

**High Level Design & Low Level Design**

The purpose of this document is to provide with a template for documenting both HLD & LLD.

**DATABASE SERVER**

**CONTENTS**

**SL.No Title Pg.no**

1. Introduction 7

1.1 Intended Audience 7

1.2 Intended Use 7

1.3 Project Purpose 7

1.4 Project Scope 7

1.5 Functional Overview 7

1.6 Design and Implementation Constraints 8

1.7 Assumptions and Dependencies 8

1.8 Risk Management 8

2 Design Overview 8

2.1 Design Objectives 8

2.2 Recommended Architecture 9

2.2.1 UML diagrams 10

2.3 Data flow diagram 11

2.4 Flow chart 13

2.5 Product perspective 13

2.6 Product features 14

2.7 User classes and characteristics 14

2.8 Operating environment 14

3 System features 14

3.1 Functional requirements 14

3.2 Implicit requirements 15

3.3 Software interfaces 16

3.4 Communication Interfaces 16

3.5 Technical requirements 16

3.6 Nonfunctional Requirements 16

4 Detailed system design 17

4.1 Key facilities 17

4.2 Detailed level database design 17

4.3 Archival and retention requirements 17

4.4 Disaster and failure requirements 17

4.5 Variables 17

4.6 Business Logic 17

5 Language support 17

5.1 Deployment considerations 19

5.2 Integration requirements 19

5.3 Network 19

5.4 Operating system 19

6 References 19

7 Appendix 19

**Introduction**

A database server is amachine running database software dedicated to providing database services. It is a crucial component in the client-server computing environment where it provides business-critical information requested by the client systems. A database server consists of hardware and software that run a database.

**1.1 Intended Audience**

This application is intended to be read by, Capgemini employees.

**1.2 Intended Use**

- Organization Team

-Clients

Since this a Organization-purpose Software others cannot access it out of organization.

**1.3 Project Purpose**

Database servers offer companies a simple way to update, maintain and save large amounts of data. If you want to incorporate a database server into your company's operations, it's important to understand the various components of database servers and the types of database servers available. In this article, we discuss what a database server is, explain how it works and provide eight examples of database servers.

**1.4 Project Scope**

A database scoped credential is a record that contains theauthentication information that is required to connect to a resource outside Server. Before creating a database scoped credential, the database must have a master key to protect the credential.

**1.5 Functional Overview**

A Database Server is a computer that is dedicated todatabase storage and retrieval. The database server holds the databases. Upon requests from the client machines, it searches the database for selected records and passes them back over the network. It is a service that is added to our databases which make our daily tasks easier. It eliminates tedious and time-consuming administration tasks and makes our tasks simpler and more flexible. Now most of the organizations are going for DBaaS as it helps organizations to accelerate their business performance by starting their working with database more easily and running the workloads without delay.

**1.6 Design and Implementation Constraints**

Database Design is a collection of processes that facilitate the designing, development, implementation, and maintenance of enterprise data management systems. The logical model concentrates on the data requirements and the data to be stored independent of physical considerations.

**1.7 Assumptions and Dependencies**

The database system requires massive amounts of peripheral storage space. This is needed not only for the database of interest to users, but also for the database system as well.

**1.8 Risk Management**

Threats to a database can come from any direction and in any form, whether Human error, Hardware failures or even simple misunderstandings. Risk mitigation needs to cover all threats, including information theft, by either preventing it or having a reaction plan in place if it occurs thereby restricting the loss. Living in the Information Age means the very survival of a business is dependent on the information they have and the availability of that information.

RDB will assess your database environment and implement procedures and practices that will mitigate virtually any database risk. RDB makes use of its own in-house developed procedures as well as features built into the database software such as Flashback Technology, Log Mining, RAC, Logical and Physical DataGuard, Auditing as well as foolproof backup and recovery strategies.

**2.Design Overview**

Database design is a collection of steps that help create, implement, and maintain a business’s data management systems. The primary purpose of designing a database is to produce physical and logical models of designs for the proposed database system.

Specific rules govern a good database design process. The first rule in creating a database design is to avoid data redundancy. It wastes space and increases the probability of faults and discrepancies within the database. The second rule is that the accuracy and comprehensiveness of information are imperative. A database containing erroneous information will lead to inaccurate analysis and reporting. Consequently, it can mislead decision-makers and adversely affect a company’s performance. Therefore, it’s essential to keep things rules in mind when designing the database for your organization.

**2.1 Design objectives**

There are many distinct objectives that you *must* achieve to design a good, sound, structured database. You can avoid many of the problems you may encounter by keeping the following objectives in mind and constantly focusing on these whilst designing your database.

*The database supports both required and ad hoc (unplanned) information retrieval.* The database must be designed to store the data necessary to support information requirements defined during the design process and any possible ad hoc queries that may be posed by the users.

**2.2 Recommended Architecture**

UML design is the shortest form of “Unified Modeling Language”. The purpose of this modeling language is to visualize the design of the system. There are total of 14 types of UML diagram. They are:

- Use case diagram

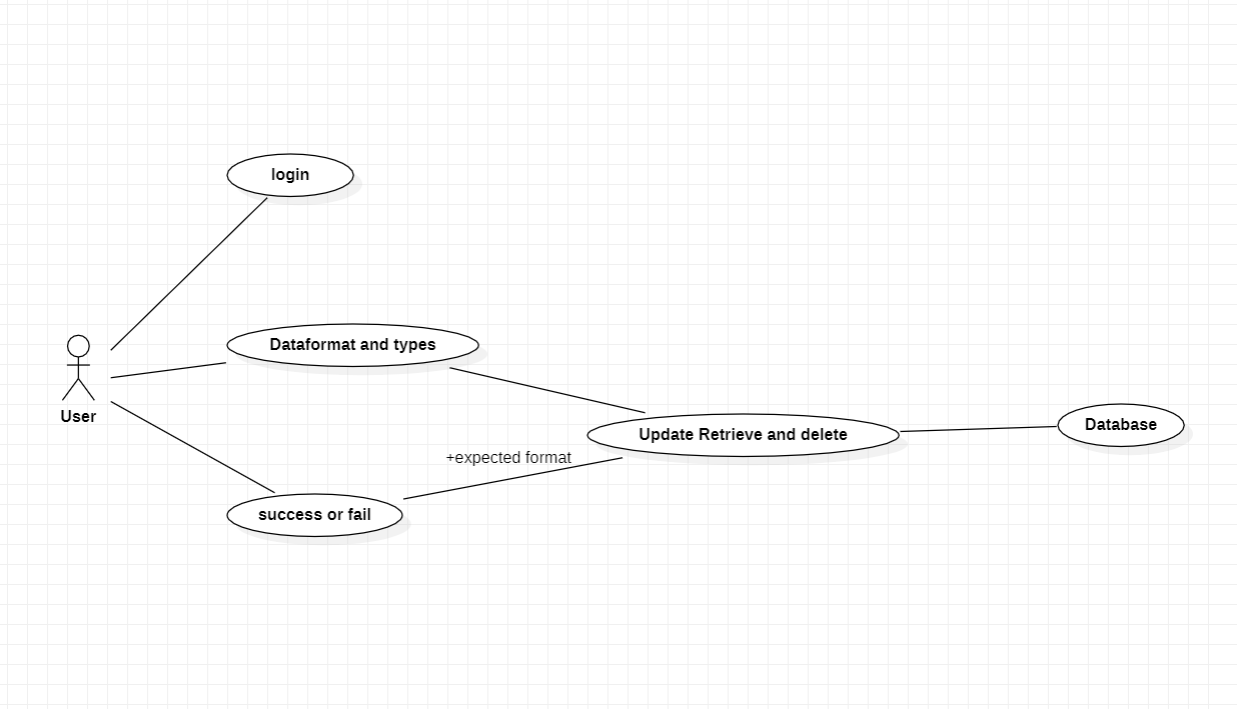
- Sequence diagram

- Class diagram

- dataflow diagram

Use case diagram

It is also called behavioral UML diagram. It gives a graphic over-view of the actors involved in a system directly. It shows how different funtions needed by the actors how they are interacted.

