**Secure Socket implementation for data management (Online Library Management)**

**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** | |
| 18/11/2022 | Version\_1.0 | Penta Lakshmi Priya | Block Diagram | | | Prasanth | |
| 19/11/2022 | Version\_1.1 | Dugunepalli Sunil Kumar | Document Designed | | | Prasanth | |
| 20/11/2022 | Version\_1.2 | GVV Priyanka | Changes in the design Overview | | | Prasanth | |
| 21/11/2022 | Version\_1.3 | P Vigneshwar | Changes made in detailed system design | | | Prasanth | |
| 22/11/2022 | Version\_1.4 | G Siva Priya reddy | Changes made in design objectives | | | Prasanth | |
| 23/11/2022 | Version\_1.5 | Dugunepalli Sunil kumar | Changes in the design Overview | | | Prasanth | |

[**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**

**1.Introduction:**

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.

Establish communication between the server-client to connect and access data from the database to download the required books from the database which contains list of URLs of the books to download.

**1.1   Intended Audience**

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

**1.2  Acronyms / Abbreviations**

|  |  |
| --- | --- |
| TCP | Transmission Control Protocol |
| IP | Internet Protocol |
| IPV4 | Internet Protocol Version 4 |
| Db | Database |
| API | Application Programming Interface |

**1.3 Project Purpose**

The purpose of this project is to provide access to client to download required books from the database which contains the list of book URLs by providing socket communication between server-client.

**1.4 Key Project Objectives**

The objective is to create access to client to get the required books digitally using database and sockets.

**1.5 Project Scope and Limitation**

SCOPE:

* The scope of the project includes insertion of book URLs to database which is under server control. Similarly, when the client needs URLs of books from database, server provides access to client over the database.
* So that client will be able to download required book digitally based on the URLs provided from database based on client requirement.

LIMITATION:

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

**1.5.2 Out of scope**

          Client/server can access/transfer different URLs based on requirement.

**1.6 Functional Overview**

* Client:

In client terminal, with the appropriate IP address communicate with the server.

Client should be able to download the books by connecting with the database.

* Server:

In the server terminal the server should first login with its credentials after that it will receive the request from the client. In the response to that request the server will send an appropriate IP address to the client. The communication/chat is established between the server-client.

* Database:

In database, the URLs are uploaded by the server with certain login credentials and the list of URLs of books which are able to download by client.

**1.7 Assumptions, Dependencies & Constraints**

OPERATING SYSTEMS:

* Client/server system
* Operating system: Linux
* Platform: Ubuntu/C++

**1.8 Risks**

            No Risk (As it is for educational purpose).

1. **Design Overview**

**SOCKET:**

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

**BIND:**

When a socket has both an IP address and a port number it is said to be 'bound to a port', or 'bound to an address'. A bound socket can receive data because it has a complete address. Binding is the process of allocating a port number to a socket.

**LISTEN:**

The listen () function applies only to stream sockets. It indicates a readiness to accept client connection requests, and creates a connection request queue of length backlog to queue incoming connection requests. Once full, additional connection requests are rejected.

**ACCEPT:**

The accept () call is used by a server to accept a connection request from a client. When a connection is available, the socket created is ready for use to read data from the process that requested the connection. The call accepts the first connection on its queue of pending connections for the given socket.

**CONNECT:**

The connect () call on a stream socket is used by the client application to establish a connection to a server. The server must have a passive open pending. A server that is using sockets must successfully call bind () and listen () before a connection can be accepted by the server with accept ().

**SEND/REC:**

The server uses the socket that is returned from the accept () call.

These functions return the amount of data that was sent or received. Because stream sockets send and receive information in streams of data, it can take more than one send () or rec () to transfer all of the data. It is up to the client and the server to agree on some mechanism to signal that all of the data has been transferred.

**EXEC:**

The execv function is most commonly used to overlay a process image that has been created by a call to the fork function. file. is the filename of the file that contains the executable image of the new process. argv is a pointer to an array of pointers to null-terminated character strings. exec is a functionality of an operating system that runs an executable file in the context of an already existing process, replacing the previous executable.

**DATABASE:**

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS).

**SIGNALS:**

A signal is a software generated interrupt that is sent to a process by the OS because of when user press ctrl-c or another process tell something to this process. There are fix set of signals that can be sent to a process. signal is identified by integers.

**2.1 Design Objectives**

       Create a login for the server and allow server to upload the URLs of books.

Client should able to access the list of book URLs and Download if required.

