**Customize TCP Socket for real-time Delivery Management System(Use Case : Food Delivery)**

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** | |
| 15/11/2022 | Version 0.1 |  |  | | | |  | |
|  |  |  |  | | | |  | |
|  |  |  |  | | | |  | |

[**1. Introduction**](#_heading=h.3znysh7) **5**

[1.1. Intended Audience](#_heading=h.2et92p0) 5

[1.2. Acronyms/Abbreviations](#_heading=h.tyjcwt) 5

[1.3. Project Purpose](#_heading=h.3dy6vkm) 5

[1.4. Key Project Objectives](#_heading=h.1t3h5sf) 5

[1.5. Project Scope and Limitation](#_heading=h.4d34og8) 5

[*1.5.1. In Scope*](#_heading=h.2s8eyo1) 6

[*1.5.2. Out of scope*](#_heading=h.17dp8vu) 6

[1.6. Functional Overview](#_heading=h.3rdcrjn) 6

[1.7. Assumptions, Dependencies & Constraints](#_heading=h.26in1rg) 6

[1.8. Risks](#_heading=h.lnxbz9) 6

[**2. Design Overview**](#_heading=h.35nkun2) **7**

2[.1 Design Objectives](#_heading=h.1ksv4uv) 8

[*2.1.1. Recommended Architecture*](#_heading=h.44sinio) 8

[2.2. Architectural Strategies](#_heading=h.2jxsxqh) 8

[*2.2.1. Design Alternative*](#_heading=h.z337ya) 8

[*2.2.2. Reuse of Existing Common Services/Utilities*](#_heading=h.3j2qqm3) 8

[*2.2.3. Creation of New Common Services/Utilities*](#_heading=h.1y810tw) 9

[*2.2.4. User Interface Paradigms*](#_heading=h.4i7ojhp) 9

[*2.2.5. System Interface Paradigms*](#_heading=h.2xcytpi) 9

[*2.2.6. Error Detection / Exceptional Handling*](#_heading=h.1ci93xb) 9

[*2.2.7. Memory Management*](#_heading=h.3whwml4) 9

[*2.2.8. Performance*](#_heading=h.2bn6wsx) 9

[*2.2.9. Security*](#_heading=h.qsh70q) 9

[*2.2.10. Concurrency and Synchronization*](#_heading=h.3as4poj) 9

[**3. System Architecture**](#_heading=h.49x2ik5) **10**

[3.1. System Architecture Diagram. (Not Necessary)](#_heading=h.2p2csry) 11

[3.2. System Use-Cases](#_heading=h.147n2zr) 12

[3.3. Subsystem Architecture](#_heading=h.3o7alnk) 13

[3.4. System Interfaces](#_heading=h.23ckvvd) 14

[*3.4.1. Internal Interfaces*](#_heading=h.ihv636) 14

[*3.4.2. External Interfaces*](#_heading=h.32hioqz) 14

[**4. Detailed System Design**](#_heading=h.1hmsyys) **14**

[4.1. Key Entities](#_heading=h.41mghml) 14

[4.2. Detailed-Level Database Design](#_heading=h.2grqrue) 14

[*4.2.1. Data Mapping Information 1*](#_heading=h.vx1227)4

[*4.2.2. Data Conversion 1*](#_heading=h.3fwokq0)4

[4.3. Archival and retention requirements](#_heading=h.1v1yuxt) 14

[4.4. Disaster and Failure Recovery](#_heading=h.4f1mdlm) 15

[4.5. Business Process workflow](#_heading=h.2u6wntf) 15

[4.6. Business Process Modeling and Management (as applicable)](#_heading=h.19c6y18) 15

[4.7. Business Logic](#_heading=h.3tbugp1) 15

[4.8. Variables](#_heading=h.28h4qwu) 15

[4.9. Activity / Class Diagrams (as applicable)](#_heading=h.nmf14n) 15

[4.10. Data Migration](#_heading=h.37m2jsg) 15

[*4.10.1. Architectural Representation*](#_heading=h.1mrcu09) 15

[*4.10.2. Architectural Goals and Constraints*](#_heading=h.46r0co2) 15

[*4.10.3. Logical View*](#_heading=h.2lwamvv) 15

[*4.10.4. Architecturally Significant Design Packages*](#_heading=h.111kx3o) 15

[*4.10.5. Data model*](#_heading=h.3l18frh) 15

[*4.10.6. Deployment View*](#_heading=h.1egqt2p) 16

[**5. Environment Description**](#_heading=h.3ygebqi) **16**

[5.1. Time Zone Support](#_heading=h.2dlolyb) 16

[5.2. Language Support](#_heading=h.sqyw64) 16

[5.3. User Desktop Requirements](#_heading=h.3cqmetx) 16

[5.4. Server-Side Requirements](#_heading=h.1rvwp1q) 16

[*5.4.1. Deployment Considerations*](#_heading=h.4bvk7pj) 16

[*5.4.2. Application Server Disk Space*](#_heading=h.2r0uhxc) 16

[*5.4.3. Database Server Disk Space*](#_heading=h.1664s55) 17

[*5.4.4. Integration Requirements*](#_heading=h.3q5sasy) 17

[*5.4.5. Jobs*](#_heading=h.25b2l0r) 17

[*5.4.6. Network*](#_heading=h.kgcv8k) 17

[*5.4.7. Others*](#_heading=h.34g0dwd) 17

[5.5. Configuration](#_heading=h.1jlao46) 17

[*5.5.1. Operating System*](#_heading=h.43ky6rz) 17

[*5.5.2. Database*](#_heading=h.2iq8gzs) 17

[*5.5.3. Desktop*](#_heading=h.3hv69ve) 17

[**6. References**](#_heading=h.1x0gk37) **17**

[**7. Appendix**](#_heading=h.4h042r0) **18**

# 

**1. Introduction:**

Food ordering system these days has one of the fastest growing market, though being a new idea. In this project we have developed something like the same to earn from and serve the nation in a much better way possible. Nowadays, people are more regular to dine-in at restaurant for their meals. The online food ordering system provides convenience for the customers that are nothing special but the general busy people of the society.

It overcomes the demerits of the manual hotel or mess system and the old fashioned queuing system. This system enhances the readymade of foods than people. Therefore, this system enhances the speed of getting food in person’s plate and quality and manner of taking the order from the customer. It provides a better communication platform. The user’s details are stored using the electronic media.

The online food ordering system provides the menu online and the customers can easily place the order by just clicking the mouse or by touching a button on their smart phones. Also with the food ordering system online, people can easily track their orders, and admin can maintain customer’s database and advance the food delivery system.

This food ordering system allows the user to select the desired food items from a list of available menu items provided by the local hotel or restaurant. The user can place orders for the food items of their like from the list.

**TCP Socket:**

TCP Socket is used to implement customer assistance. Transmission Control Protocol, or TCP, is a group of protocols for interconnecting network devices on the internet. Additionally, a private computer network use TCP as its communications protocol (an intranet or extranet).

TCP employs the client-server model of communication, where a user or machine (a client) requests a service from another computer in the network (a server), such as sending a webpage. Data packets are rearranged by TCP in the designated order.

The integrity and receipt of the sent data in its original order are absolutely guaranteed. Before sending any user data, TCP performs flow control and necessitates the sending of three packets to establish a socket connection.

## 1.1 Intended Audience

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

## 1.2 Acronyms/Abbreviations

|  |  |
| --- | --- |
| TCP | Transmission Control Protocol |
| IP | Internet Protocol |
| GDB | GNU Debugger |
| FH | File Handling |

## 1.3 Project Purpose

Customers can easily place orders as they like using the online meal ordering system, which sets up a food menu online. The management keeps the food delivery service improving. The conventional queueing system is defeated by it. Customers who have questions about the nutritional content of food can consult with the manager.

## 1.4 Key Project Objectives

Customers can easily place orders from a food menu that is set up online by the food delivery management system.

**1.5 Project Scope and Limitation**

Delivering meals directly from a restaurant to the consumer without interruption.

1.Establish communication between customer and manager

2. Transferring data from customer to manager.

**Limitations:**

* Quality of food may suffer.
* Food delivery services are often late. You may get lazy.
* Deliverymen Put Themselves in Danger

### 1.5.1 In Scope

* when the Manager is sending text, the customer should respond with the text.
* When the customer is sending, the Manager should respond to respond with the text.

### 1.5.2 Out of scope

Customer/Manager can continue the conversation till the customer send the “Thank You” text.

## 1.6 Functional Overview

Food Ordering is having many modules, which make the software more efficient and user friendly. The modules make the maintenance of the database easier. Every module is divided on the basis of the senarios. The main three senarios are

* Kitchen
* Manager
* Floor

The different modules in this project are described below:

* Food Items
* Order
* Staffs
* Cancellations
* Expenses
* Home delivery

**Menu Module:**

In this module the number of food, their prices, oners ete, like details will stores:

* Insert food item
* Insert price
* Delete food item.
* Delete price
* Modify price.

**Order Module:**

In this module the type of order, the amount ete like details will stores.

* Insert Order
* Insert type of order.
* Insert price
* Delete order
* Delete price
* Modify price.

**Staffs Module:**

In this module the number of staffs, their names, salary details, shift timings etc will stores:

* Insert staff.
* Delete staff

**Cancelations Modules:**

In this module the canceled foods details are stores

* Insert Canceled order.
* Delete canceled order

**Expenses Module:**

In this module the expenses of the money are stores.

* Insert Expenses.
* Delete Expenses
* Modify Expenses

**Home Delivery:**

In this module, there will section which will store the data about home delivery orders

* Add order
* Add bill
* Delete order
* Delete bill

**CUSTOMER SUPPORT PART:**

* Whenever a user chooses customer service.
* The customer service manager will be turned on. The manager will ask question “How can we help?”.
* Customer care agents listen to customers' issues, address their inquiries, and inform them about the food order service support.
* If the user will write text as “Thank You” ,conversation ends. Otherwise , the conversation continues.

## 1.7 Assumptions, Dependencies & Constraints

OPERATING SYSTEMS:

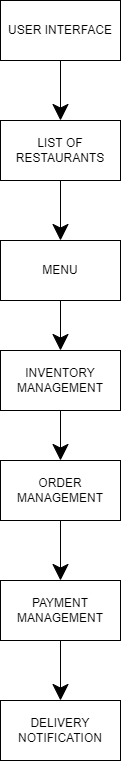
Operating environment for implementing FTP are:

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

## 1.8 Risks

No Risk (As it is for educational purpose)

# Design Overview



## 2.1 Design Objectives

  Create login credential page for both Manager and customer

       Take domain name as an input from the customer after successful login

       Manager will check whether domain names exist or not.

       If exists, Manager will send IP address to customer.

### 2.1.1 Recommended Architecture

Generic

## 2.2 Architectural Strategies

* Header files
* File Handling
* Macros

### 2.2.1 Design Alternative

NA

### 2.2.2 Reuse of Existing Common Services/Utilities

NA

### 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. Invalid Login Credentials
2. IP address does not exist
3. 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 Manager and customer.

### 2.2.10 Concurrency and Synchronization

NA

### 2.2.11 Housekeeping and Maintenance

NA

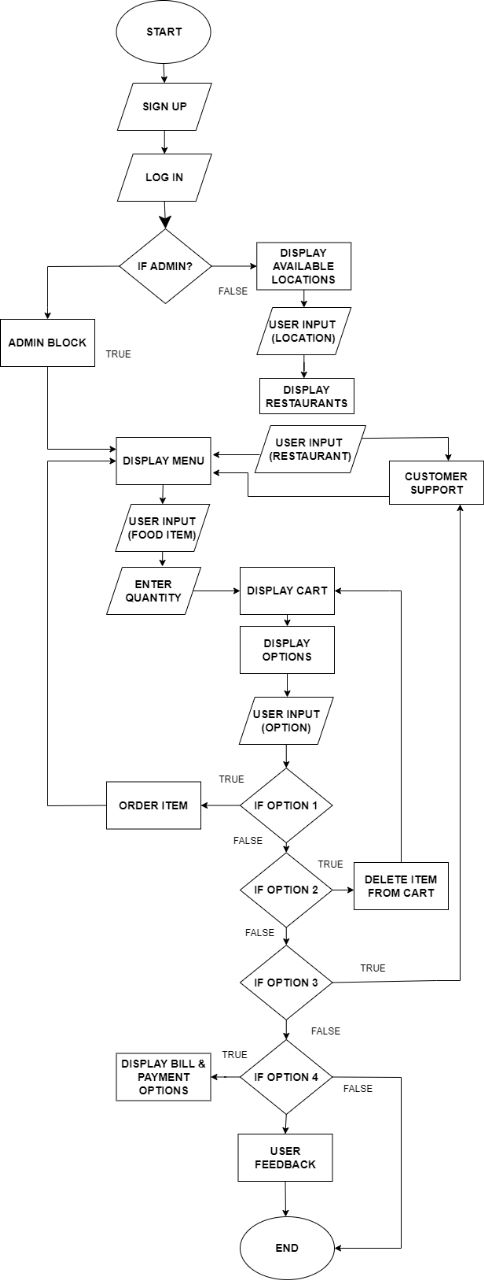
# 3. System Architecture

**LEVEL 0 DFD:** NA

**LEVEL 1 DFD:**NA

## 

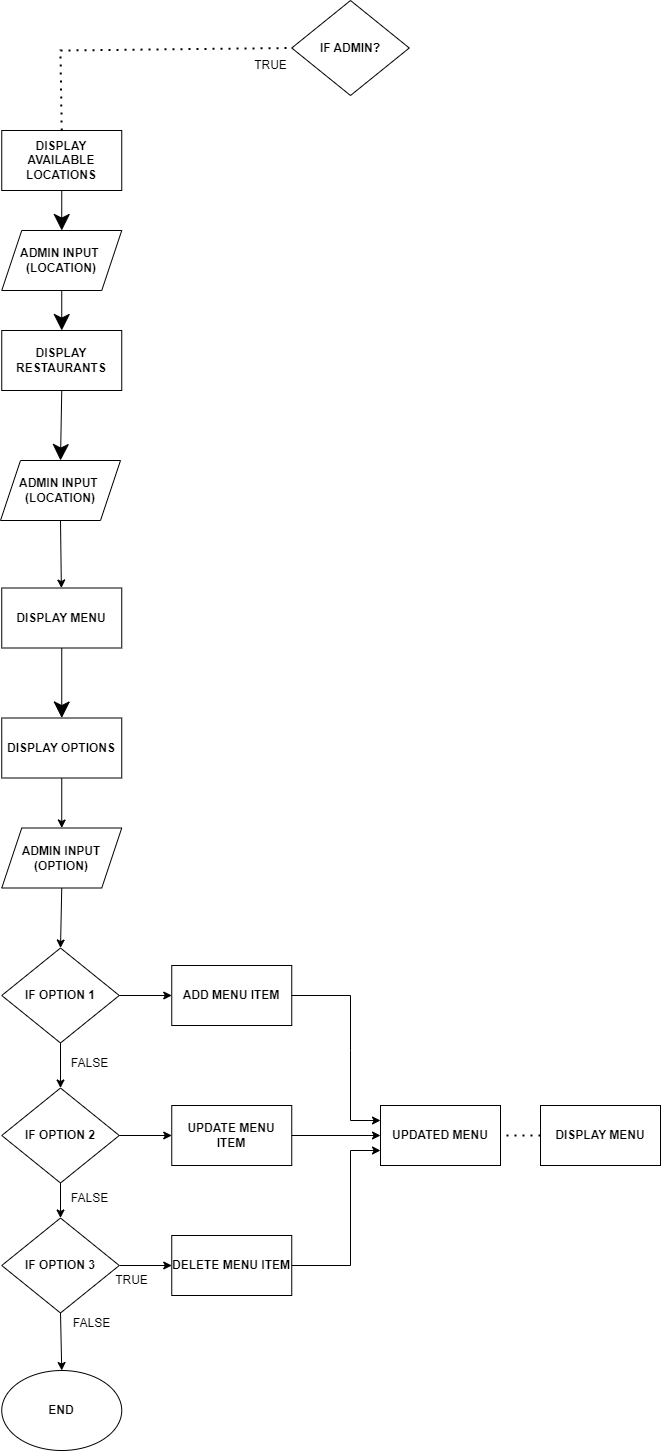
**3.1 System Architecture Diagram**



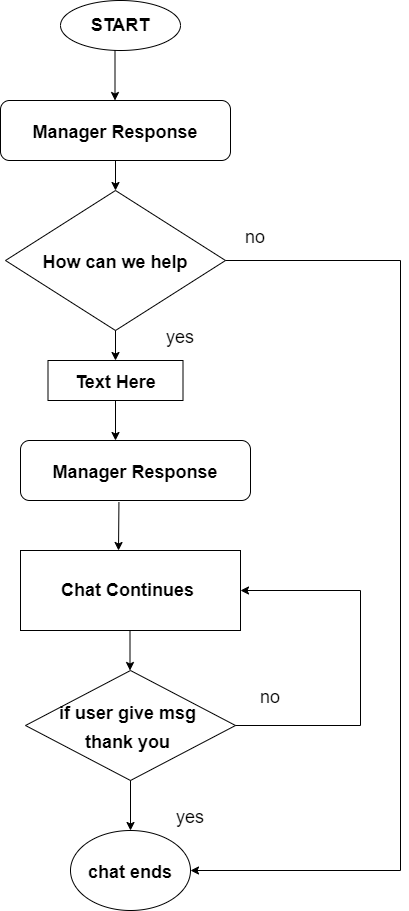
## 3.2 System Use-Case

NA

## 3.3 Subsystem Architecture (ADMIN BLOCK)



**(Customer Support)**

****

## 3.4 System Interfaces

NA

### 3.4.1 Internal Interfaces

NA

### 3.4.2 External Interfaces

NA

# Detailed System Design

* If User doesn't have an account, they can sign up and then login. Otherwise they can login directly.
* Admins can login directly without signing up.

**USER PART**

* If the user is login Successfully, user directs to Location page then has to enter the location then it displays the available restaurants. If Entered location is invalid it redirects to Locations page.
* After Selection of Hotel Display the corresponding Restaurant menu with Options.
* 1.Order 2. Delete 3.paybill 4.Cancel
* If User select the Order then order displays order successful and shows the options again.
* If User select the Delete then item is deleted from orders list.
* If User select the paybill it shows total amount to pay and gives a message Ordered Successfully and Thank you choose again.
* If User choose cancel – exit from the App.

**ADMIN PART**

* If the admin is login Successfully, admin directs to Location page then admin has to enter the location then it displays the available restaurants. If Entered location is invalid it redirects to Locations page.
