# RESULTAND DISCUSSION

### Test Reports

|  |  |  |
| --- | --- | --- |
|  |  |  |
| *Test Case Id* | *Comment* | *Status* |
| *BDMS-001* | *NA* | *PASS* |
| *BDMS-002* | *NA* | *PASS* |
| *BDMS-003* | *NA* | *PASS* |
| *BDMS-004* | *NA* | *PASS* |
| *BDMS-005* | *NA* | *PASS* |
| *BDMS-006* | *NA* | *PASS* |
| *BDMS-007* | *NA* | *PASS* |
| *BDMS-008* | *NA* | *PASS* |
| *BDMS-009* | *NA* | *PASS* |
| *BDMS-010* | *NA* | *PASS* |
| *BDMS-011* | *NA* | *PASS* |
| *BDMS-012* | *NA* | *PASS* |
| *BDMS-013* | *NA* | *PASS* |
| *BDMS-014* | *NA* | *PASS* |
| *BDMS-015* | *NA* | *PASS* |
| *BDMS-016* | *NA* | *PASS* |
| *BDMS-017* | *NA* | *PASS* |
| *BDMS-018* | *NA* | *PASS* |
| *BDMS-019* | *NA* | *PASS* |
| *BDMS-020* | *NA* | *PASS* |
| *BDMS-021* | *NA* | *PASS* |
| *BDMS-022* | *NA* | *PASS* |
| *BDMS-023* | *NA* | *PASS* |
| *BDMS-024* | *NA* | *PASS* |
| *BDMS-025* | *NA* | *PASS* |
| *BDMS-026* | *NA* | *PASS* |
| *BDMS-027* | *NA* | *PASS* |
| *BDMS-028* | *NA* | *PASS* |
| *BDMS-029* | *NA* | *PASS* |
| *BDMS-030* | *NA* | *PASS* |
| *BDMS-031* | *NA* | *PASS* |
| *BDMS-032* | *NA* | *PASS* |
| *BDMS-033* | *NA* | *PASS* |
| *BDMS-034* | *NA* | *PASS* |
| *BDMS-035* | *NA* | *PASS* |
| *BDMS-036* | *NA* | *PASS* |
| *BDMS-037* | *NA* | *PASS* |
| *BDMS-038* | *NA* | *PASS* |
| *BDMS-039* | *NA* | *PASS* |
| *BDMS-040* | *NA* | *PASS* |
| *BDMS-041* | *NA* | *PASS* |
| *BDMS-042* | *NA* | *PASS* |
| *BDMS-043* | *NA* | *PASS* |
| *BDMS-044* | *NA* | *PASS* |
| *BDMS-045* | *NA* | *PASS* |
| *BDMS-046* | *NA* | *PASS* |
| *BDMS-047* | *NA* | *PASS* |
|  |  |  |
| *BDMS-048* | *NA* | *PASS* |
| *BDMS-049* | *NA* | *PASS* |
| *BDMS-050* | *NA* | *PASS* |
| *BDMS-051* | *NA* | *PASS* |
| *BDMS-052* | *NA* | *PASS* |
| *BDMS-053* | *NA* | *PASS* |
| *BDMS-054* | *NA* | *PASS* |
| *BDMS-055* | *NA* | *PASS* |
| *BDMS-056* | *NA* | *PASS* |
| *BDMS-057* | *NA* | *PASS* |
| *BDMS-058* | *NA* | *PASS* |
| *BDMS-059* | *NA* | *PASS* |
| *BDMS-060* | *NA* | *PASS* |

## User Documentation

Donor:

*All donor functionalities can be done here. User can add donor with details. Can view the donor details, which is already exists indatabase. Editing of donor details is also available here.*

PATIENT:

*Patient register for blood, view registered patient and if necessary modification of patient details can be done here. After giving successful service a patient can be removed from the data base.*

MEMBER:

*All member of the organization can be managed here. Here user can add, viewand delete the members. Modification of existing member is also possible.*

Well-WISHER:

*It is necessary to maintain well-wishers details for a voluntary organization .here the application can keep track of the well-wishers. Addition, viewing, deletion and modification of well-wishers are an easy task.*

FUNDS:

*Here the user can maintain fund information’sdetails. Whenever a well-wisher wishes to donate money for fund it is easy for the user to accept and update the fund into the database. Here the users don’t need to put the well wishers name manually. Adropdown menu of well-wishers name will appear .From where users just need to select the well wishers name and update other necessary in formations.*

*Figure out all the expenses are also done in fund.Addition, viewing, deletion and modification of fund are done here.*

FUNDS:

*User can save, edit and delete the job should be done in upcoming days.*

EVENTS:

*Keep accounts of events that will be organized by the organization is very much important. Within event user can maintain the history as well as the upcoming events. Modification of events is also possible here.*

DONOR ASSIGNMENT:

*It is mandatory to assign which donor will donate blood for a particular patient .Here, within assign donor, the job is done. Whenever user wish to assign a donor, he just need to select the patient and all the patient information will be generated and user just need to select a donor name from a drop down list.*

Help:

*Within help there some important information like address of a hospital or blood bank or doctors can be stored.Modifications of existing information are also possible.*

*This project is developed in such a way that it will be self-dependent. This real time useable application is user-friendly and robust. The unique concept and implementation will make this application more acceptable.*