**2.1.1 Recommended Architecture**

Generic

**2.2 Architectural Strategies**

* Header files
* Structures
* Macros

**2.2.1 Design Alternative**

NA

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

* 1. Database
  2. File Handling
  3. Signals
  4. Exec Family
  5. Sockets

**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.
3. Binding Failure.
4. Connect Failure.
5. Signal controls.

**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**

      NA

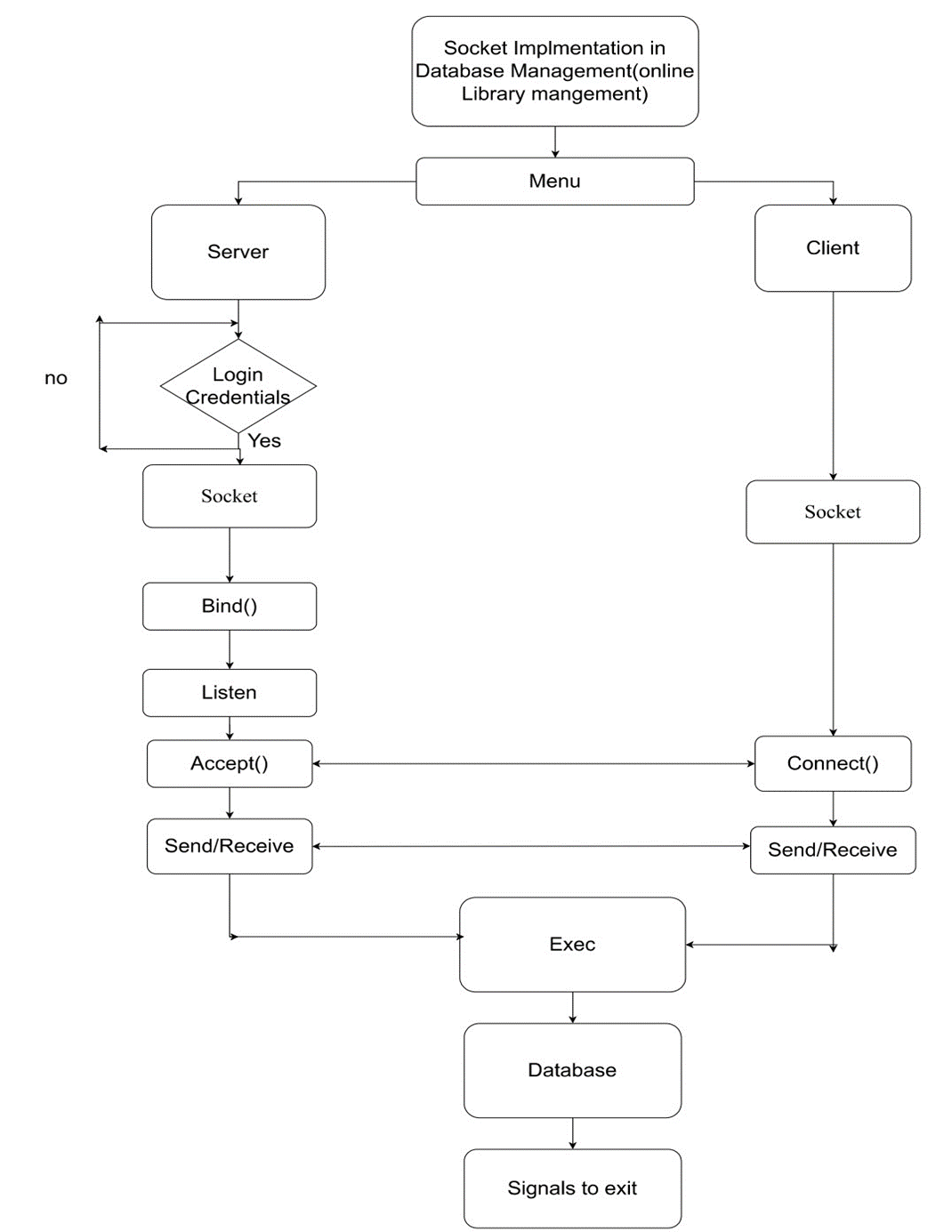
**2.2.11 Housekeeping and Maintenance**

        NA

**3.  System Architecture**

**LEVEL 0 DFD:**

**LEVEL 1 DFD:**

**3.1 System Architecture Diagram:**

**3.2 System Use-Cases**

**

**3.3 Subsystem Architecture**

    NA

**3.4 System Interfaces**

      NA

**3.4.1 Internal Interfaces**

     NA

**3.4.2 External Interfaces**

     NA

1. **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.

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.

Connection will be established between server-client to communicate with each other and client should able to download book by using URLs provided in database.

**4.1 Key Entities**

NA

**4.2 Detailed-Level Database Design**

    NA

**4.2.1 Data Mapping Information**

             NA

**4.2.2 Data Conversion**

NA

**4.3 Archival and retention requirements**

NA

**4.4 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

1. **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**

*[Describe the Network configuration requirements here. Details of all the Network Components etc.]*

**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: 512 GB SSD/HDD.

1. **References**

<https://man7.org/linux/man-pages/index.html>

[Introduction to Sockets Programming in C using TCP/IP](https://www.csd.uoc.gr/~hy556/material/tutorials/cs556-3rd-tutorial.pdf)

<https://www.ibm.com/docs/en/zos/2.2.0?topic=reference-library-functions>

1. **Appendix**

**Change Log**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **QMS Template Version Control (Maintained by QA)** | | | | | |
|  |  |  |  |  |  |
| **Date** | **Version** | **Author** | | **Description** | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |