**Concurrent/Multithreading Application for real time Booking**

**High Level Design & Low Level Design**

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

**Document Control :**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Revision History** | | | | | | | |
|  |  |  |  |  |  |  |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | **Approver Signature** | |
|  |  |  |  | | |  | |
|  |  |  |  | | |  | |
|  |  |  |  | | |  | |

[**1. Introduction**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3znysh7) **5**

[1.1. Intended Audience](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2et92p0) 5

[1.2. Acronyms/Abbreviations](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.tyjcwt) 5

[1.3. Project Purpose](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3dy6vkm) 5

[1.4. Key Project Objectives](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1t3h5sf) 5

[1.5. Project Scope and Limitation](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.4d34og8) 5

[*1.5.1. In Scope*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2s8eyo1) 6

[*1.5.2. Out of scope*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.17dp8vu) 6

[1.6. Functional Overview](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3rdcrjn) 6

[1.7. Assumptions, Dependencies & Constraints](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.26in1rg) 6

[1.8. Risks](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.lnxbz9) 6

[**2. Design Overview**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.35nkun2) **7**

2[.1 Design Objectives](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1ksv4uv) 8

[*2.1.1. Recommended Architecture*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.44sinio) 8

[2.2. Architectural Strategies](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2jxsxqh) 8

[*2.2.1. Design Alternative*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.z337ya) 8

[*2.2.2. Reuse of Existing Common Services/Utilities*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3j2qqm3) 8

[*2.2.3. Creation of New Common Services/Utilities*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1y810tw) 9

[*2.2.4. User Interface Paradigms*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.4i7ojhp) 9

[*2.2.5. System Interface Paradigms*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2xcytpi) 9

[*2.2.6. Error Detection / Exceptional Handling*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1ci93xb) 9

[*2.2.7. Memory Management*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3whwml4) 9

[*2.2.8. Performance*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2bn6wsx) 9

[*2.2.9. Security*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.qsh70q) 9

[*2.2.10. Concurrency and Synchronization*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3as4poj) 9

[*2.2.11. Housekeeping and Maintenance*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1pxezwc) 9

[**3. System Architecture**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.49x2ik5) **10**

[3.1. System Architecture Diagram. (Not Necessary)](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2p2csry) 11

[3.2. System Use-Cases](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.147n2zr) 12

[3.3. Subsystem Architecture](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3o7alnk) 13

[3.4. System Interfaces](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.23ckvvd) 14

[*3.4.1. Internal Interfaces*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.ihv636) 14

[*3.4.2. External Interfaces*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.32hioqz) 14

[**4. Detailed System Design**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1hmsyys) **14**

[4.1. Key Entities](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.41mghml) 14

[4.2. Detailed-Level Database Design](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2grqrue) 14

[*4.2.1. Data Mapping Information 1*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.vx1227)4

[*4.2.2. Data Conversion 1*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3fwokq0)4

[4.3. Archival and retention requirements](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1v1yuxt) 14

[4.4. Disaster and Failure Recovery](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.4f1mdlm) 15

[4.5. Business Process workflow](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2u6wntf) 15

[4.6. Business Process Modeling and Management (as applicable)](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.19c6y18) 15

[4.7. Business Logic](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3tbugp1) 15

[4.8. Variables](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.28h4qwu) 15

[4.9. Activity / Class Diagrams (as applicable)](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.nmf14n) 15

[4.10. Data Migration](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.37m2jsg) 15

[*4.10.1. Architectural Representation*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1mrcu09) 15

[*4.10.2. Architectural Goals and Constraints*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.46r0co2) 15

[*4.10.3. Logical View*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2lwamvv) 15

[*4.10.4. Architecturally Significant Design Packages*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.111kx3o) 15

[*4.10.5. Data model*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3l18frh) 15

[*4.10.6. Deployment View*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1egqt2p) 16

[**5. Environment Description**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3ygebqi) **16**

[5.1. Time Zone Support](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2dlolyb) 16

[5.2. Language Support](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.sqyw64) 16

[5.3. User Desktop Requirements](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3cqmetx) 16

[5.4. Server-Side Requirements](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1rvwp1q) 16

[*5.4.1. Deployment Considerations*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.4bvk7pj) 16

[*5.4.2. Application Server Disk Space*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2r0uhxc) 16

[*5.4.3. Database Server Disk Space*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1664s55) 17

[*5.4.4. Integration Requirements*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3q5sasy) 17

[*5.4.5. Jobs*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.25b2l0r) 17

[*5.4.6. Network*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.kgcv8k) 17

[*5.4.7. Others*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.34g0dwd) 17

[5.5. Configuration](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1jlao46) 17

[*5.5.1. Operating System*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.43ky6rz) 17

[*5.5.2. Database*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2iq8gzs) 17

[*5.5.3. Network*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.xvir7l) 17

[*5.5.4. Desktop*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3hv69ve) 17

[**6. References**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1x0gk37) **17**

[**7. Appendix**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.4h042r0) **18**

**INTRODUCTION:**

* It is a web-based application. The main purpose of “Concurrent/Multithreading Application for Real Time Booking” is to provide a convenient way for a customer to book cars and hotels for traveling and staying purposes. The objective of this project is to develop a system that automates the processes and activities of a travel agency. In this project, we will make an easier task of booking cars and hotels. In the present system a customer has to approach various agencies to find details of places and to book tickets. This often requires a lot of time and effort.
* This application is developed by using the sockets to exchange the information between the admin and user and by using the multithreading to enable more than one user at a time without requiring multiple copies of the program running on the computer also using the database to enabling users to store data in a structured form and then access it.
* Each device connected to the Internet has a unique IP address which other machines use to find the device. IP addresses such as 192.168.1.1 (in IPv4).

**1.1   Intended Audience**

|  |  |
| --- | --- |
| BU Authority |  |
|  |  |

**1.2  Acronyms/Abbreviations**

|  |  |
| --- | --- |
| TCP | Transmission Control Protocol |
| IP | Internet Protocol |
| IPv4 | Internet Protocol version 4 |
|  |  |

**1.3  Project Purpose**

The purpose of this project is to get a solid grasp on the fundamentals of the Socket API. Writing such an application in C gives a basic understanding on how the client - server architecture works and overall, on how to use the Socket API to establish communication between client and server applications with the goal of information exchange between the two.

**1.4 Key Project Objectives**

This application is develop to provide Booking services to the customers. We have developed this application to provide a search platform where a customer can find some locations and book their tickets according to their choices.

**1.5  Project Scope and Limitation**

SCOPE:

* + Establish communication between server and client using sockets.
  + User can easily search the bookings, cancel their booking and modify their details.
  + Admins can add new locations, view total no of bookings.
  + More user friendly.

LIMITATION:

* Multiple clients can connect to only one server.
* One client can get the IP address of one domain at a time.

### 1.5.1 In Scope

* Clients/servers should provide a valid file name
* when the server is sending, the client should respond to  read the file.
* When the client is sending,the server should respond to read the file

### 1.5.2  Out of scope

          Client/server can transfer different types of files..

## Functional Overview

1.Client:

In client terminal first it will login through credentials after giving correct credentials one can see the packages and do bookings according to their respective locations and also we can modify the locations. We can easily search and cancel the reservations.

2.Server:

The server process waits for requests from a client. To do this, the server first establishes (binds) an address that clients can use to find the server. When the address is established, the server waits for clients to request a service.

## 1.7  Assumptions, Dependencies & Constraints

OPERATING SYSTEMS:

Operating environment for implementing TCP are:

* Client/server system
* Operating system: Linux
* Platform: Intel x86

## 1.8  Risks

            No Risk(As it is for educational purpose)

# **2.Design Overview**

1. Start

This is the start block which indicates the start of the program.  
which will accept the client and server credentials (like the username and password). On validation of these credentials the system will allow the client and server to further communicate else if the credentials are wrong it will display an error message indicating “Invalid Credentials”.

3.Start

This is the start block which indicates the start of the program.  
which will accept the client and server credentials (like the username and password). On validation of these credentials the system will allow the client and server to further communicate else if the credentials are wrong it will display an error message indicating “Invalid Credentials”.

3.Client Login

This is the module used for the client login where the client, if new, has to register with an username and password and then login by entering the same credentials (username and password). If already registered the client can login with the credentials.

1. Login Credentials

In this module the credentials entered by the server/client are then validated by the system. If the server/client enters valid credentials then it will move to the further step else the system will prompt the server/client with an error message.

5. Admin Login

In this module, as we know admins are fixed initially we take login attempts

is equal to zero and by using if conditions we can check admin login.

6. Admin Menu

In this module, We can perform different operations like to enter new trip,

display trip, display reservation, update trip and exit.

7. Customer Login

In this module, we have multiple number of users, so we have used text file to save the existing accounts. If the user entered details matches with the details in file then we can proceed to dashboard. If the account is new, then by entering details like name, phone number, age, gender and email then we can create a account for customer

8. Customer Menu

In this module, we can perform different operations like mew reservation, confirm reservation and cancel reservation and exit

1. Server provides respective IP address

When a domain name corresponding to the client entered domain name is found, the server will give the respective IP address to the client. 

1. Display error message
2. End

This ensures that the program has terminated.

**2.1 Design Objectives**

       Create login credential page for both admin and customer.

       Take request from the server to the client after successful login.

       Server will check whether the credentials are exist or not.

       If exists, server will send IP address to client.

### **2.1.1 Recommended Architecture**

Generic

## Architectural Strategies

* Header files
* Structures
* Macros

### **Design Alternative**

NA

**2.2.2 Reuse of Existing Common Services/Utilities**

#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

#include<string.h>

#include<arpa/inet.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<netdb.h>

#include<regex.h>

#include<unistd.h>

### **2.2.3 Creation of New Common Services/Utilities**

           NA

### **2.2.4 User Interface Paradigms**

                          Command Line Interface: Terminal

### **2.2.5 System Interface Paradigms**

             Command Line Interface: Terminal

### **2.2.6 Error Detection / Exceptional Handling**

             Error detection :

1. IP address does not exist
2. Errors will be handled by perror

### **2.2.7 Memory Management**

NA

### **2.2.8 Performance**

NA

### **2.2.9 Security**

                  For security purposes the system asks for login credentials from server

and client.

### **2.2.10 Concurrency and Synchronization**

      Applicable

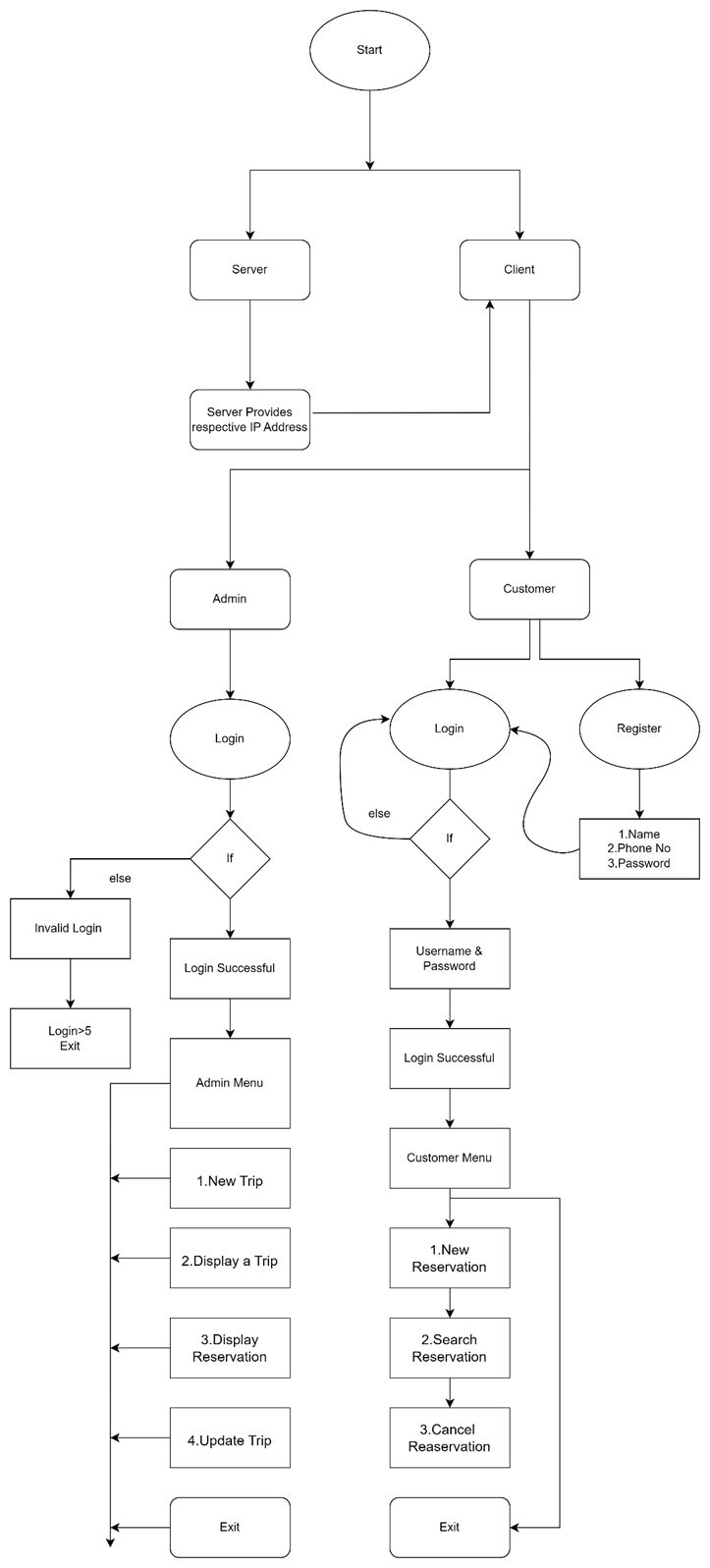
### **2.2.11 Housekeeping and Maintenance**

        NA

**3.  System Architecture**

**LEVEL 0 DFD:**

**LEVEL 1 DFD:**



## 3.1 System Architecture Diagram

**3.3  Subsystem Architecture**

    NA

## 3.4  System Interfaces

      NA

### 3.4.1 Internal Interfaces

     NA

### 3.4.2 External Interfaces

     NA

**4.Detailed System Design**

The code starts by declaring the struct sockaddr\_in and hostent. After that client socket will be created. Using connect() the client establishes connection with the server. After a successful connection client must enter the domain name the validation of this domain name will be done by checkURL().

The message will be sent to the server.

The server will first do binding after that it will be listening on a particular port which will be the same for server and client.

Accept() will be called which will accept the connection from the client. The received domain name will be accepted by the server and will search for the equivalent domain name from the local DNS cache, Root server, tld server or Authoritative server.

After fetching the IP address server will send the IP address to the client.

## 4.1  Key Entities

* Valid domain/host name
* IP Address

## 4.2  Detailed-Level Database Design

    NA

### 4.2.1  Data Mapping Information

             Mapping the IP address from server side is done by gethostbyname()

### 4.2.2  Data Conversion

Converting the IP address(IPv4 format) from binary to standard text format using   inet\_ntop()

## 4.3  Archival and retention requirements

NA

## Disaster and Failure Recovery

## We don’t have any control over the system.In case of failure,source code is safe.

## Use of Git.

## 4.5  Business Process workflow

    NA

## 4.6  Business Process Modeling and Management (as applicable)

                              NA

## 4.7  Business Logic

     NA

## 4.8  Variables

       NA

## 4.9  Activity / Class Diagrams (as applicable)

Pseudocode for Server Side:

       Create a server socket

       Bind socket to specific port where client will connect with the server

       Listen for connections on the socket

       Loop

         Accept new connection(AcceptRetStatus)

         Read and Write data into AcceptRetStatus to communicate with the client

         close AcceptRetStatus

       End Loop

       Close

       Pseudocode for Client Side:

       Create a client socket

       Establish connection with server by calling connect()

       Read and Write data in ConnectRetStatus to communicate with server

       Close

## 4.10 Data Migration

       NA

### 4.10.1 Architectural Representation

          NA

### 4.10.2 Architectural Goals and Constraints

                     The project is just for educational purposes.

### 4.10.3 Logical View

          NA

### 4.10.4 Architecturally Significant Design Packages

          NA

### 4.10.5 Data model

         NA

**Legacy system data model**

**Proposed system data model**

**Interface data model**

### 4.10.6 Deployment View

  NA

# Environment Description

GCC: In Linux, the GCC stands for GNU Compiler Collection. It is a compiler system for the various programming languages. It is mainly used to compile the C and C++ programs.

Socket Programming: Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.

UBUNTU: Ubuntu is an open-source operating system (OS) based on the Debian GNU/Linux distribution. Ubuntu incorporates all the features of a Unix OS with an added customizable GUI, which makes it popular in universities and research organizations. Ubuntu is primarily designed to be used on personal computers, although a server edition does also exist.

GITHUB: GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. This tutorial teaches you GitHub essentials like repositories, branches, commits, and pull requests.

## 5.1 Time Zone Support

                       NA

## 5.2 Language Support

                      NA

## 5.3 User Desktop Requirements

   Linux, Ubuntu

## 5.4 Server-Side Requirements

   Linux, Ubuntu

### 5.4.1 Deployment Considerations

          NA

### 5.4.2 Application Server Disk Space

          NA

### 5.4.3 Database Server Disk Space

          NA

### 5.4.4 Integration Requirements

            NA

### 5.4.5 Jobs

                                      NA

### 5.4.6 Network

           NA

### 5.4.7 Others

            NA

## 5.5  Configuration

                     NA

### 5.5.1 Operating System

Linux desktop editions with 8 GB RAM- A GUI-based LINUX system must be   used

### 5.5.2 Database

      NA

### 5.5.3  Network

INTERNET

### 5.5.4  Desktop

* CPU : Intel i3/i5/i7 generation 3 and later
* RAM: 4GB or greater - For optimal performance, 6GB or 8GB are recommended if you will be running multiple browser tabs and/or multiple applications at the same time
* Internal memory:476 GB SSD/HDD.