*We wanted to select a project that not only includes challenging work but also provides us a clear concept of real life project and applicability of languages and technologies we are using for completing the project. We chose C sharp as our programming language as it is one of the best and most used programming language in the IT industry. We used WPF, the latest GUI technology of Microsoft to provide an easy to use and splendid user interface. MYSQL is used here which is the most popular (and free) DBMS software in the world. We started the work keeping in mind that we must develop a flexible and fast application so that it becomes popular among the users. Our modular approach has made it easier to understand, easy to debug application. We can reuse the codes in different applications as well. Our vigorous testing has made it an error free application. It is password protected so all the data kept in the application is secured. With a few improvements and modifications it could be a very useful application in real time.*

## Limitation Of the system

*Blood Donor Management System is not a substitute of blood bank or blood donation camps. This is a parallel eco-system along with other existing system to improve & utilize the blood donation process. This application alone can’t fulfill all requirements for this purpose.*

## Purpose and Scope

### Purpose

The main purpose of Blood Donor Management System is to bring up a unified process of blood donation and better utilize the donated blood. Below diagram shows the different stake holders of**BDMS**.



The main features of Blood Donor Management System:

1. Secure user login and data access.
2. Computerized records of Donor details and Patient requirements.
3. Query & update donor information from various clients like Desktop.
4. Manage Well wishers, to do, Members etc.
5. Manage the financial transactions & donations.
6. Manage Helpful information like Hospital, Doctor, Blood Bank details etc.

### Scope

1. Blood Donor Management System is not a substitute of blood bank or blood donation camps. This is a parallel eco-system along with other existing system to improve & utilize the blood donation process.
2. This application will be developed for Windows operating system (Win7, Win XP) only; in future we are planning to make it runable under LINUX, MAC operating system also.
3. In future we would extend it to make it run able under mobile operating systems like Android, iOS or Windows Mobile OS.
4. Our web client will be developed using Google App Framework & Google Doc interface. Web client & Mobile client will not be synced automatically; it will require a manual sync with the server.
5. It is under a continuous process of development and we are working hard to make it perfect and error free project.

### Applicability

1. Blood Donor Management System will create a bridge between blood donor and patient. Any voluntary organization or clubs, whoever wishes to serve people by bringing the blood donors under the same roof, can be benefited by this application. Using this application will be more time efficient and take less effort. The application will make it easy to find out the perfect donor for a patient. The application will fulfill all the needs for this propose.

## Achievements

1. After the completion we gain overall working knowledge on C#, XAML, SQL PLUS. We also achieved knowledge on Microsoft Visual C# 2010.
2. The project was started with a goal of managing the blood donors, patients, well-wishers, events related this topic, managing funds and some miscellaneous task related with this topic. There were also some goals like make the application runable on the internet by giving the database into an online server, synchronizing with mobile phones. The goals are partially achieved for now and it is under a continuous developing process.

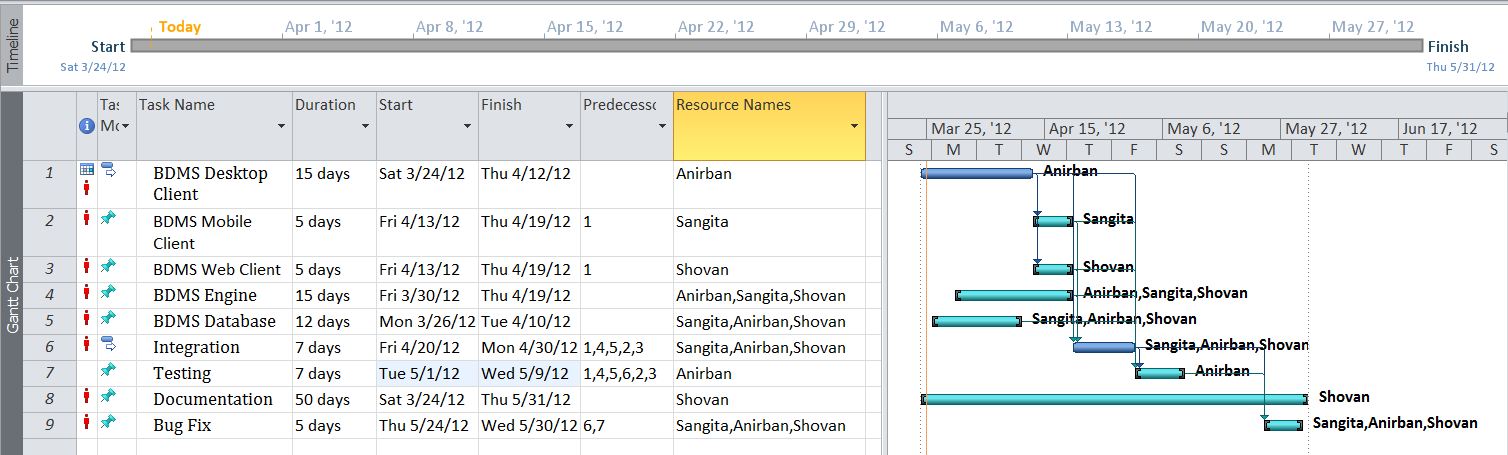
## Organization of report

1. Here after, this report will carry
2. Survey of technology.
3. Requirement analysis.
4. Planning and scheduling.
5. System design.
6. Data design
7. Logic diagrams.
8. Testing reports.
9. Snapshots.

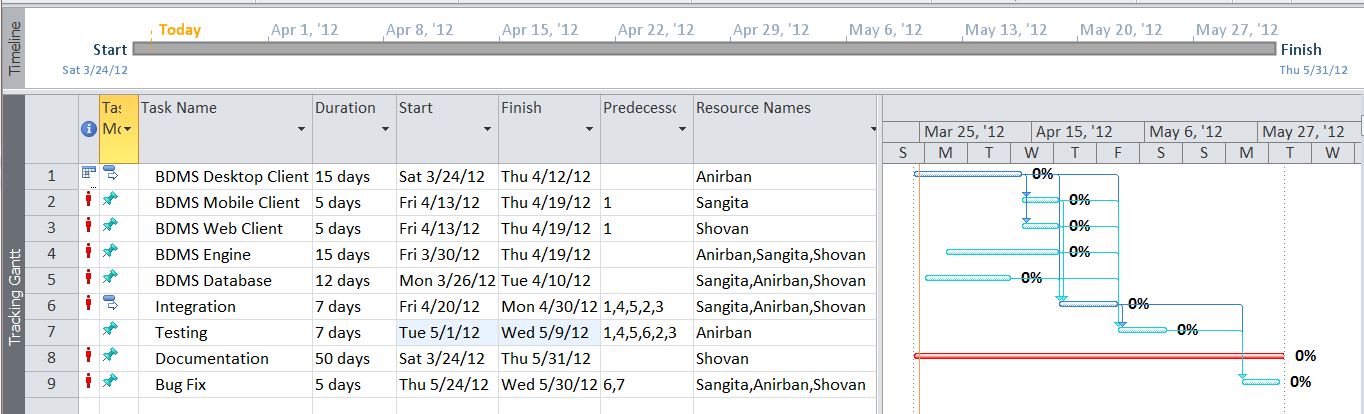
### Non-functional Requirements

## Planning and Scheduling

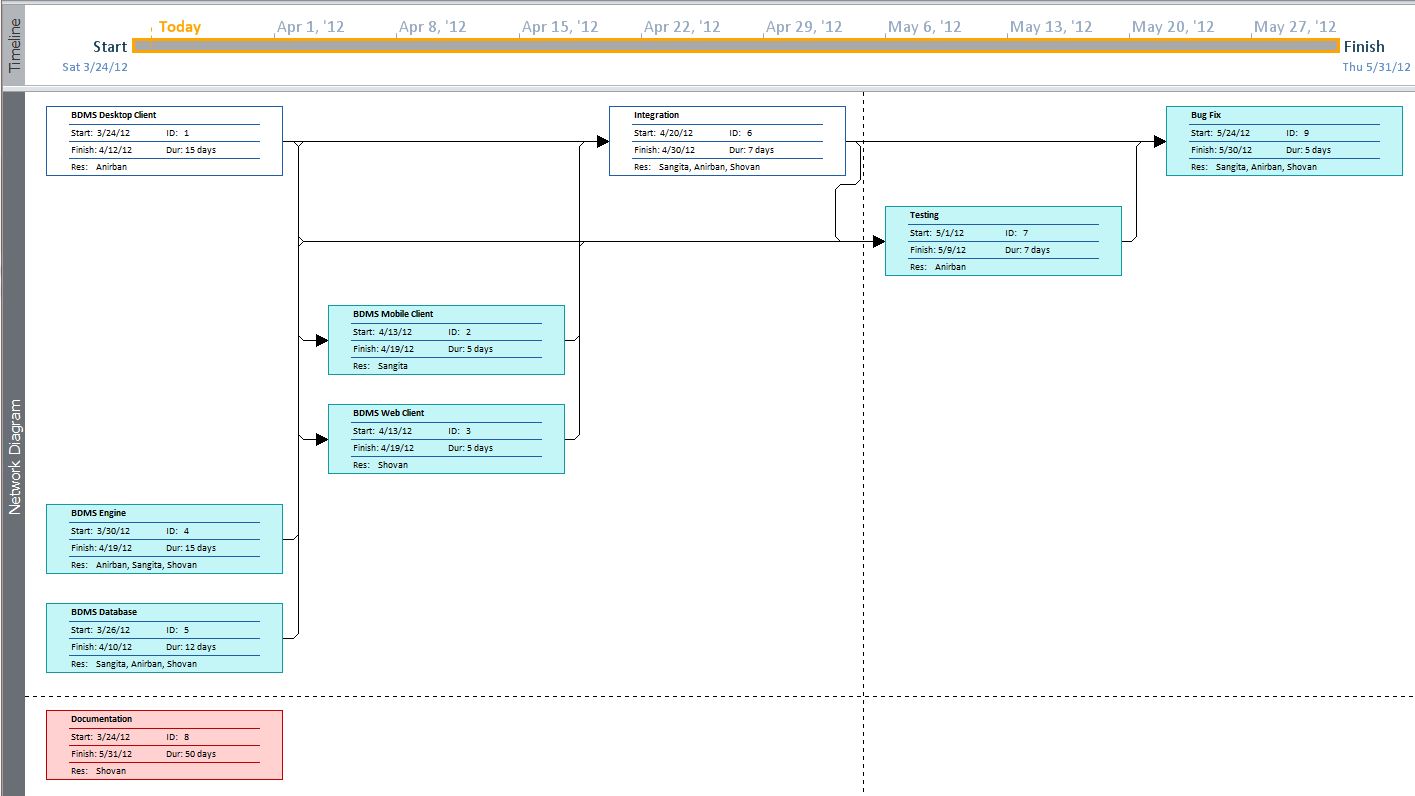
### Gantt chart

******

### Tracking Gantt

******

### Pert chart (Network Diagram)



## Hardware and Software Requirements

### Hardware requirement

* **Disc capacity :** 10 MB of available hard disk space
* **RAM :** 1 GB (32 Bit) or 2 GB (64 Bit)
* **Processor :** 1.6GHz or faster
* DVD-ROM Drive / USB **Port**

