID	Bigint	Foreign key	BDA_state
6	CountryID	Bigint	Allow Null	BDA_Country
7	Zipcode	varchar(10)	Allow Null	
8	Active	Bit	Not Null	

BDA_BloodDonationDetails

Sno	Column name	Datatype	Constraint	reference
1	ID	Bigint	Primary Key	
2	DonorID	Bigint	Foreign key	BDA_BloodRequest
3	DonationDate	Datetime	Notnull	
4	TakerID	Bigint	Foreign key	BDA_UserAccount
5	OrgID	Bigint	Foreign key	Bda_Organisation
6	Quantity	varchar(10)	Allow Null	
7	Comment	varchar(1024)	Allow Null	
8	Active	Bit	Notnull	

BDA_BloodDonationPreference

Sno	Columnname	Datatype	Constraint	Reference
1	PreferenceID	bigint	Primary key	

2	UserAccountID	bigint	Foreign key	BDA_UserAccount
3	DonationFrequencyID	tinyint	Foreign key	BDA_Donation Frequencies
4	WantToDonateWhitecells	bit	AllowNull	
5	Active	bit	AllowNull	

BDA_BloodGroup

Sno	Columnname	Datatype	Constraint	reference
1	BloodGroupID	tinyint	Primary Key	
2	BloodGroup	varchar(10)	Not Null	
3	Description	varchar(1024)	Allow Null	
4	Active	bit	Not Null	

BDA_BloodRequest

Sno	Columnname	Datatype	Constraint	Reference
1	RequestID	int	Primary Key	
2	DonorId	bigint	Foreign key	BDA_UserAccount
3	OrgId	bigint	Foreign key	BDA_Organisation
4	Name	varchar(50)	Allow Null	
5	Email	varchar(50)	Allow Null	
6	Phone	varchar(20)	Allow Null	
7	Country	varchar(50)	Allow Null	
8	State	varchar(50)	Allow Null	
9	City	varchar(50)	Allow Null	
10	Location	varchar(50)	Allow Null	
11	BloodRequireAddress	varchar(100)	Allow Null	
12	BloodType	varchar(50)	Allow Null	
13	BloodGroup	varchar(50)	Allow Null	

14	AppDate	datetime	Allow Null	
15	ReqDate	datetime	Allow Null	
16	Status	varchar(10)	Allow Null	
17	RequestType	varchar(50)	Allow Null	

BDA_City

Sno	Columnname	Datatype	Constraint	Reference
1	CityID	bigint	Primary key	
2	CityName	varchar(100)	Not null	
3	CityDesc	varchar(1024)	allownull	
4	CityCode	varchar(5)	Allownull	
5	StateID	bigint	Foreign key	BDA_State
6	Active	bit	notnull	

BDA_BloodType

Sno	Columnnames	Datatypes	Constraint	reference
1	BloodTypeID	tinyint	Primary Key	
2	TypeName	varchar(50)	Not null	
3	TypeDesc	varchar(1024)	AllowNull	
4	Active	bit	Notnull	

BDA_Country

Sno	Columnname	Datatype	Constraint	Reference
1	CountryID	bigint	Primary key	

2	CountryName	varchar(100)	Not null	
3	CountryDesc	varchar(1024)	Allow Null	
4	CountryCode	varchar(5)	Allow Null	
5	Active	bit	Not null	

BDA_DonationFrequencies

Sno	Columnname	Datatype	Constraint	Reference
1	FrequencyID	tinyint	Primary key	
2	Frequency	varchar(50)	Not null	
3	Description	varchar(255)	Allow null	
4	Active	bit	Allow null	

BDA_DonorPreferredOrganisation

Sno	Columnname	Datatype	Constraint	Reference
1	UserAcountID	bigint	Primary key	
2	OrganisationID	bigint	Not null	
3	Active	bit	Allow null	

BDA_EmployeeDetail

Sno	Columnname	Datatype	Constraint	reference
1	EmpId	bigint	Primary Key	
2	Name	varchar(60)	Allow Null	
3	Address	varchar(150)	Allow Null	
4	Phone	varchar(20)	Allow Null	
5	Email	varchar(25)	Allow Null	
6	Active	tinyint	Allow Null	

BDA_FAQ

Sno	Columnname	Datatype	Constraint	reference
1	FaqID	bigint	Primary key	
2	Question	varchar(300)	Allow null	
3	Answer	varchar(1000)	Allow null	
4	Active	bit	Allow null	

BDA_Location

Sno	Columnname	Datatype	Constraints	reference
1	LocationID	bigint	Primary Key	
2	LocationName	varchar(100)	Not Null	
3	LocationDesc	varchar(1024)	Allow null	
4	LocationCode	varchar(5)	Allow null	
5	CityID	bigint	Allow null	
6	Pincode	varchar(10)	Allow null	
7	Active	bit	Allow null	

BDA_Organisation

Sno	Columnname	Datatype	Constraint	reference
1	OrgID	bigint	Primary key	
2	OrgName	varchar(100)	Not null	

3	OrgType	tinyint	Foreign key	BDA_Organisation Type
4	Email	varchar(50)	Allownull	
5	OrgAddrID	bigint	Foreign key	BDA_Address
6	OrgImageURL	varchar(155)	Allownull	
7	OrgDescription	varchar(1024)	Allownull	
8	ContactNo	varchar(20)	Allownull	
9	MobileNo	varchar(20)	Allownull	
10	Active	bit	Allownull	
11	Comment	varchar(512)	Allownull	

BDA_OrganisationType

Sno	Columnname	Datatype	Constraint	Reference
1	TypeID	tinyint	Primary key	
2	TypeName	varchar(50)	Not null	
3	TypeDescription	varchar(50)	Allow null	
4	OrgImage	varchar(300)	Allow null	
5	Active	bit	Allow null	

BDA_PreferredDonationDayTime

Sno	Columnnames	Datattype	Constarint	Reference
1	DonorPreferenceID	bigint	Primary Key	
2	WeekDay	varchar(10)	Not null	
3	TimeFrom	varchar(10)	Allow null	
4	TimeUpto	varchar(10)	Allow null	
5	UserAccountID	bigint	Allow null	
6	Comment	varchar(1024)	Allow null	
7	Active	bit	Not null	
8				

BDA_PersonalDetails

Sno	Columnname	Datatype	Constraint	Reference
1	UserAccountID	bigint	Primary Key	

2	FirstName	varchar(50)	Not Null	
3	MiddleName	varchar(50)	Allow Null	
4	LastName	varchar(50)	Allow Null	
5	Email	varchar(100)	Allow Null	
6	DOB	datetime	Notnull	
7	Weight	float	Allow Null	
8	Gender	varchar(6)	Not Null	
9	ImageURL	varchar(155)	Allow Null	
10	BloodGroupID	tinyint	Foreign key	BDA_Bloodgroup
11	BloodType	tinyint	Foreign key	BDA_BloodType
12	AddressID	bigint	Foreign key	BDA_Address
13	ContactNo_Office	varchar(20)	Allow Null	
14	ContactNo_Residence	varchar(20)	Allow Null	
15	MobileNo	varchar(20)	Not Null	
16	Active	bit	Allow Null	

BDA_UserRole

Sno	Columnname	DAtatype	Constraint	Reference
1	RoleID	tinyint	Primary key	
2	RoleName	varchar(50)	Not null	
3	RoleDesc	varchar(1024)	Allow null	
4	Active	bit	Not null	

BDA_State

Sno	Columnname	Datatype	Constraint	Reference
1	StateID	bigint	Primary key	
2	StateName	varchar(100)	Not null	
3	StateCode	varchar(5)	Allow null	
4	StateDesc	varchar(1024)	Allow null	
5	CountryID	bigint	Foreign Key	BDA_Country
6	Active	bit	Not null	

BDA_UserAccount

Sno	Columnname	Datatype	Constraint	Reference
1	AccountID	bigint	Primary key	
2	Username	varchar(100)	Not null	
3	Password	varchar(100)	Not null	
4	UserDesc	varchar(1024)	Allow null	
5	HintQuestion	varchar(155)	Allow null	
6	Answer	varchar(155)	Allow null	
7	RoleID	tinyint	Foreign key	BDA_UserRole
8	Active	bit	Not null	

Software Development Environment

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.

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.

.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.

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.

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

  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).

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.

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.

- **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.
- **Data Adapters**. For pushing data into a **DataSet**, and reconciling data against a database.

).

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.

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.

CODING

WEBCONFIG FILE:(Design Code)

Used to set the connections of each page.

```
<?xml version="1.0"?>
<!--
    Note: As an alternative to hand editing this file you can
use the
    web admin tool to configure settings for your application.
Use
    the Website->Asp.Net Configuration option in Visual Studio.
    A full list of settings and comments can be found in
    machine.config.comments usually located in
    \Windows\Microsoft.Net\Framework\v2.x\Config
-->
<configuration>
     <appSettings>
          <add key="ConnStr" value="data
source=RAMYA-2DCA5B123;database=BloodBequeathFederalAgent;integr
ated security=sspi"/>
     </appSettings>
     <connectionStrings>
  <add name="BloodDonationAgentConnectionString"
connectionString="Data Source=RAMYA-2DCA5B123;Initial
Catalog=BloodDonationAgent;integrated security=sspi"
   providerName="System.Data.SqlClient" />
 </connectionStrings>
     <system.web>
          <!--
            Set compilation debug="true" to insert debugging
            symbols into the compiled page. Because this
            affects performance, set this value to true only
            during development.
        -->
          <compilation debug="true">
               <assemblies>
                    <add assembly="System.Design,
Version=2.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A"/>
                    <add assembly="System.Web.Extensions,
Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
                    <add assembly="System.Web.Extensions.Design,
Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
```

```
                    <add assembly="System.Windows.Forms,
Version=2.0.0.0, Culture=neutral,
PublicKeyToken=B77A5C561934E089"/></assemblies></compilation>
          <!--
            The <authentication> section enables configuration
            of the security authentication mode used by
            ASP.NET to identify an incoming user.
        -->
            <authentication mode="Windows"/>
            <!--
            The <customErrors> section enables configuration
            of what to do if/when an unhandled error occurs
            during the execution of a request. Specifically,
            it enables developers to configure html error pages
            to be displayed in place of a error stack trace.

        <customErrors mode="RemoteOnly"
defaultRedirect="GenericErrorPage.htm">
            <error statusCode="403" redirect="NoAccess.htm" />
            <error statusCode="404" redirect="FileNotFound.htm"
/>
        </customErrors>
        -->
    </system.web>
</configuration>
```

User Login Form

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class Login : System.Web.UI.Page
{
    CheckUser user = new CheckUser();
    UserAccountBusinessLayer account = new UserAccountBusinessLayer();
    OrganizationAccountBusinessLayer org = new
OrganizationAccountBusinessLayer();
    protected void Page_Load(object sender, EventArgs e)
    {
        txtUsername.Focus();
    }

    protected void btnLogin_Click(object sender, EventArgs e)
    {
```

```
try
{
    user.Username = txtUsername.Text;
    user.Password = txtPassword.Text;
    //Check User

    if (user.GetUser() == true)
    {
        account.Username = txtUsername.Text;
        DataSet ds = new DataSet();
        ds = account.GetAccountId();
        string AcId = ds.Tables[0].Rows[0][0].ToString();
        Session["username"] = txtUsername.Text;
        Session["Acid"] = AcId;
        DataSet ds1 = new DataSet();
        account.Accountid =int.Parse(AcId);
        ds1 = account.GetAddressId();
        Session["addid"] = ds1.Tables[0].Rows[0][0].ToString();
        Response.Redirect("~/Donor/DonorHome.aspx");
    }
    else
        Image2.Visible = true;
    lblMsg.Text = "Your Login Attempt Is Failed Plz try Again....!";
    txtPassword.Text = "";
    txtUsername.Focus();
    //Checking Organization
    if (user.GetOrganization() == true)
    {
        account.Username = txtUsername.Text;
        DataSet ds = new DataSet();
        ds = account.GetAccountId();
        string AcId = ds.Tables[0].Rows[0][0].ToString();
        Session["username"] = txtUsername.Text;
        Session["Acid"] = AcId;
        DataSet ds1 = new DataSet();
        org.Orgid =int.Parse(AcId);
        ds1 = org.GetOrgAddressId();
        Session["addid"]=ds1.Tables[0].Rows[0][0].ToString();
        Response.Redirect("~/Organization/OrganizationHome.aspx");
    }
    else
        Image2.Visible = true;
    lblMsg.Text = "Your Login Atempt Is Failed Plz try Again....!";
    txtPassword.Text = "";
    txtUsername.Focus();
    //Employee Checking
    if (user.CheckEmployee() == true)
    {
        account.Username = txtUsername.Text;
        DataSet ds = new DataSet();
        ds = account.GetAccountId();
        string AcId = ds.Tables[0].Rows[0][0].ToString();
        Session["username"] = txtUsername.Text;
        Session["Acid"] = AcId;
        Response.Redirect("~/CallCenter/CallCenterHome.aspx");
    }
```