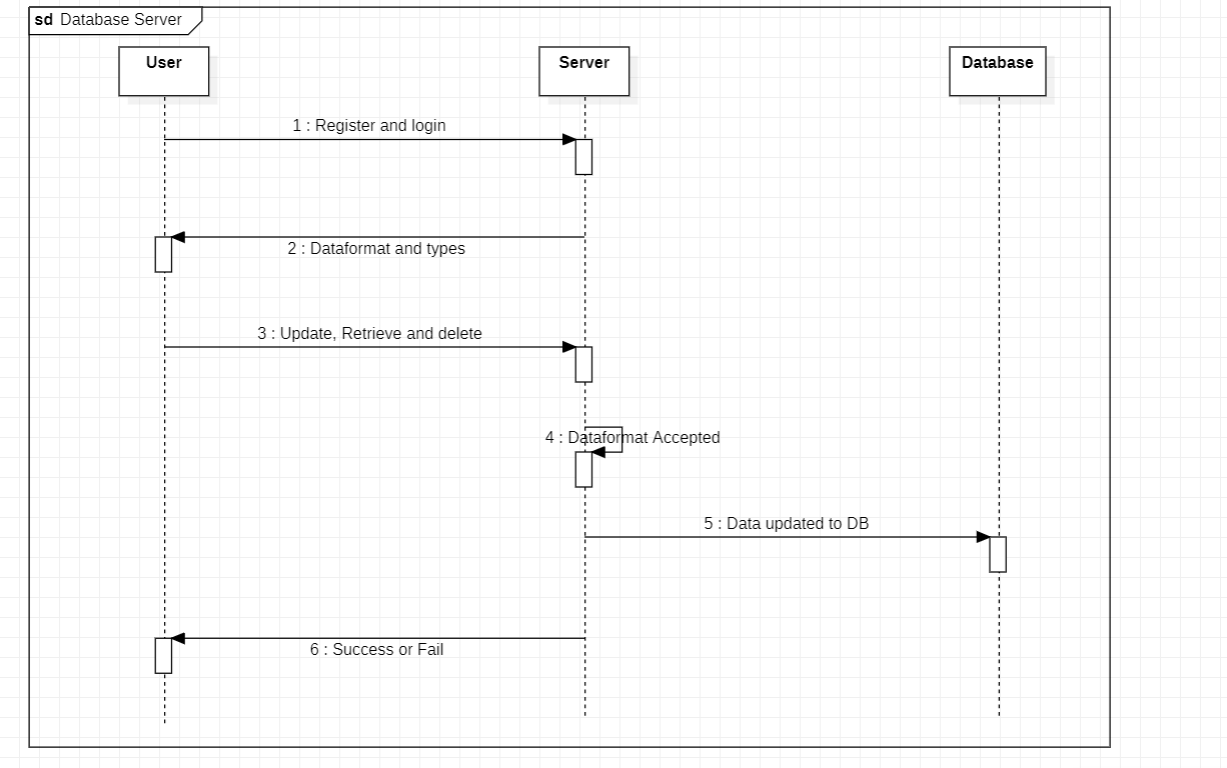
Below is the “USE CASE DIAGRAM”.

Sequence diagram

A sequence diagram is a Unified Modeling Language(UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the inetraction.

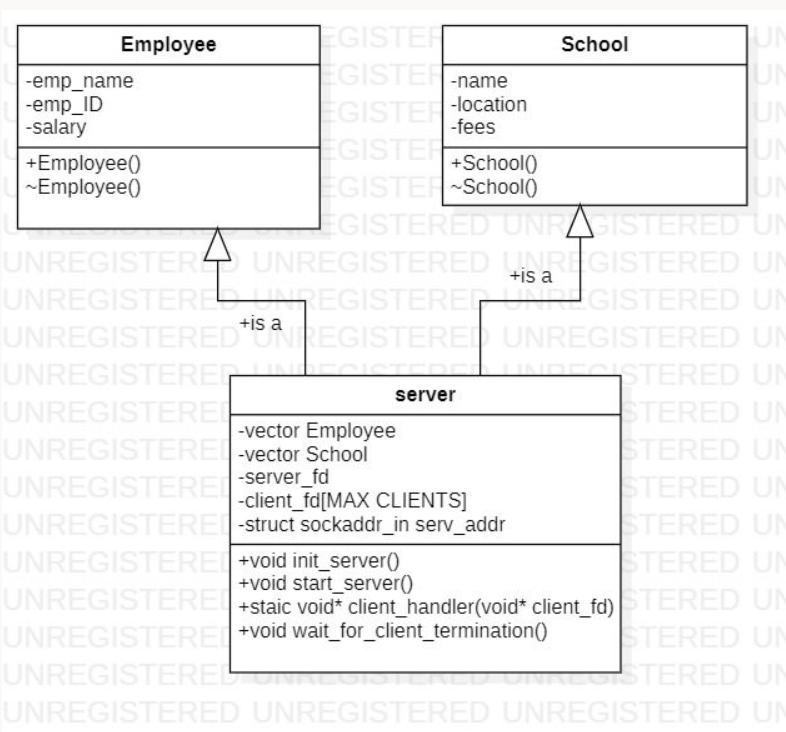
A sequence diagram shows the dequence of messages passed between objects. Sequence diagrams can also show the control structures between objects.

Below is the “SEQUENCE DIAGRAM”.



Class diagram

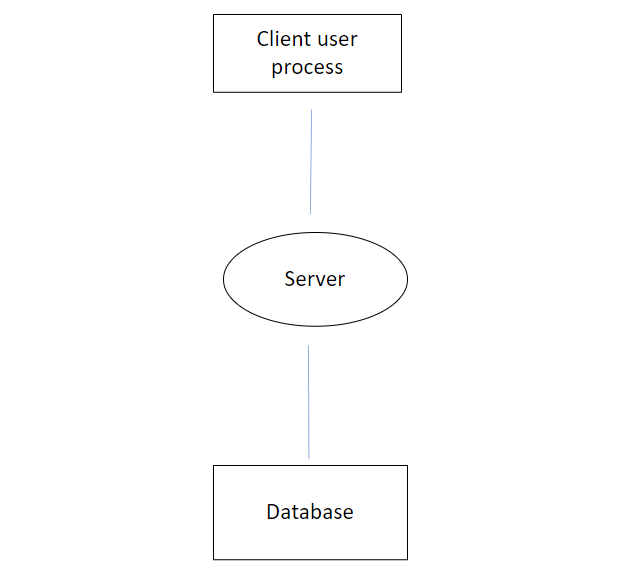
This is the most used UML diagram in the field of software engineering design. It is called as a main building block of any object-oriented solution. Usually, it illustrates the classes in a system, attributes and operations of each class and the relationship between each class.

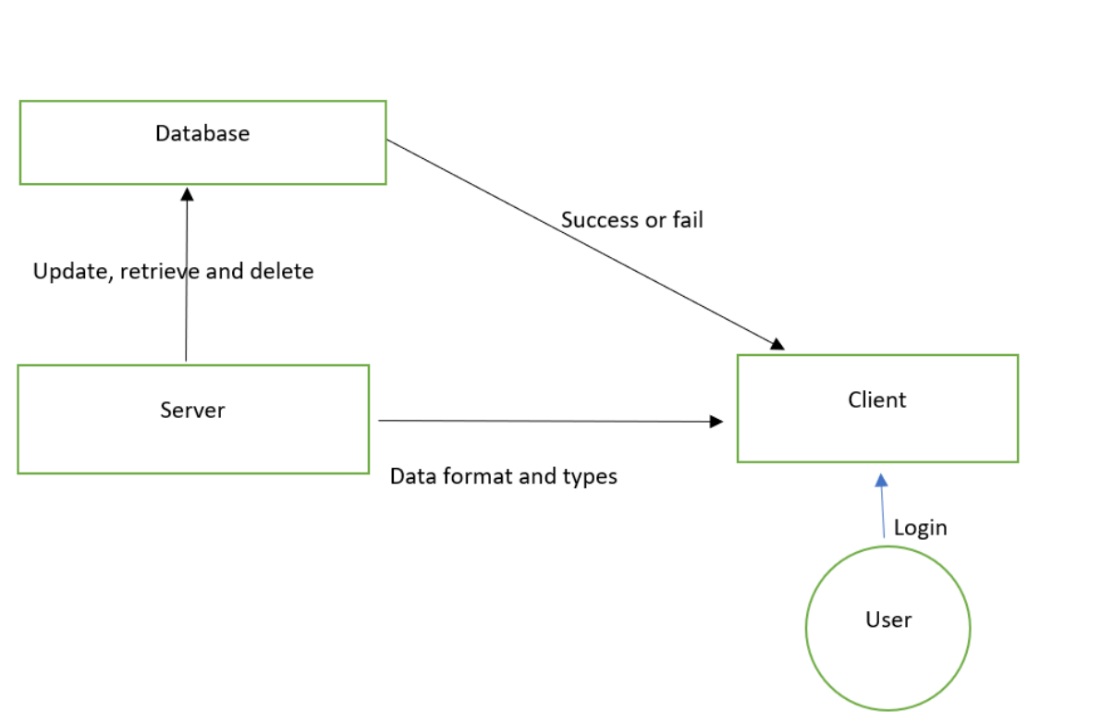


2.3 Data flow diagram

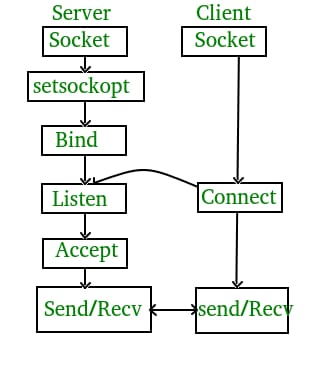
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. I can be manual, automated, or a combination of both. It shows how data enters and leaves the system, what changes the information, and where data is stored.

**DFD0 and DFD1:**





**Dataflow Diagram**



**2.4 Product Perspective**

A database server is amachine running database software dedicated to providing database services. It is a crucial component in the client-server computing environment where it provides business-critical information requested by the client systems. A database server consists of hardware and software that run a data.

**2.5 Product Features**

- Dataformats and types(Employee data)

- Server acceptance

- Updation, Retrieveing and Deletion

**2.6 User Classes and Characteristics**

- Client

- Server

- Database

- Client are key part of the systems. Clients have to register and login into the application.

- Server has access to the database. Once the client login into the application it will be displaying data formats and types. If the data formats and types match with expected, then the server will allow the user to modify and update the information into the database.

-Database will contains information which are storing

**2.7 Operating Environment**

-Language Requirements: Software must be only in English.

-Memory: System will have only 10GB space of data server.

-Implementation Constraints: Application should be based on Cpp.

-Reliability Requirements: System should sync frequently to backup server in order to avoid the data loss during failure, so it can be recovered.

**3. System Features**

3.1 Functional Requirements

3.1.1: The server should maintain different types and formats of data (ex types: school data, employee data etc..)

3.1.2: The server will receive the type and format of data from the client followed by actual data entry and the server will add to DB.

3.1.3: The server should store the data in an appropriate structure and return success/fail to client

3.1.4: The server should maintain some statistics related to different type so f data

3.1.5: The server will publish the different types and format for each type of data to clients during accepting the new client connection.

3.1.6: The client provides a user menu interface for adding data to DB, deleting from DB etc.

3.1.7: The client gets the different types and format for each type during connection establishment and stores this info locally

3.1.8: The client accepts input from user for these types only and as per the format for each type. In case of error in type or format will return error and not send this data to server.

3.1.9: The client should be able to get the statistics information from server about how many types are currently stored.

3.1.10: The client should be able to print the current values stored in DB based on given type

3.1.11: The client should not maintain the DB it should act as an interface only i.e. do not store any data except the types supported by server and format of each type

3.1.12: The client and server can run on different systems or on same system.

3.1.13: The server should accept min of 1 client and max of 2 clients

3.1.14: The server should employ proper protection of DB in case 2 clients Attempt to write/modify the DB values at same time.

3.1.15: Handling of errors for different scenarios (ex: server closed but client tries to connect, server is writing to DB for another client and at same time new client is requesting connection etc..)

**3.2 Implicit Requirements**

**3.2.1 Error message for invalid users**

**NA**

**3.2.1 Modification**

The server supports modifications of the users details.

**3.3 External Interface Requirements**

**3.3.1 User Interfaces**

Menu

**3.3.2 Hardware Interfaces**

- GHz processor, 2 GB RAM or more (system memory)

- 20 GB of hard-drive space or more

- VGA capable of 1024x768 screen resolution

- Necessary computer peripherals such as keyboards etc

- Internet Connectivity (Wired/Wireless)

**3.3 Software Interfaces**

- Window/Linux Based OS/Mac OS/ Any OS capable of running c++

- Databa

- Server

**3.4 Communication Interfaces**

- User can access the application.