### Software Requirements

* Windows XP (x86) with Service Pack 3 / Windows Vista (x86 & x64) withService Pack 2 / Windows 7 (x86 & x64)
* Microsoft .NET 4.0

## Preliminary product description

The main components of Blood Donation Management system are:

BDMS Server :

This is the backbone of BDMS. This dedicated server stores all the data and handles the business logic of the software. It also handles interaction with clients and data syncing. This is divided into two major blocks.

BDMS Engine:

This is the controller of BDMS. This takes decision based on the business logic and employs other components.

BDMS Database:

This is the centralized storage of all BDMS data.

BDMS Desktop Client :

This is the full functionality client of BDMS which will be deployed on desktop computers and laptops.

## Conceptual Models

### E-R Diagram

We will design a RDBMS for Blood Donor Management System. The entities and their attributes are listed below. Attributes in Bold letter is the unique key.

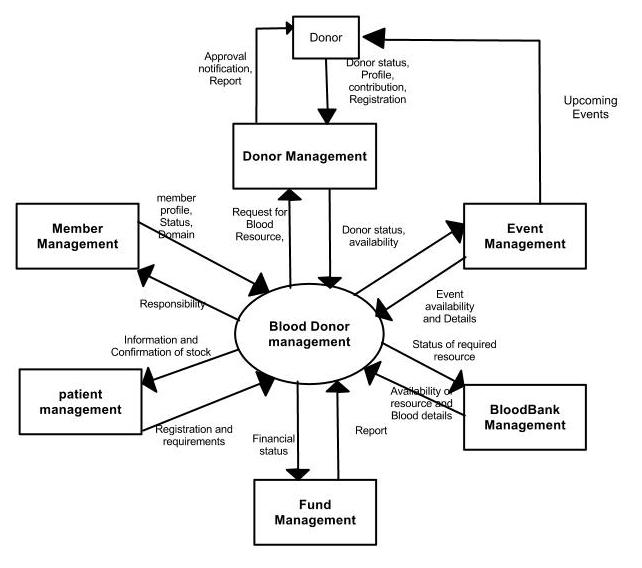
|  |  |
| --- | --- |
| **Entities** | **Attributes** |
| *Donor* | ***donor\_Id****, donor\_name, donor\_blood\_group, donor\_dob, donor\_address, donor\_contact, donor\_last\_blood\_donate* |
| *Patient* | ***patient\_id****, patient\_name, patient\_blood\_group, patient\_age, patient\_contact, patient\_address, patient\_admited\_address, patient\_expected\_date* |
| *Member* | ***member\_id,*** *member\_name, member\_doj, member\_address, member\_contect* |
| *Event* | ***event\_id,*** *event\_title, event\_doe, event\_venue, event\_goal* |
| *Hospital/Doctor/Blood Bank* | ***hdbb\_id,*** *hdbb\_type, hdbb\_name, hdbb\_address, hdbb\_contact* |
| *Well Wisher* | ***well\_wisher\_id,*** *well\_wisher\_name, well\_wisher\_address, well\_wisher\_doj, well\_wisher\_contact,* |
| *Fund* | ***fund\_id,*** *fund\_wellwisher\_name, fund\_contact, fund\_dod, fund\_received\_by, fund\_amount* |
| Expense | **expense\_id,** expense\_purpose, expense\_doe, expensed\_by, expensed\_amount |
| To Do | **todo\_id,** todo\_date, todo\_details |

***Relationship between Entities:***

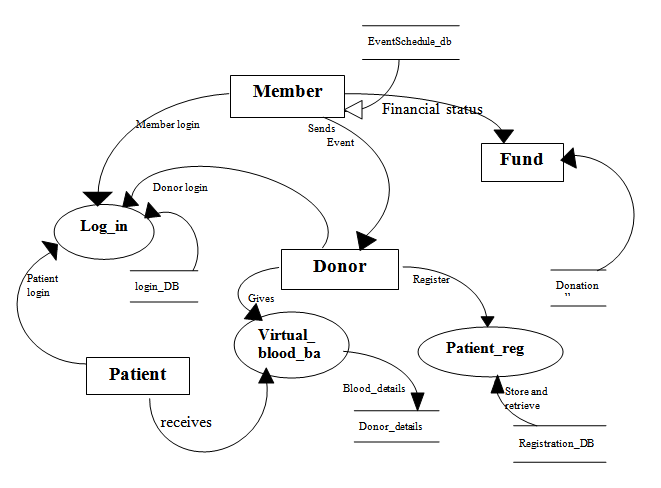
* Blood Donor Management System has donors  1 : N
* Blood Donor Management System has Machine 1 : N
* Users System uses Session 1 : 1
* Blood Donor Management System avails donation  1 : N
* Users provide Preferences  M : N
* Donor donates blood to patient→1:1
* Admin organizes events→1:1



### Context Diagram



### Data Flow Diagram



# SYSTEM DESIGN

## Basic Modules



*The main components of Blood Donation Management system are:*

* **BDMS Server**

This is the backbone of BDMS. This dedicated server stores all the data and handles the business logic of the software. It also handles interaction with clients and data syncing. This is divided into two major blocks.