```
    }
}
```

Testing

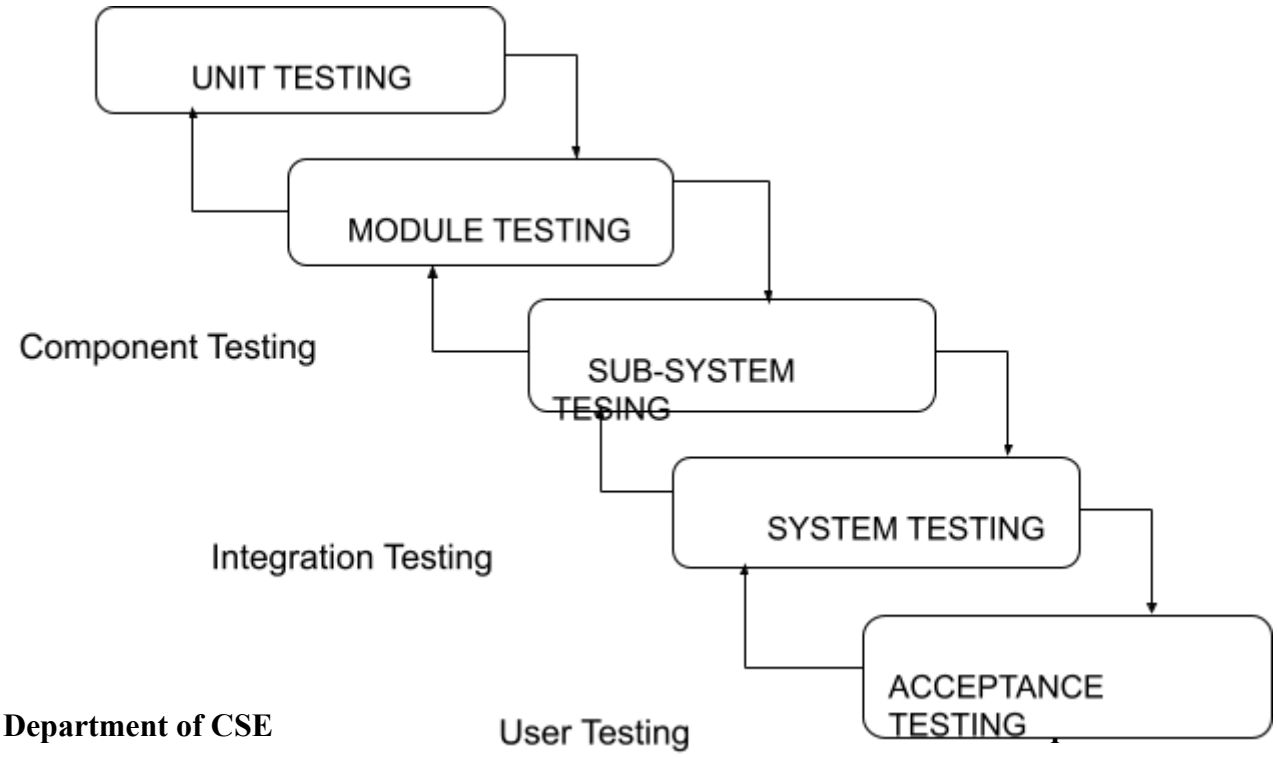
7.1. INTRODUCTION

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. STRATEGIC APPROACH TO 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.



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:

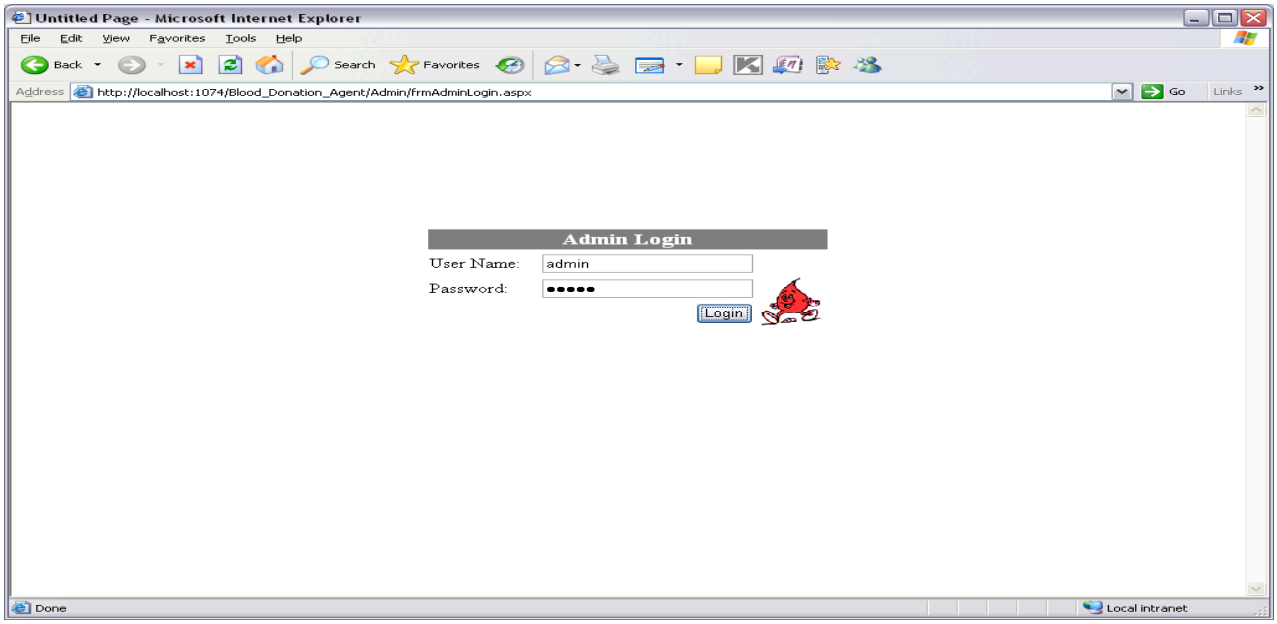
- All the loops were tested at their limits, just above them and just below them.
- All the loops were skipped at least once.
- For nested loops test the inner most loop first and then work outwards.
- For concatenated loops the values of dependent loops were set with the help of connected loop.
- Unstructured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

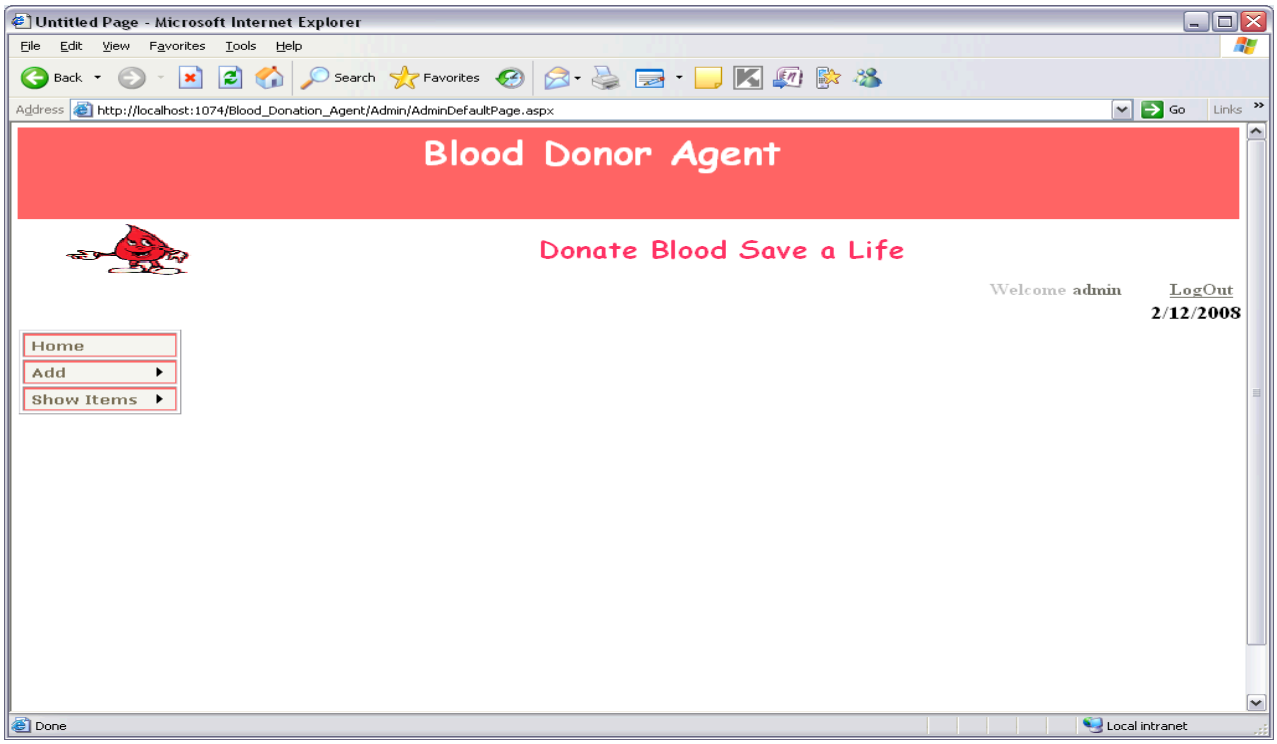
Output Screens

Home Page

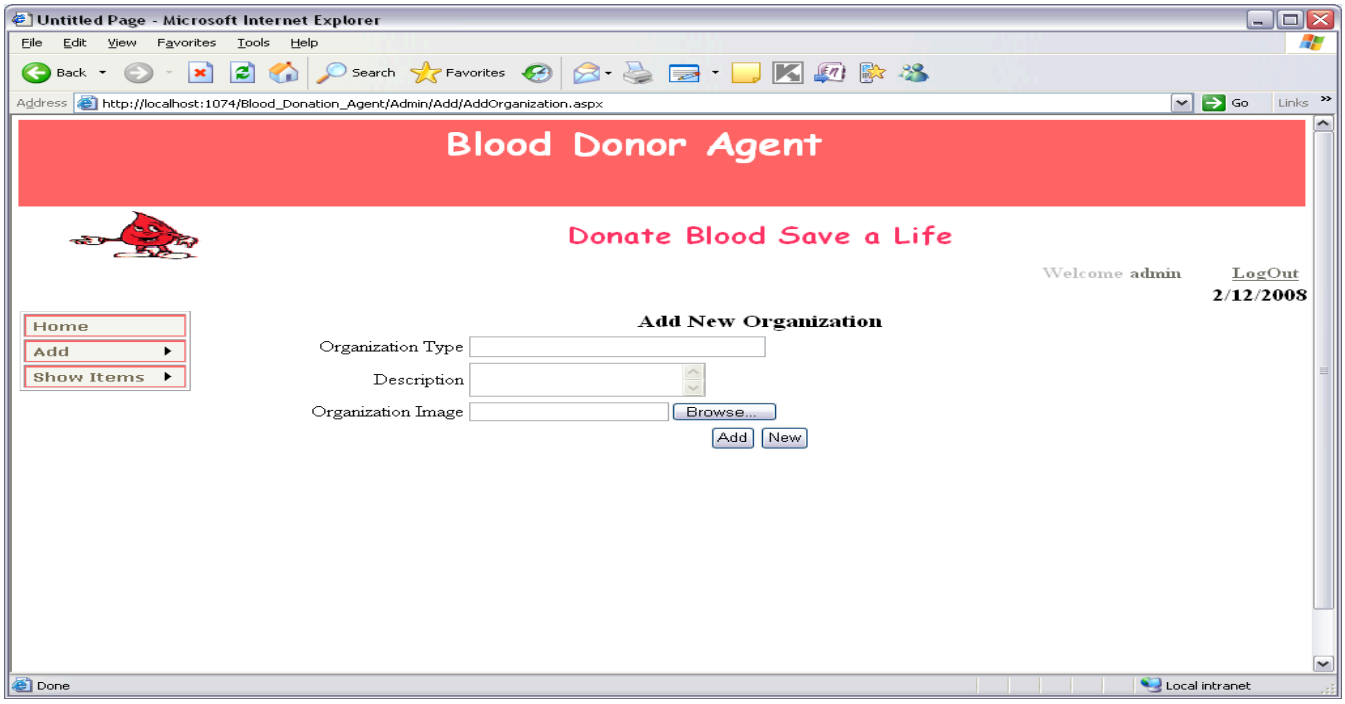
Admin Login



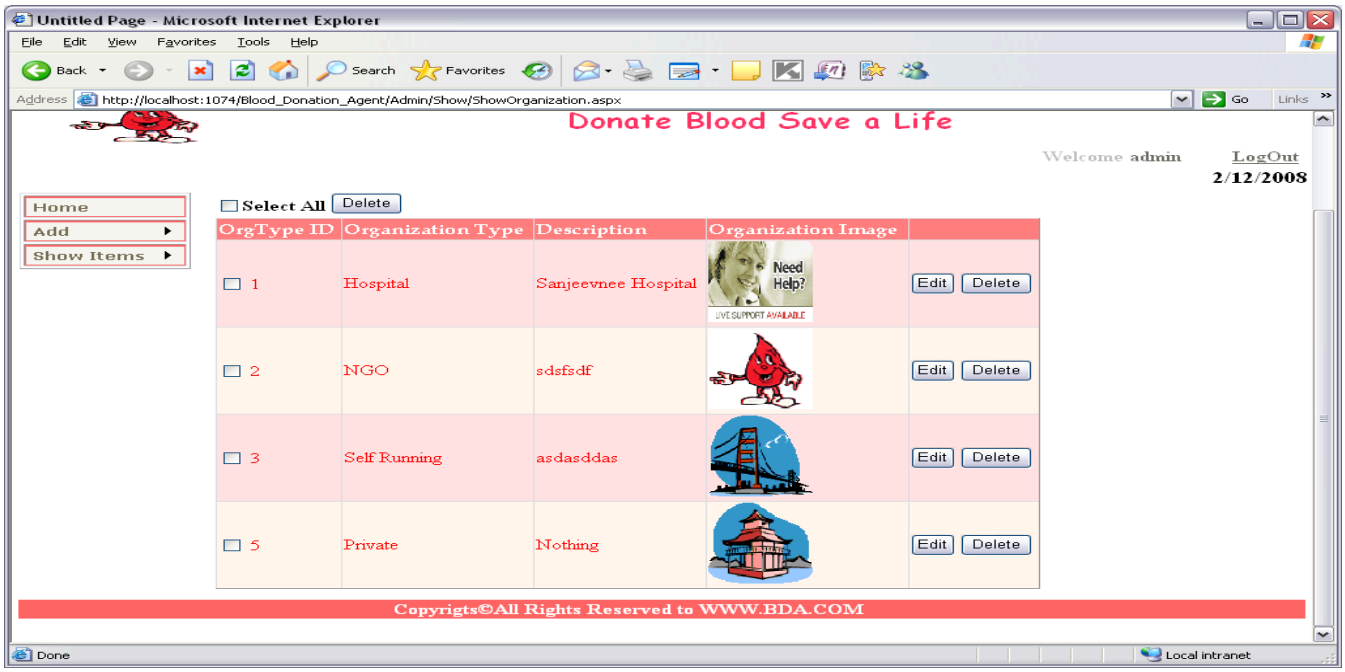
Admin Home



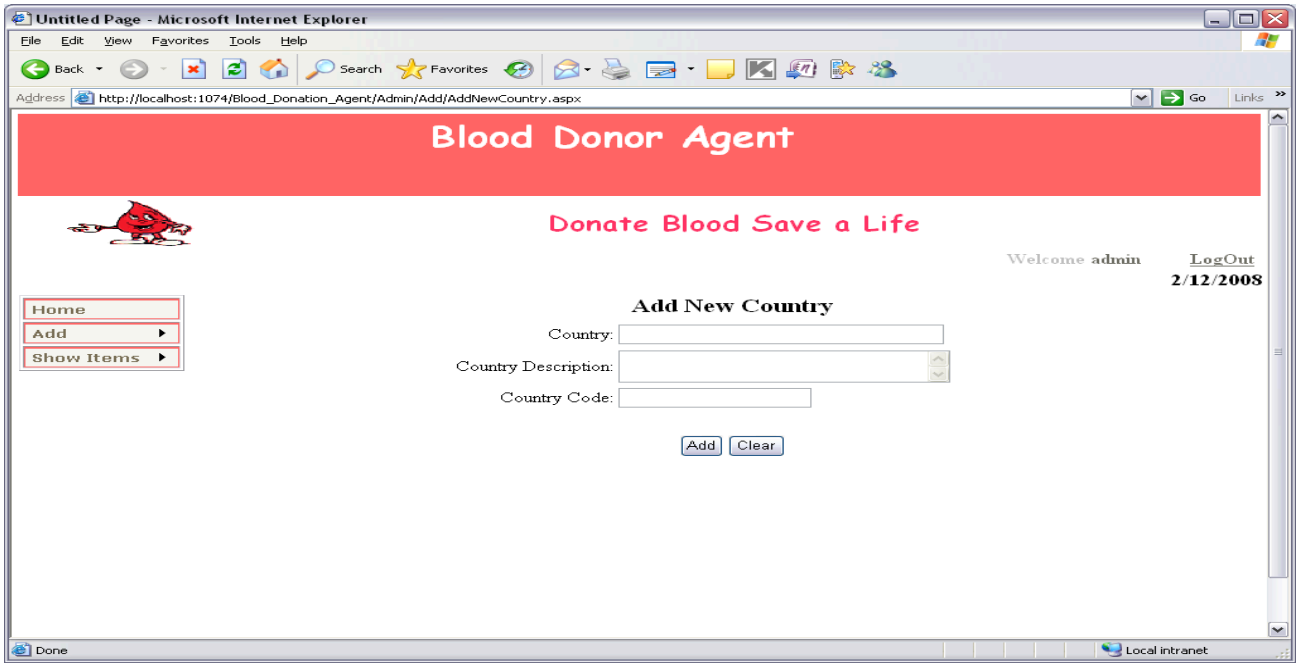
Add new Organization



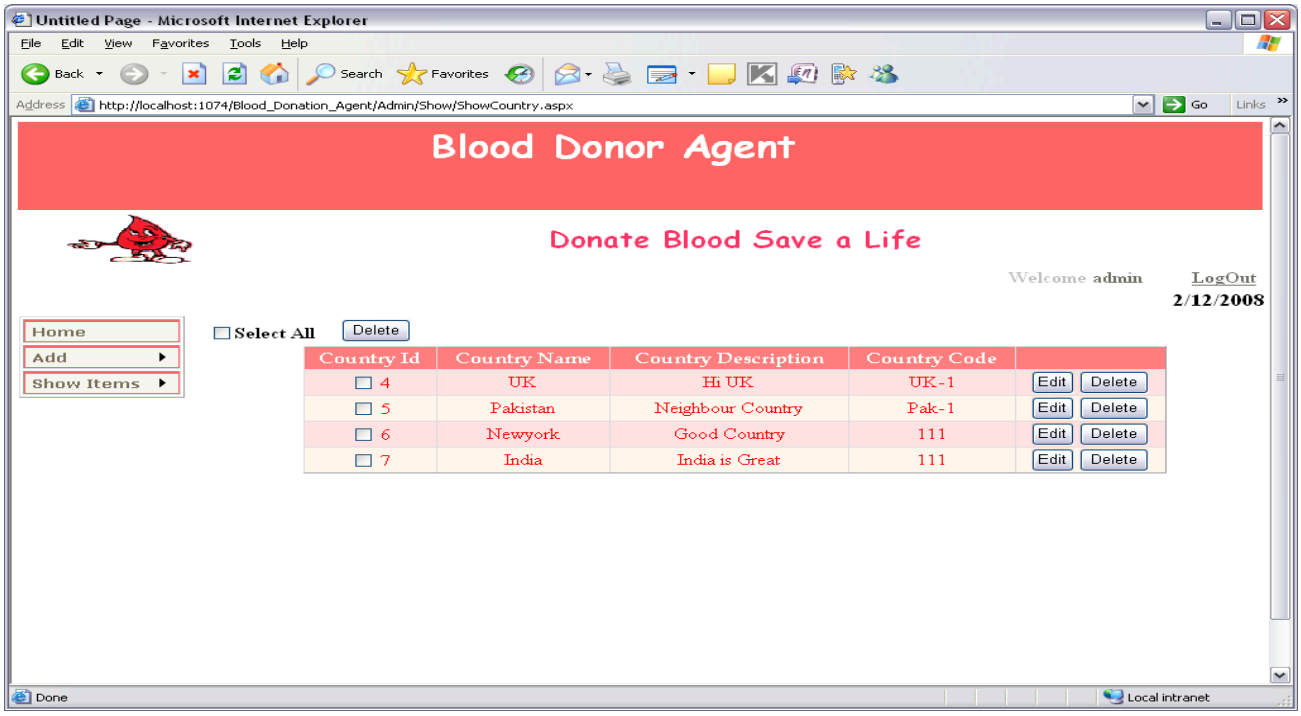
Retrieving Organization information



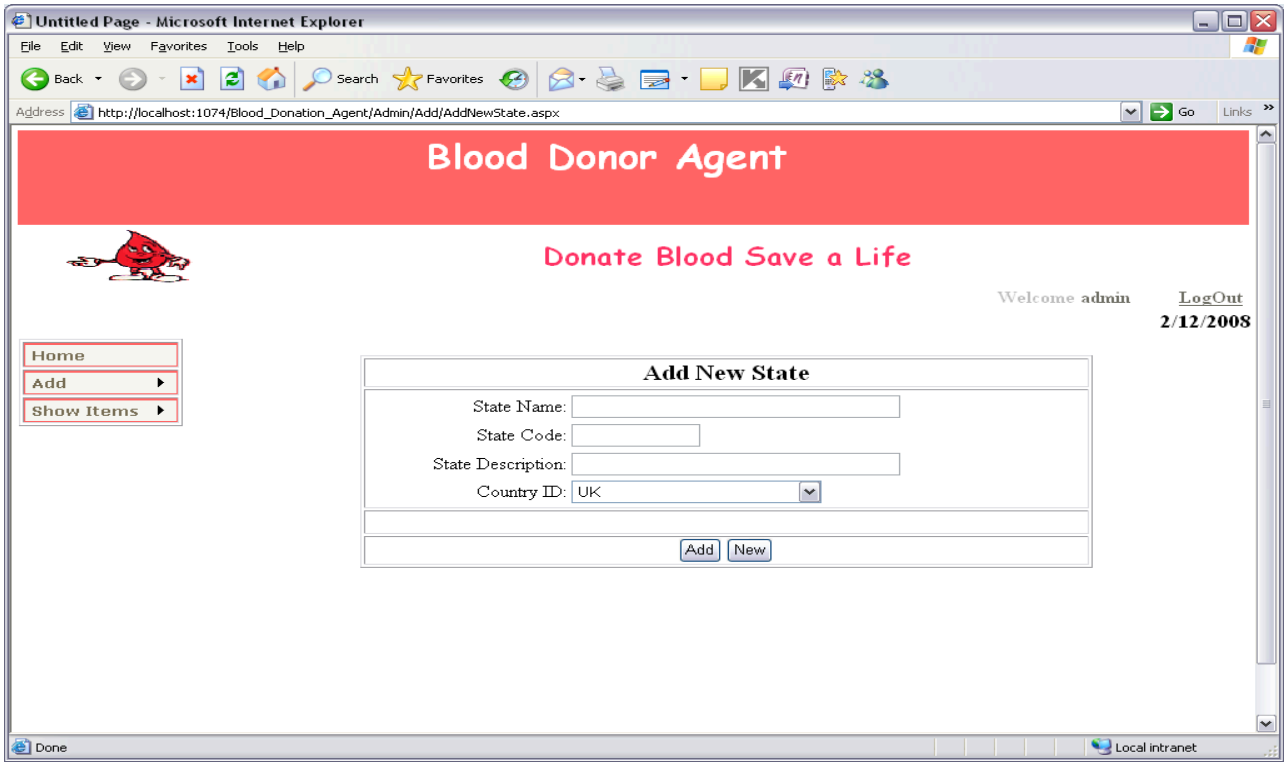
ADD NEW COUNTRY



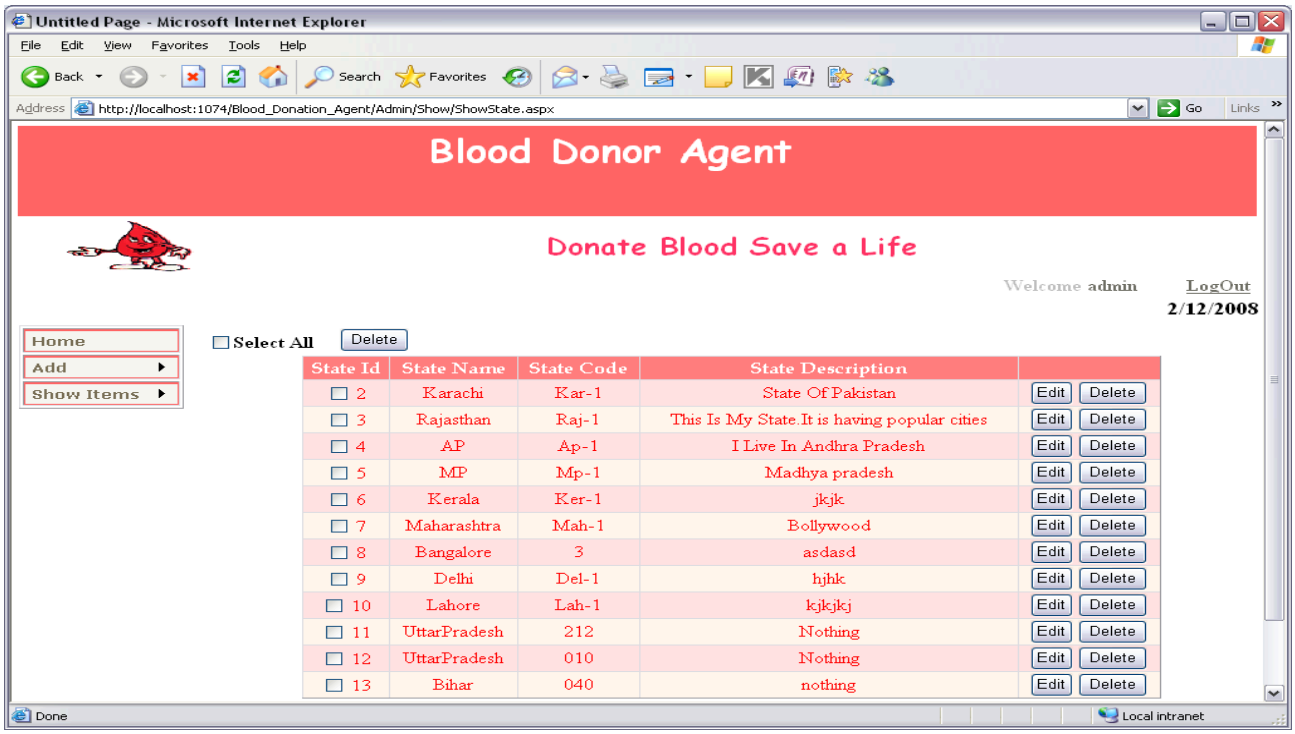
Country Information



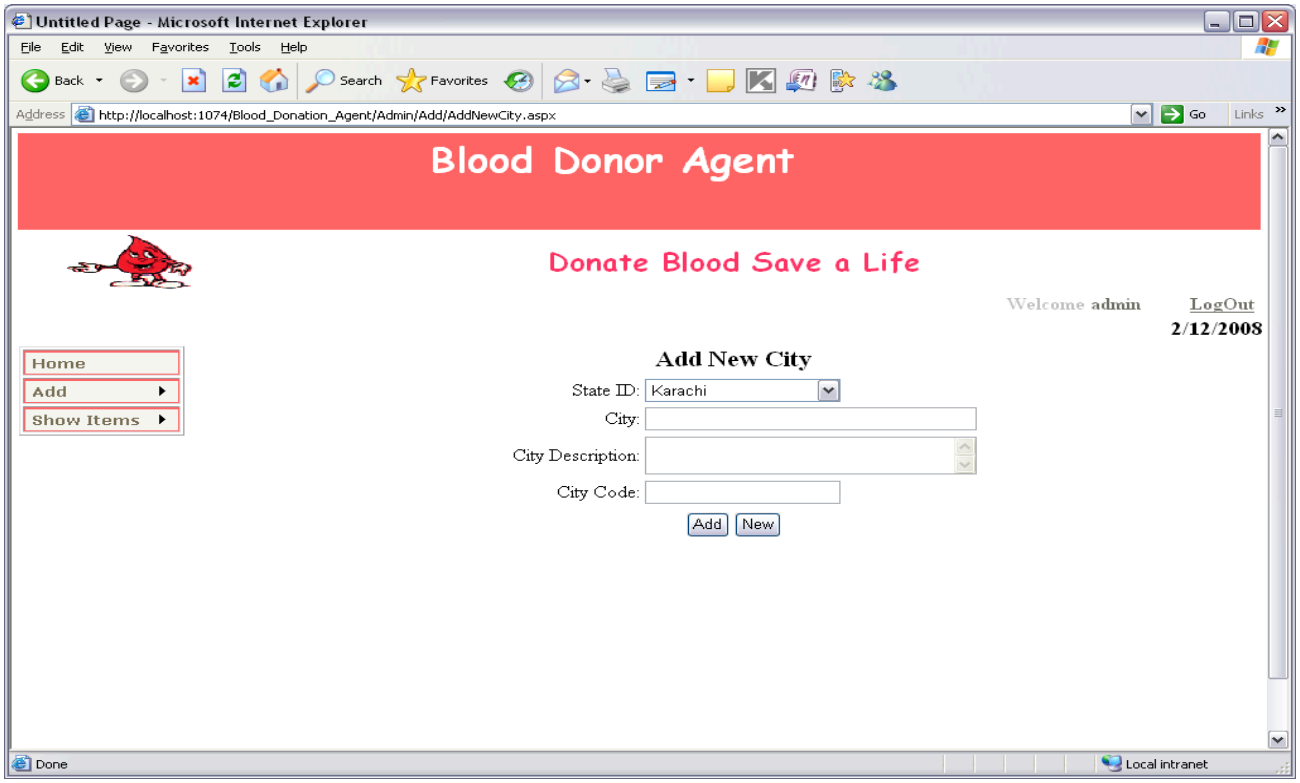
ADD NEW STATE



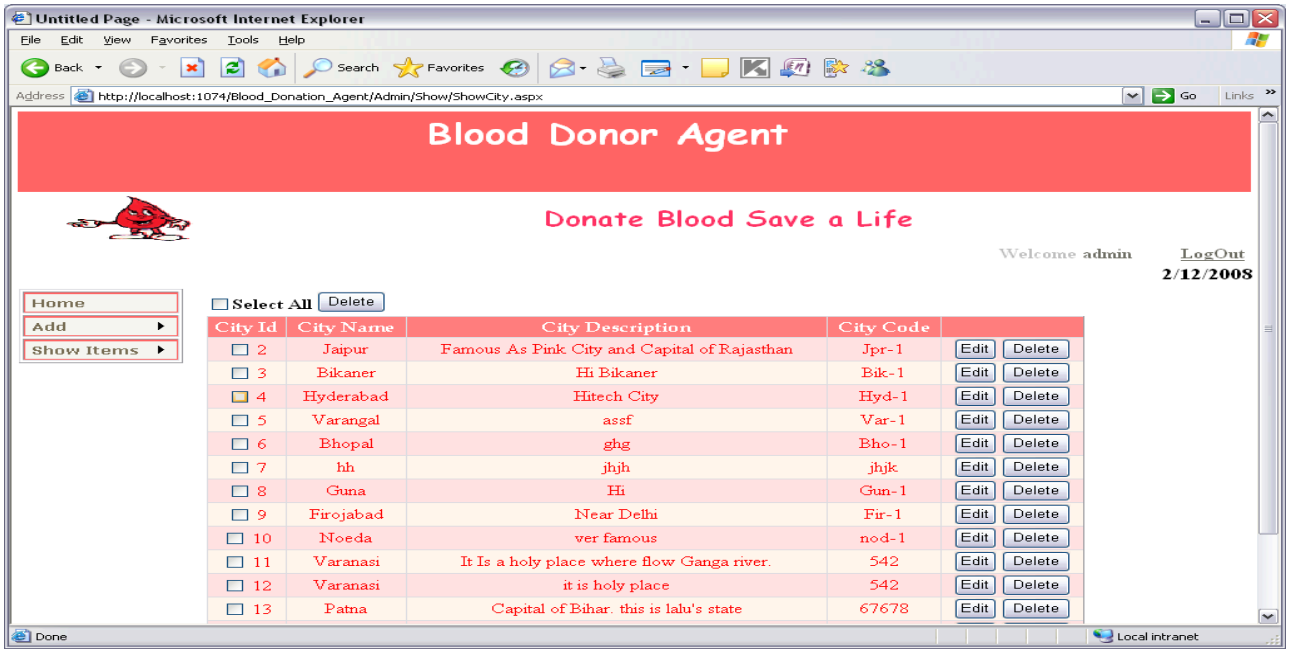
State Information



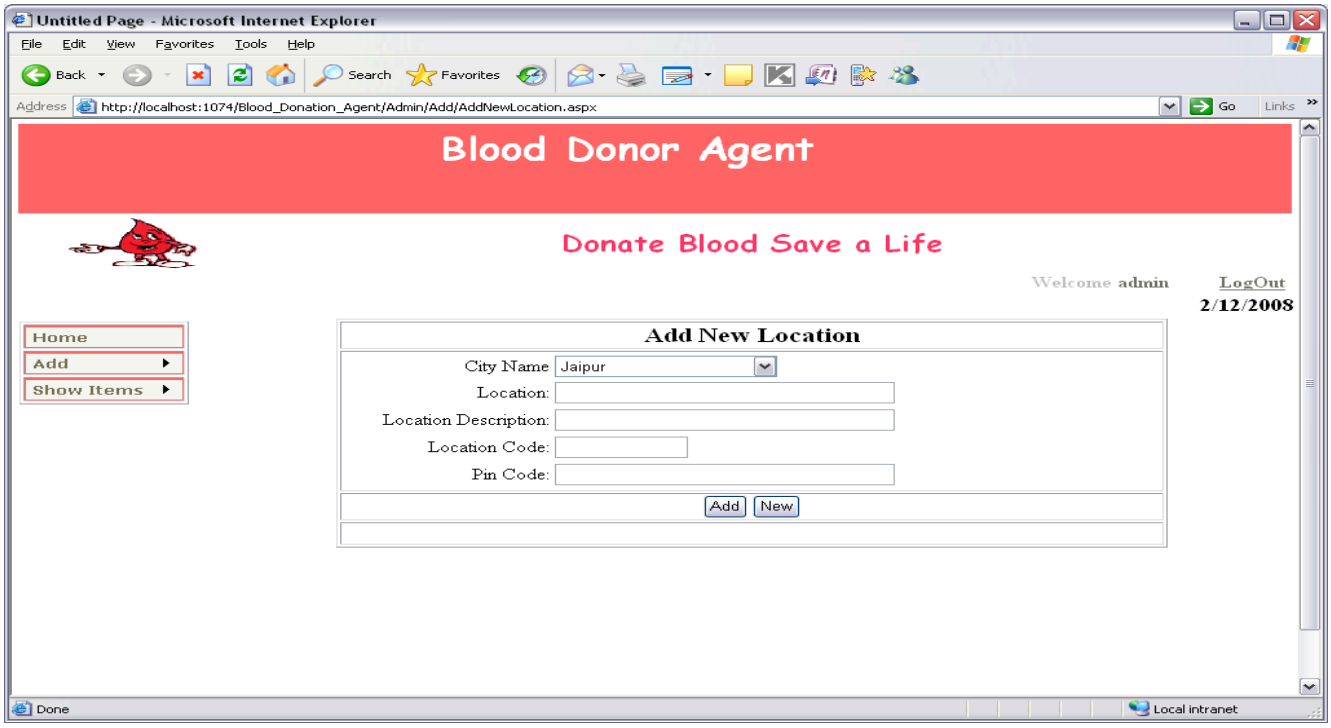
ADD NEW CITY



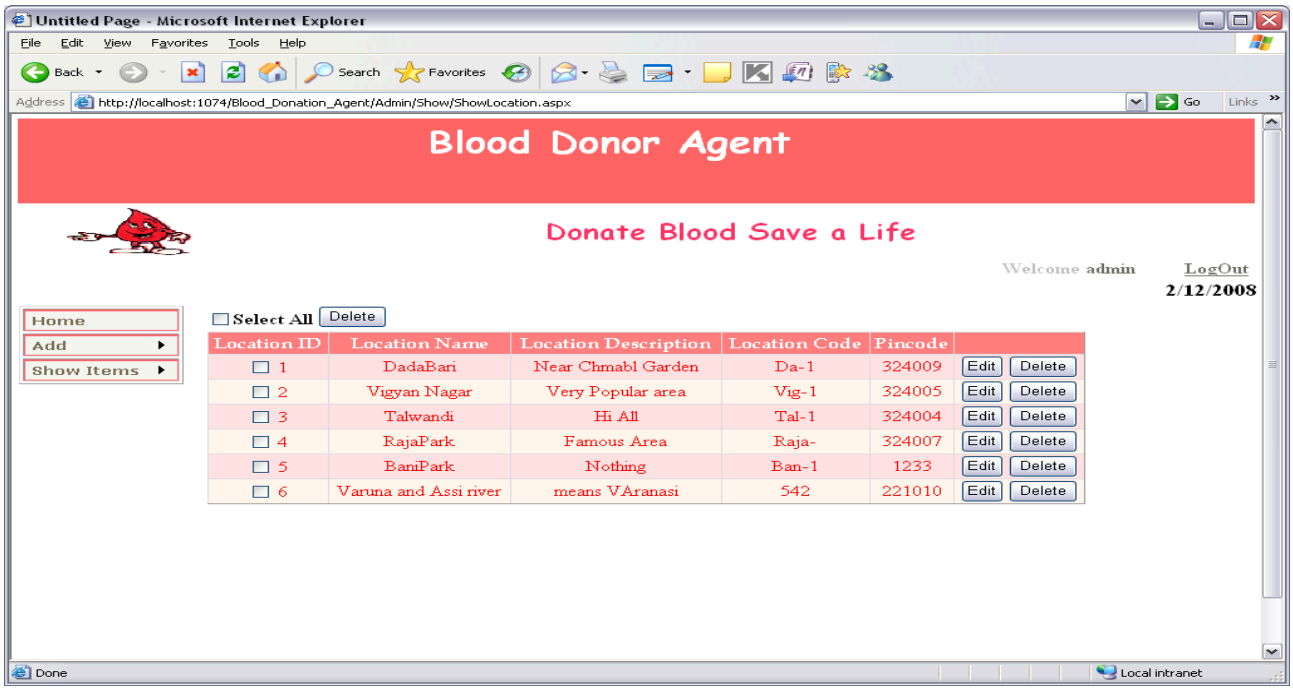
City Information



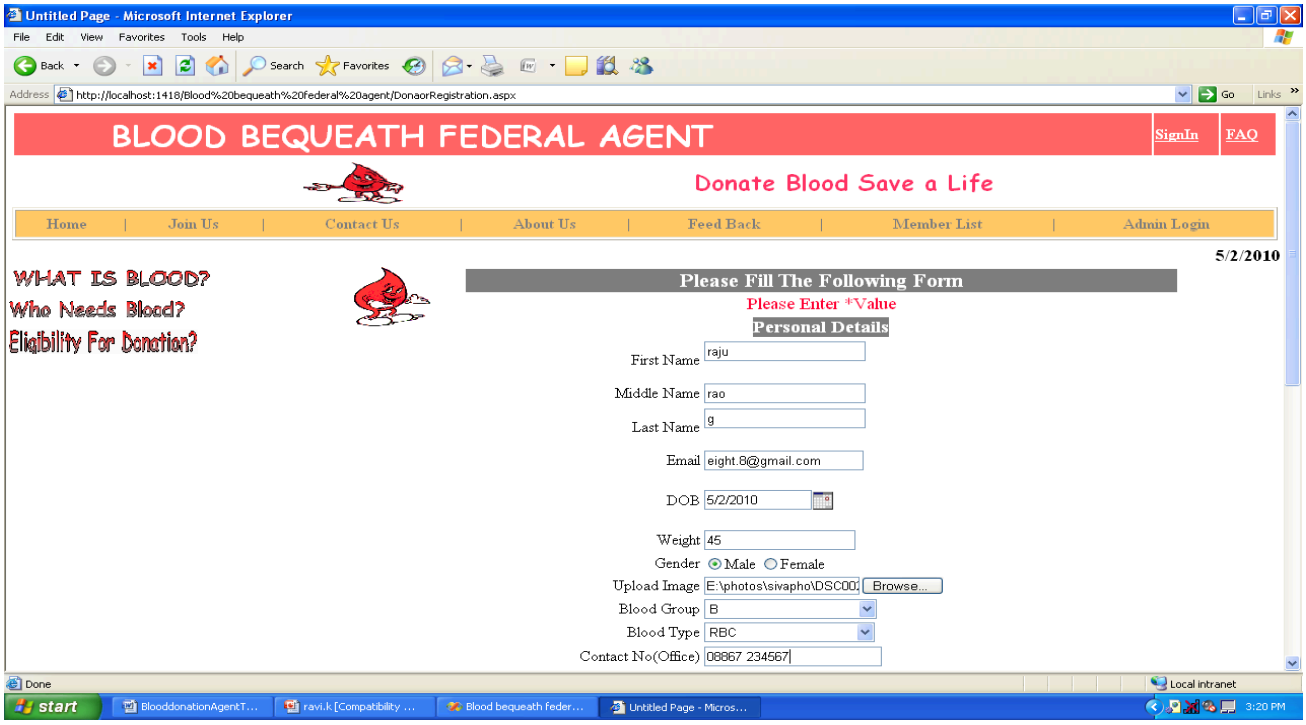
ADD NEW LOCATION



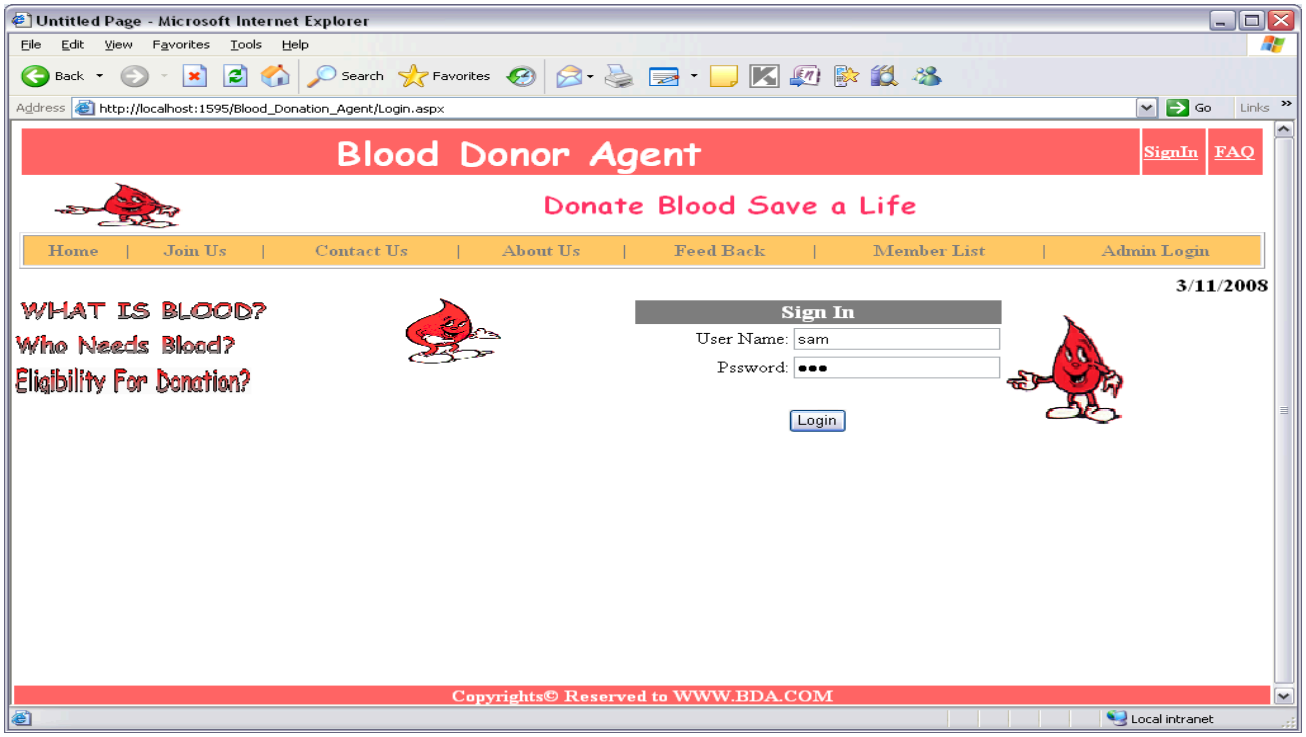
Location Information



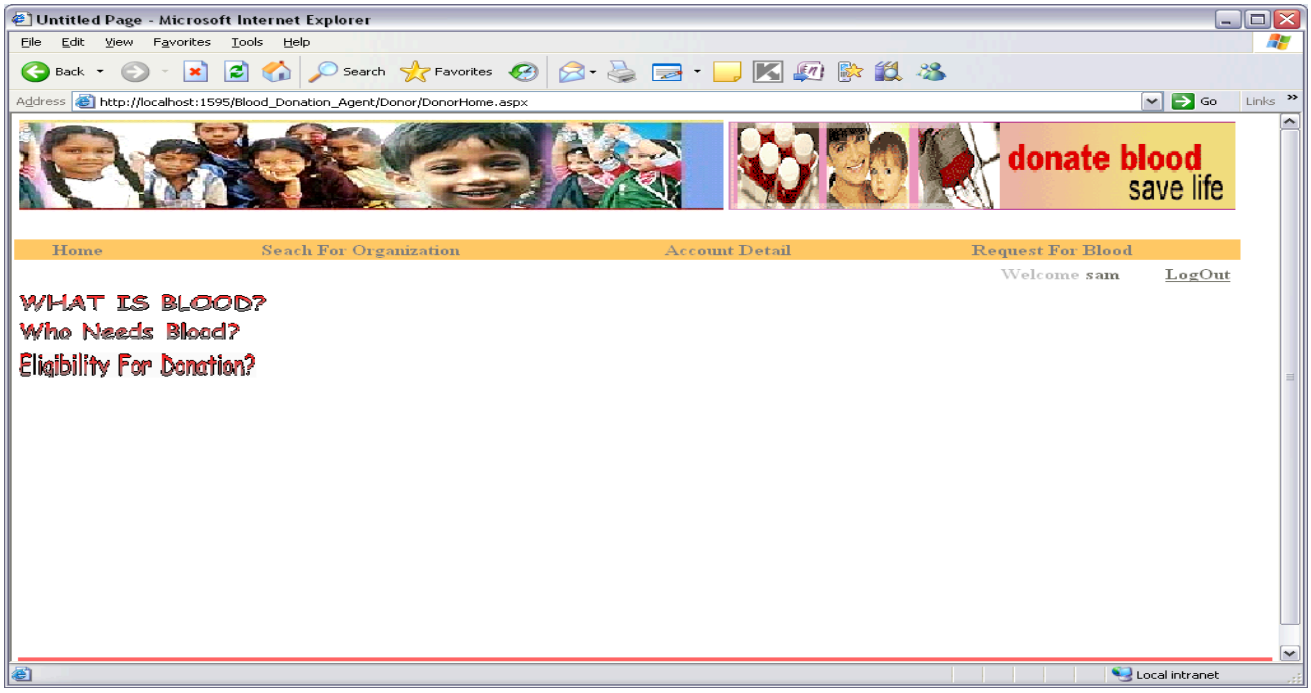
Registration Form For All Users



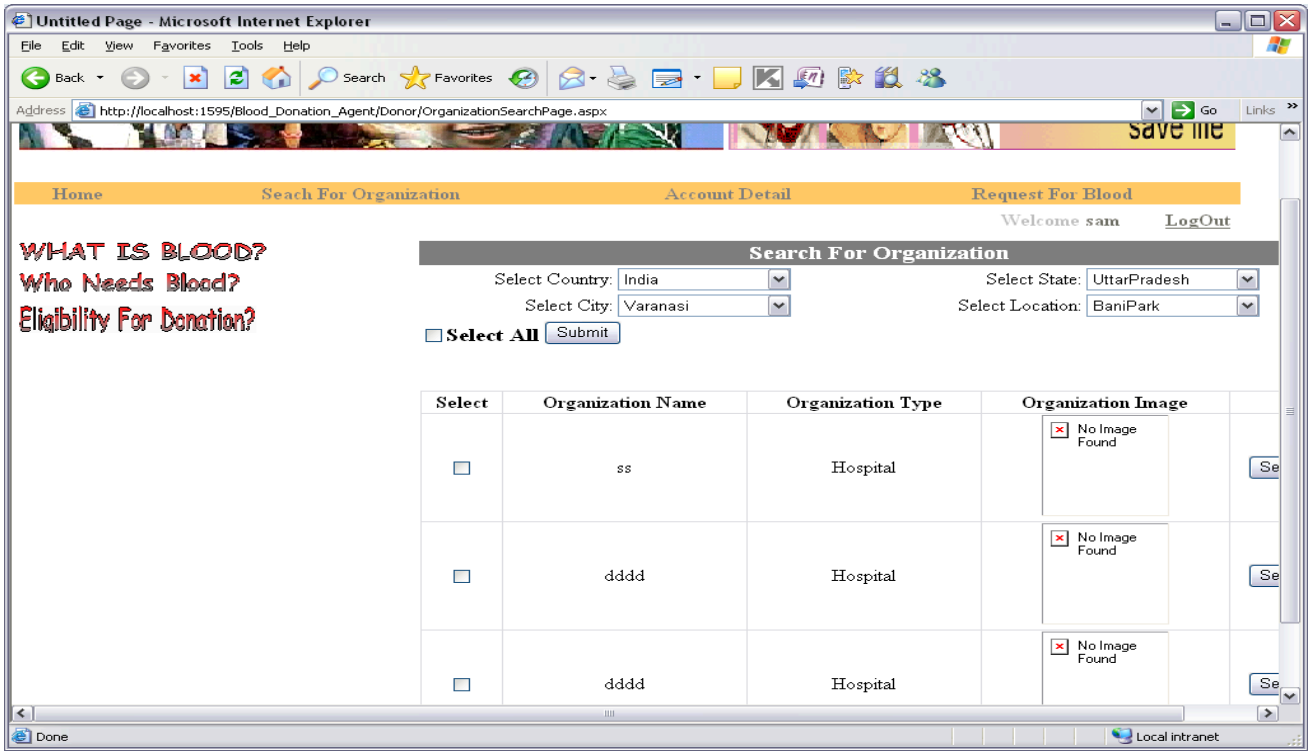
Login form



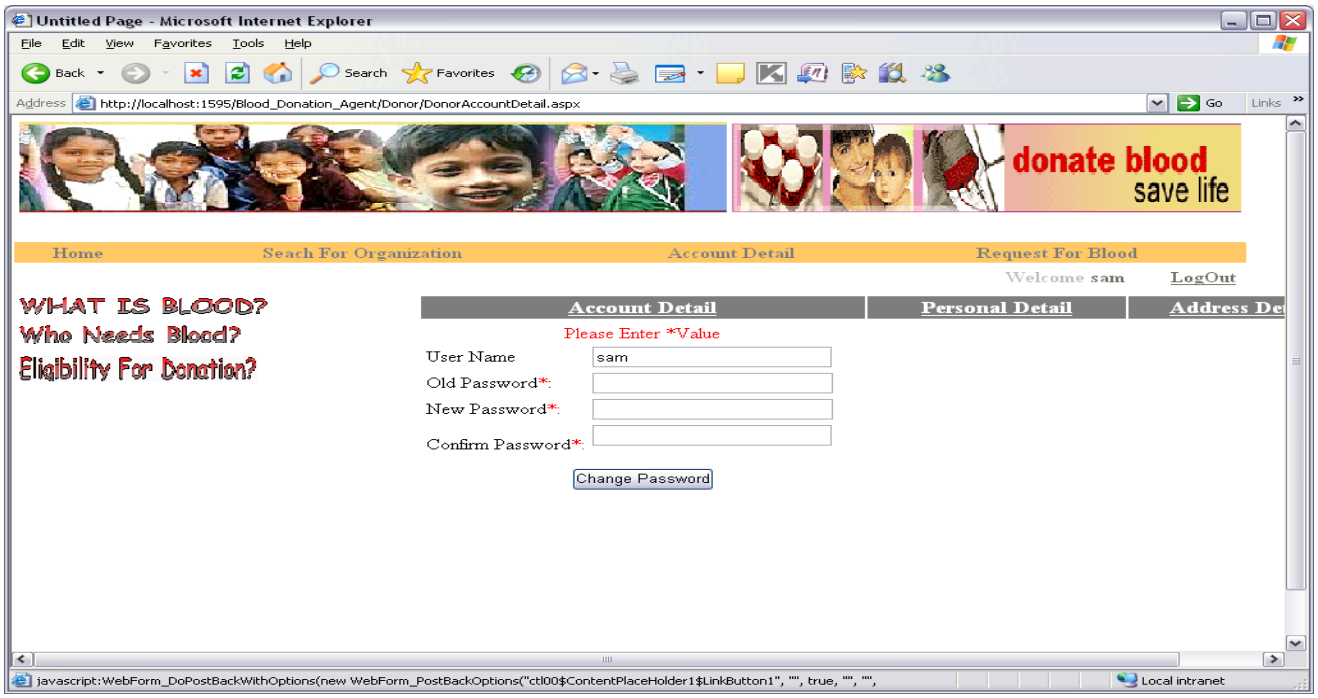
User Home



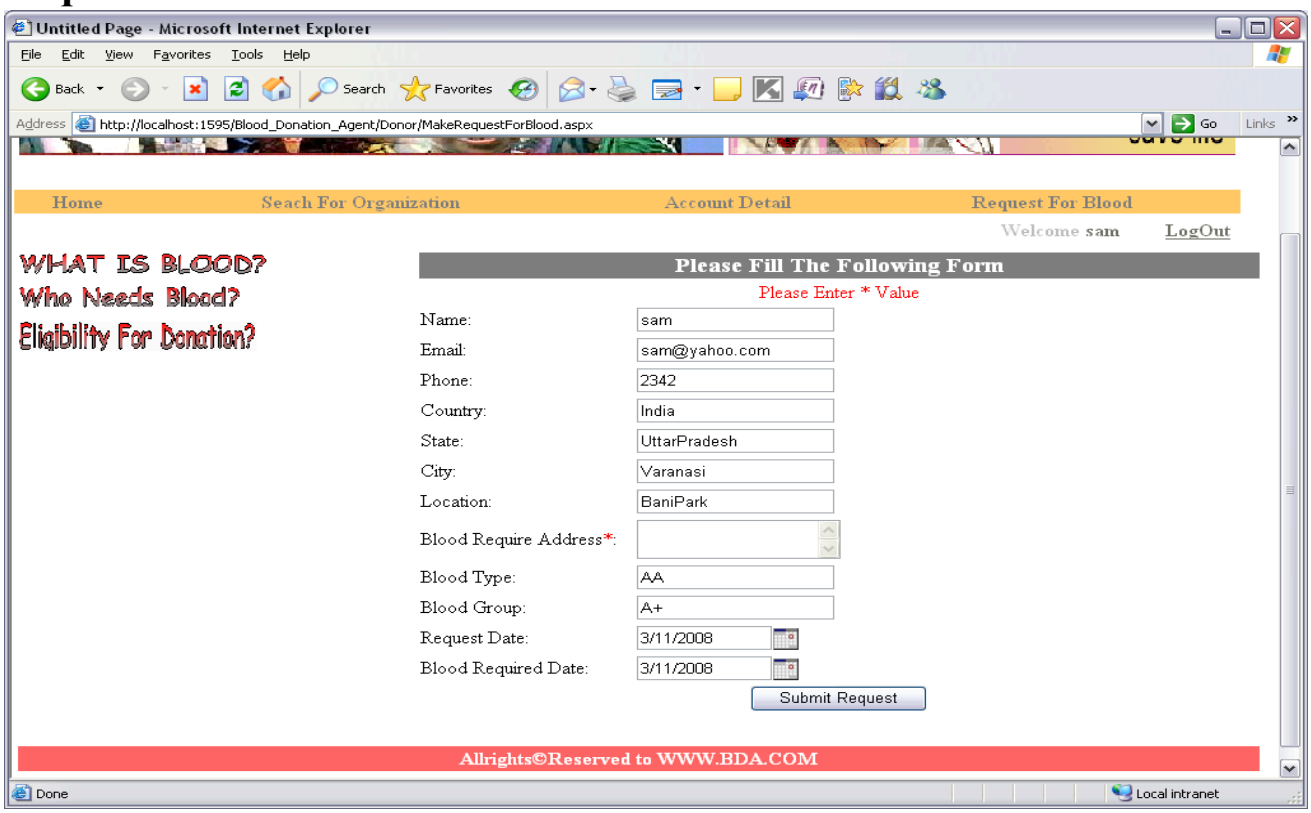
SearchForDonor



Update Account Details



Request for blood



CONCLUSION

It has been a great pleasure for me to work on this exciting and challenging project. This project proved good for me as it provided practical knowledge of not only programming in ASP.NET and VB.NET web based application and no some extent Windows Application and SQL Server, but also about all handling procedure related with **"Blood Bequeath Federal".** It also provides knowledge about the latest technology used in developing web enabled application and client server technology that will be great demand in future. This will provide better opportunities and guidance in future in developing projects independently.

BENEFITS:

The project is identified by the merits of the system offered to the user. The merits of this project are as follows: -

- It's a web-enabled project.
- This project offers user to enter the data through simple and interactive forms. This is very helpful for the client to enter the desired information through so much simplicity.
- The user is mainly more concerned about the validity of the data, whatever he is entering. There are checks on every stages of any new creation, data entry or updation so that the user cannot enter the invalid data, which can create problems at later date.
- Sometimes the user finds in the later stages of using project that he needs to update some of the information that he entered earlier. There are options for him by which he can update the records. Moreover there is restriction for his that he cannot change the primary data field. This keeps the validity of the data to longer extent.
- User is provided the option of monitoring the records he entered earlier. He can see the desired records with the variety of options provided by him.

- From every part of the project the user is provided with the links through framing so that he can go from one option of the project to other as per the requirement. This is bound to be simple and very friendly as per the user is concerned. That is, we can sat that the project is user friendly which is one of the primary concerns of any good project.
- Data storage and retrieval will become faster and easier to maintain because data is stored in a systematic manner and in a single database.
- Decision making process would be greatly enhanced because of faster processing of information since data collection from information available on computer takes much less time then manual system.
- Allocating of sample results becomes much faster because at a time the user can see the records of last years.
- Easier and faster data transfer through latest technology associated with the computer and communication.
- Through these features it will increase the efficiency, accuracy and transparency,

9.1 LIMITATIONS:

- The size of the database increases day-by-day, increasing the load on the database back up and data maintenance activity.
- Training for simple computer operations is necessary for the users working on the system.

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