**3.5 Technical Requirements**

Process synchronization

Shared Memory in Linux

Socket Programming in C - TCP)

Support for statistics

I/O Multiplexing (in Linux using C)

Logging and debugging framework

**3.5.1 Tools to be Used**

- P thread

- System Programming

- C pp Language

**3.6 Non-functional Requirements**

- CppUnit to automatic unit testing

- Val grind to detect memory task

- Make file

- Multi file multi directory solution with two step compilation process.

**3.6.1 System Features**

- Supportability: The system is easy to

- Design Constraints: The system is built using Cpp and system programming language.

- Usability: Users after login. Users can able to update, retrieve and delete from database.

- Reliability & Availability: The system is available 24/7 that is whenever the user would like to use the system they can use it up to its functionalities.

- Performance: The system will work on the user’s in organization.

**4. DETAILED SYSTEM DESIGN**

**4.1 Key Entities**

Users, Updation, Retrieve and Deleting of data from Database.

**4.2 Detailed-Level Database Design**

Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space. Therefore, there has to be a brilliant concept of designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

**4.2.1 Data Mapping Information**

Data mapping is the process ofmatching fields from one database to another. It's the first step to facilitate data migration, data integration, and other data management tasks. Before data can be analyzed for business insights, it must be homogenized in a way that makes it accessible to decision makers.

**4.3 Archival and Retention Requirements**

Archive and purge could be resource (like CPU) intensive. If archiving is required, make sure there is enough disk space available on the database server since the data would be moved to archive tables instead of simply purging. Archival space should beequal to or greater than the current table's storage.

**4.4 Disaster and Failure Requirements**

The Reasons for failure are

* 1. File Corruption
* 2. File System Damage
* 3. Software and Hardware Failure

**4.5 Variables**

**4.5.1** class SocketClient

{

public:

SocketClient(std::string ip="127.0.0.1",int port=5000):port(port),address(ip){}

~SocketClient(){};

void getinfo(){cout<<this->address<<" "<<this->port<<endl;}

int CreateSocket();

int Bind();

int Connect();

int DisplayMenu();

int SendAndClose();

int HandleEdit();

private:

int port;

int socket\_fd;

char ServerMsg[SERVER\_MSG\_SIZE];

data Data;

struct sockaddr\_in ServerInfo;

string address;

};

**4.5.2** class SocketServer

{

public:

SocketServer(int port=5000):port(port){}

~SocketServer(){};

void getinfo(){cout<<this->port<<endl;}

int CreateSocket();

int Bind();

int Listen();

void Accept();

void ReceiveAndClose();

private:

int port;

int socket\_fd;

int data\_socket\_fd;

struct sockaddr\_in serv\_addr;

data Data;

};

**4.6 Business Logic**

**4.6.1 CreateSharedMemory**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | CreateSharedMemory |  | |
| Input | Shmid,data | int | |
| Output | 0 or -1 | int | none |
| Process | Creating shared memory for storing database |  | |

**4.6.2 Accept**

|  |  |  |
| --- | --- | --- |
| Name | Accept |  |
| Input | pid | int |
| Output | none | void |
| Process | Accepting connection with client |  |

**4.6.3 getinfo**

|  |  |  |
| --- | --- | --- |
| Name | getinfo |  |
| Input | none |  |
| Output | Address and port | void |
| Process | Display address and port number |  |

**4.6.4 CheckData**

|  |  |  |
| --- | --- | --- |
| Name | CheckData |  |
| Input | none |  |
| Output | Name id number | void |
| Process | Displays employee list |  |

**4.6.5 DisplayMenu**

|  |  |  |
| --- | --- | --- |
| Name | DisplayMenu |  |
| Input | Name id number | int |
| Output | choice | int |
| Process | Main menu for client to add edit and delete data |  |

**4.6.6 HandleEdit**

|  |  |  |
| --- | --- | --- |
| Name | HandleEdit |  |
| Input | Servermsg | int |
| Output | 0 or -1 | int |
| Process | Handles edit for input data |  |

**5 Language Support**

System Programming, IPC, Socket Programming

**5.1 Deployment Considerations**

Hardware, Software, File Storage, Session Storage

**5.2 Integration Requirements**

An Intergration is connecting systems, applications and devices together so that you have a better flow of data and processes

**5.3 Network**

The Networks used in the Organization Software are Local Area Network(LAN).

**5.4 Operating System**

Linux.

**6 References**

**7 Appendix**