* + **BDMS Engine**: This is the controller of BDMS. This takes decision based on the business logic and employs other components.
  + **BDMS Database**: This is the centralized storage of all BDMS data.
* **BDMS Desktop Client**

This is the full functionality client of BDMS which is deployed for desktop computers and laptops.

*DONOR:*

*All donor functionalities can be done here. User can add donor with details. Can view the donor details, which is already exists in database. Editing of donor details is also available here.*

*PATIENT:*

*Patient register for blood, view registered patient and if necessary modification of patient details can be done here. After giving successful service a patient can be removed from the data base.*

*MEMBER:*

*All member of the organization can be managed here. Here user can add, view and delete the members. Modification of existing member is also possible.*

*Well-WISHER:*

*It is necessary to maintain well-wishers details for a voluntary organization .here the application can keep track of the well-wishers. Addition, viewing, deletion and modification of well-wishers are an easy task.*

*FUNDS:*

*Here the user can maintain fund information’s details. Whenever a well-wisher wishes to donate money for fund it is easy for the user to accept and update the fund into the database. Here the users don’t need to put the well wishers name manually. A dropdown menu of well-wishers name will appear .From where users just need to select the well wishers name and update other necessary in formations.*

*Figure out all the expenses are also done in fund. Addition, viewing, deletion and modification of fund are done here.*

*FUNDS:*

*User can save, edit and delete the job should be done in upcoming days.*

*EVENTS:*

*Keep accounts of events that will be organized by the organization is very much important. Within event user can maintain the history as well as the upcoming events. Modification of events is also possible here.*

*DONOR ASSIGNMENT:*

*It is mandatory to assign which donor will donate blood for a particular patient .Here, within assign donor, the job is done. Whenever user wish to assign a donor, he just need to select the patient and all the patient information will be generated and user just need to select a donor name from a drop down list.*

*HELP:*

*Within help there some important information like address of a hospital or blood bank or doctors can be stored. Modifications of existing information are also possible.*

## Data Design

### Schema Design

|  |  |  |
| --- | --- | --- |
| ***Donor*** |  |  |
| ***Column Name*** | ***Data type*** | ***Default*** |
| *donor\_id* | *VARCHAR(45)* | *NOT NULL* |
| *donor\_name* | *VARCHAR(30)* | *NOT NULL* |
| *donor\_blood\_group* | *VARCHAR(6)* | *NULL* |
| *donor\_dob* | *DATE* | *NULL* |
| *donor\_address* | *VARCHAR(100)* | *NULL* |
| *donor\_contact* | *INT(12)* | *NULL* |
| *donor\_last\_blood\_donate* | *DATE* | *NULL* |
|  |  |  |
| ***Event*** |  |  |
| ***Column Name*** | ***Data type*** | ***Default*** |
| *event\_id* | *VARCHAR(45)* | *NOT NULL* |
| *event\_title* | *VARCHAR(45)* | *NOT NULL* |
| *event\_doe* | *DATE* | *NULL* |
| *event\_venue* | *VARCHAR(100)* | *NULL* |
| *event\_goal* | *VARCHAR(300)* | *NULL* |
|  |  |  |
| ***Member*** |  |  |
| ***Column Name*** | ***Data type*** | ***Default*** |
| *member\_id* | *VARCHAR(45)* | *NOT NULL* |
| *member\_name* | *VARCHAR(45)* | *NULL* |
| *member\_doj* | *DATE* | *NULL* |
| *member\_doj* | *VARCHAR(100)* | *NULL* |
| *member\_contect* | *VARCHAR(12)* | *NULL* |
|  |  |  |
| ***Patient*** |  |  |
| ***Column Name*** | ***Data type*** | ***Default*** |
| *patient\_id* | *VARCHAR(45)* | *NOT NULL* |
| *patient\_name* | *VARCHAR(30)* | *NOT NULL* |
| *patient\_blood\_group* | *VARCHAR(6)* | *NULL* |
| *patient\_age* | *INT(3)* | *NULL* |
| *patient\_contact* | *VARCHAR(45)* | *NULL* |
| *patient\_address* | *VARCHAR(100)* | *NULL* |
| *patient\_admited\_address* | *VARCHAR(100)* | *NULL* |
| *patient\_expected\_date* | *DATE* | *NULL* |
| *assigned\_donor* | *VARCHAR(45)* | *TBA'* |
| *assigned\_donor\_contact* | *VARCHAR(45)* | *TBA'* |
|  |  |  |
| ***Expense*** |  |  |
| ***Column Name*** | ***Data type*** | ***Default*** |
| *expense\_id* | *VARCHAR(45)* | *NOT NULL* |
| *expense\_purpose* | *VARCHAR(45)* | *NULL* |
| *expense\_doe* | *DATETIME* | *NULL* |
| *expensed\_by* | *VARCHAR(45)* | *NULL* |
| *expensed\_amount* | *DOUBLE* | *NULL* |
|  |  |  |
| ***Todo*** |  |  |
| ***Column Name*** | ***Data type*** | ***Default*** |
| *todo\_id* | *VARCHAR(45)* | *NOT NULL* |
| *todo\_date* | *DATETIME* | *NOT NULL* |
| *todo\_details* | *LONGTEXT* | *NOT NULL* |
|  |  |  |
| ***Blood\_banks*** |  |  |
| ***Column Name*** | ***Data type*** | ***Default*** |
| *login\_id* | *VARCHAR(45)* | *NOT NULL* |
| *login\_name* | *VARCHAR(30)* | *NULL* |
| *login\_type* | *VARCHAR(10)* | *NULL* |
| *login\_password* | *VARCHAR(20)* | *NULL* |
| *login\_address* | *VARCHAR(100)* | *NULL* |
| *login\_dob* | *DATE* | *NULL* |
|  |  |  |
| ***Fund*** |  |  |
| ***Column Name*** | ***Data type*** | ***Default*** |
| *fund\_Id* | *VARCHAR(100)* | *NOT NULL* |
| *fund\_wellwisher\_name* | *VARCHAR(45)* | *NULL* |
| *fund\_contact* | *VARCHAR(45)* | *NULL* |
| *fund\_dod* | *DATETIME* | *NULL* |
| *fund\_received\_by* | *VARCHAR(45)* | *NULL* |
| *fund\_amount* | *DOUBLE* | *NULL* |
|  |  |  |
| ***Hdbb*** |  |  |
| ***Column Name*** | ***Data type*** | ***Default*** |
| *hdbb\_id* | *VARCHAR(45)* | *NOT NULL* |
| *hdbb\_type* | *VARCHAR(15)* | *NULL* |
| *hdbb\_name* | *VARCHAR(45)* | *NULL* |
| *hdbb\_address* | *VARCHAR(100)* | *NULL* |
| *hdbb\_contact* | *VARCHAR(12)* | *NULL* |
|  |  |  |
| ***Well\_wisher*** |  |  |
| ***Column Name*** | ***Data type*** | ***Default*** |
| *well\_wisher\_id* | *VARCHAR(45)* | *NOT NULL* |
| *well\_wisher\_name* | *VARCHAR(30)* | *NOT NULL* |
| *well\_wisher\_address* | *VARCHAR(100)* | *NULL* |
| *well\_wisher\_doj* | *DATE* | *NULL* |
| *well\_wisher\_contact* | *VARCHAR(12)* | *NULL* |
| *well\_wisher\_remarks* | *VARCHAR(20)* | *NULL* |

## Procedural Design

### Data Structure

*DonorInfo.cs*

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Text;*

*namespace BDMSData*

*{*

*publicclassDonorInfo*

*{*

*publicstring id { get; set; }*

*publicstring name { get; set; }*

*publicstring bloodGroup { get; set; }*

*publicstring phone { get; set; }*

*publicstring address { get; set; }*

*publicDateTime lastDonateDate { get; set; }*

*publicDateTime dob { get; set; }*

*}*

*}*

*PatientInfo.cs*

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Text;*

*namespace BDMSData*

*{*

*publicclassPatientInfo*

*{*

*publicstring id { get; set; }*

*publicstring name { get; set; }*

*publicstring bloodGroup { get; set; }*

*publicint age { get; set; }*

*publicstring address { get; set; }*

*publicstring phone { get; set; }*

*publicstring admittedAddress { get; set; }*

*publicDateTime expectedDate { get; set; }*

*publicstring assignedDonor { get; set; }*

*publicstring donorContact { get; set; }*

*}*

*MemberInfo.cs*

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Text;*

*namespace BDMSData*

*{*

*publicclassMemberInfo*

*{*

*publicstring id { get; set; }*

*publicstring name { get; set; }*

*publicDateTime doj { get; set; }*

*publicstring address { get; set; }*

*publicstring phone { get; set; }*

*}*

*}*

*WellWisherInfo.cs*

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Text;*

*namespace BDMSData*

*{*

*publicclassWellWisherInfo*

*{*

*publicstring id { get; set; }*

*publicstring name { get; set; }*

*publicstring address { get; set; }*

*publicDateTime doj { get; set; }*

*publicstring phone { get; set; }*

*publicstring remarks { get; set; }*

*}*

*}*

*EventInfo.cs*

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Text;*

*namespace BDMSData*

*{*

*publicclassEventInfo*

*{*

*publicstring id { get; set; }*

*publicstring eventTitle { get; set; }*

*publicDateTime eventDoe { get; set; }*

*publicstring eventVenue { get; set; }*

*publicstring eventGoal { get; set; }*

*}*

*}*

*ExpenseInfo.cs*

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Text;*

*namespace BDMSData*

*{*

*publicclassExpenseInfo*

*{*

*publicstring id { get; set; }*

*publicstring purpose { get; set; }*

*publicDateTime doe { get; set; }*

*publicstring expensed\_by { get; set; }*

*publicdouble amount { get; set; }*

*}*

*}*

*FundInfo.cs*

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Text;*

*namespace BDMSData*

*{*

*publicclassFundInfo*

*{*

*publicstring id { get; set; }*

*publicstring wellwisher\_name { get; set; }*

*publicstring contact { get; set; }*

*publicDateTime dod { get; set; }*

*publicstring received\_by { get; set; }*

*publicdouble amount { get; set; }*

*}*

*}*

*TodoInfo.cs*

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Text;*

*namespace BDMSData*

*{*

*publicclassTodoInfo*

*{*

*publicstring id { get; set; }*

*publicDateTime date { get; set; }*

*publicstring details { get; set; }*

*}*

*}*

*HdbbInfo.cs*

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Text;*

*namespace BDMSData*

*{*

*publicclassHdbbInfo*

*{*

*publicstring id { get; set; }*

*publicstring type { get; set; }*

*publicstring name { get; set; }*

*publicstring address { get; set; }*

*publicstring phone { get; set; }*

*}*

*}*

### Algorithm Design

*Algorithm for searching:*

*Following algorithm is used to search and view a customer details by its id-*

*Input taken by GUI: A user provides the id of a specific customer inside a textbox. The text of that textbox is then taken inside the data module. The data module carries the data to the Controller module.*

*Searching is done inside the database: The controller module sends the data to the Storage module through the data module and tells it to search any available answer or answers inside the specific table of the database For example, in this case, the query would be ,’select \* from customer where id = @id’.*

*Output is shown at the GUI: The storage module does the desired searching and sends the output to the controller module through the data module and the output is displayed to the user through a list view.*

## User interface Design

## Security Issue

*BDMS provides password based security for its user. In terms of security unauthorized access will be denied and register user will be able to change as necessary. User must impute a valid* ***user name*** *and* ***password*** *to login to the application. Without a valid* ***user name*** *and* ***password*** *no one can access to the application. This way it prevents misuse of application and database.*

*For data security there will be a digital backup and restore system. It is possible to take backup after some period of time.*

## Test Case Design

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Test Case Id* | *Item* | *Description* | *Actual Result* | *Comment* | *Status* |
| *BDMS-001* | *Login* | *Enter Type, user Id and Password for log in.* | *Successfully Logged in.* | *NA* | *PASS* |
| *BDMS-002* | *Cancel* | *Select Cancel to close login window* | *Successfully Canceled.* | *NA* | *PASS* |
| *BDMS-003* | *Create An Account* | *Create a new account to login as a new user. Enter Name, User ID, Type, Password, Date of Birth, and Address to create new account.* | *New account successfully created.* | *NA* | *PASS* |
| *BDMS-004* |  | *Show the donors details.* | *Successfully displayed.* | *NA* | *PASS* |
| *BDMS-005* | *Add Donor* | *To add a new Donor enter the Donor Name, Blood group, Date of Birth, Address, Contact, Last Donation Date etc..* | *New Donor account successfully created.* | *NA* | *PASS* |
| *BDMS-006* | *Edit Donor* | *Select the Donor and click the Edit option. Now edit the Donor Details and submit the details.* | *Donor account successfully updated.* | *NA* | *PASS* |
| *BDMS-007* | *Delete Donor* | *Select the Donor and click the Delete option.* | *Donor account successfully deleted.* | *NA* | *PASS* |
| *BDMS-008* | *OK* | *Select OK to close Donors window* | *Successfully Closed.* | *NA* | *PASS* |
| *BDMS-009* |  | *Show the patients details.* | *Successfully displayed.* |  | *PASS* |
| *BDMS-010* | *Search For Blood* | *Enter Blood Group, Expected Date and Location for searching blood Donor.* | *Successfully Display Search result.* | *NA* | *PASS* |
| *BDMS-011* | *Register foe Blood* | *Enter Patient's name, Address, Blood group, Age, Expected Date, Contact etc. for patients requirement registration.* | *Successfully registered.* | *NA* | *PASS* |
| *BDMS-012* | *OK* | *Select OK to close Patients window* | *Successfully Closed.* | *NA* | *PASS* |
| *BDMS-013* |  | *Show the available donor details.* | *Successfully displayed.* | *NA* | *PASS* |
| *BDMS-014* | *Print* | *Print the Available donor details.* | *Successfully printed.* | *NA* | *PASS* |
| *BDMS-015* | *OK* | *Select OK to close Virtual Blood Bank window* | *Successfully Closed.* | *NA* | *PASS* |
| *BDMS-016* |  | *Show the members details.* | *Successfully displayed.* | *NA* | *PASS* |
| *BDMS-017* | *Add Member* | *To add a new Member enter the Member Name, Date of Joining, Address, Contact etc..* | *New Member account successfully created.* | *NA* | *PASS* |
| *BDMS-018* | *Edit Member* | *Select the Member and click the Edit option. Now edit the Member Details and submit the details.* | *Member account successfully updated.* | *NA* | *PASS* |
| *BDMS-019* | *Delete Donor* | *Select the Member and click the Delete option.* | *Member account successfully deleted.* | *NA* | *PASS* |
| *BDMS-020* | *OK* | *Select OK to close Members window* | *Successfully Closed.* | *NA* | *PASS* |
| *BDMS-021* |  | *Show the Available Funds details.* | *Successfully displayed.* | *NA* | *PASS* |
| *BDMS-022* | *Report* | *Show the Details about Funds.* | *Successfully displayed.* | *NA* | *PASS* |
| *BDMS-023* | *Add Fund (Nested)* | *Enter Well Wisher Name, Contact No., Date of Donation, Received By, Amount etc. to add Received fund Details.* | *Successfully new fund details created.* | *NA* | *PASS* |
| *BDMS-024* | *Add Expense (Nested)* | *Enter Expense purpose, Date of Expensed, Expensed By, Amount etc. to add Expensed fund Details.* | *Successfully new expended fund details created.* | *NA* | *PASS* |
| *BDMS-025* | *Well Wishers* | *Show the Details about Funds.* | *Successfully displayed.* | *NA* | *PASS* |
| *BDMS-026* | *Add Well Wisher (Nested)* | *To add a new Well Wisher enter the Well Wisher's Name, Date of Joining, Address, Contact No. etc..* | *New Well Wisher account successfully created.* | *NA* | *PASS* |
| *BDMS-027* | *Edit Well Wisher (Nested)* | *Select the Well Wisher and click the Edit option. Now edit the Well Wisher Details and submit the details.* | *Well Wisher's account successfully updated.* | *NA* | *PASS* |
| *BDMS-028* | *Delete Well Wisher (Nested)* | *Select the Well Wisher and click the Delete option.* | *Well Wisher's account successfully deleted.* | *NA* | *PASS* |
| *BDMS-029* | *OK* | *Select OK to close Well Wishers window* | *Successfully Closed.* | *NA* | *PASS* |
| *BDMS-030* |  | *Show the Things To Do details.* | *Successfully displayed.* | *NA* | *PASS* |
| *BDMS-031* | *Add To Do* | *Write in the text field and click on the Add To Do button to add details about to do.* | *Successfully To Do added.* | *NA* | *PASS* |
| *BDMS-032* | *Edit Details* | *Select the To Do Details and click the Edit option. Now edit the To Do Details and submit the details.* | *To Do details successfully updated.* | *NA* | *PASS* |
| *BDMS-033* | *Delete Details* | *Select the To Do Details and click the Delete option.* | *To Do Details successfully deleted.* | *NA* | *PASS* |
| *BDMS-034* | *OK* | *Select OK to close To Do Details window.* | *Successfully Closed.* | *NA* | *PASS* |
| *BDMS-035* |  | *Show the Upcoming event details.* | *Successfully displayed.* | *NA* | *PASS* |
| *BDMS-036* | *Add Upcoming Event* | *Enter Event Title, Event Date, Goal of Event, Venue etc. to add new Events details.* | *Successfully Upcoming Event added.* | *NA* | *PASS* |
| *BDMS-037* | *Previous Events* | *Show the celebrated events.* | *Successfully displayed.* | *NA* | *PASS* |
| *BDMS-038* | *Edit Event* | *Select the Event Details and click the Edit option. Now edit the Event Details and submit the details.* | *Event details successfully updated.* | *NA* | *PASS* |
| *BDMS-039* | *Delete Event* | *Select the Event Details and click the Delete option.* | *Event Details successfully deleted.* | *NA* | *PASS* |
| *BDMS-040* | *OK* | *Select OK to close Event window* | *Successfully Closed.* | *NA* | *PASS* |
| *BDMS-041* |  | *Open Calculator to calculate Age, Funds etc..* | *Successfully opened.* | *NA* | *PASS* |
| *BDMS-042* |  | *Show the details about Organization.* | *Successfully displayed.* | *NA* | *PASS* |
| *BDMS-043* | *Hospital/Doctor/Blood Bank* | *Show the details about Hospital, Doctor, Blood Bank.* | *Successfully displayed.* | *NA* | *PASS* |
| *BDMS-044* | *Add Hospital/Doctor/Blood Bank* | *Enter Type, Address and Contact No. to add new Hospital or Doctor or Blood Bank details.* | *Successfully Hospital/Doctor/Blood Bank added.* | *NA* | *PASS* |
| *BDMS-045* | *Edit Details (Nested)* | *Select the Hospital or Doctor or Blood Bank Details and click the Edit option. Now edit the Hospital or Doctor or Blood Bank Details and submit the details.* | *Hospital or Doctor or Blood Bank details successfully updated.* | *NA* | *PASS* |
| *BDMS-046* | *Delete Details (Nested)* | *Select Hospital or Doctor or Blood Bank Details and click the Delete option.* | *Hospital or Doctor or Blood Bank Details successfully deleted.* | *NA* | *PASS* |
| *BDMS-047* | *Change Password* | *To change password enter current password, new password and re enter new password and click submit button,* | *Password successfully changed.* | *NA* | *PASS* |
| *BDMS-048* | *OK* | *Select OK to close Help Details window.* | *Successfully Closed.* | *NA* | *PASS* |
| *BDMS-049* |  | *Select close option to close the application.* | *Application successfully closed.* | *NA* | *PASS* |

# IMPLIMENTATION AND TESTING

## Implementation Approaches

*We started the project aiming to develop it such a way that after completing of the project we can easily study the codes for understanding how it works. We used one of the best features of object oriented programming, the modular approach perfectly and made all the codes well separated from each other according to its purpose. For example, we have kept all the database interaction related codes in a single CS page and we have modularized it even further inside the page. We have separated all the GUI tools style definitions in a single resource files. We have separated the data classes in a different cs page. So we can not only study the codes or find errors in the codes easily, we can also use them in other projects in no times which saves lots of time and effort.*

### Code Efficiency

*We started working on the project keeping in mind that we must develop it in a way that it not only provides a very easy to use GUI but also provide a fast and flexible service to the users. We know that a particular work can be done in more than one ways. We have tried all the options and then chose the one which provides the fastest and most secure performance. First of all, we have used the latest technologies of Microsoft like visual studio 2010 as IDE and WPF as GUI to keep our application’s performance few steps ahead. We have studiesd all the rules of software development life cycle and applied them to keep our application felexible. We have given special attention to the storage related codes. We have avoided all the unnecessary database codes and kept them as short as possible without harming our purpose so that insertion, updation, deletion and fetching of data take place flexibly. You can see the result as a user; our application does all the works very smoothly.*