* After Selection of Hotel Display the corresponding Restaurant menu with Options.
* 1.ADD Item 2.Update Item 3.Delete Item 4.Exit
* If Admin select add item then compiler ask for new item then it will directly added to the menu.
* If Admin select Update item then compiler ask to select the item from menu to update the name or price of the food item.
* If Admin select Delete then food item is delete from the menu.
* If Admin select the exit then it shoes Successfully exit from Admin Block. Thank You

**CUSTOMER SUPPORT PART:**

* Whenever a user chooses customer service.
* The customer service manager will be turned on. The manager will ask question “How can we help?”.
* Customer care agents listen to customers' issues, address their inquiries, and inform them about the food order service support.
* If the user will write text as “Thank You” ,conversation ends. Otherwise , the conversation continues.

## 4.1 Key Entities

* Valid login credentials
* IP Address

## 4.2 Detailed-Level Database Design

NA

### 4.2.1 Data Mapping Information

Mapping the IP address from manager side is done by getcustomersupport()

### 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**

Start

* It’s asking for login or signup. If he is an account he go with login else go with signup option.
* Here admin data to be stored. So admin directly choose with login.
* If login done by admin he go to admin side. If login done by user go to user side.
* After successful login, it displays the locations.
* BEGIN if he is a manager
* Click on ADD or REMOVE or Update
* IF add button is pressed
* IF ingredient does not exist

THEN add ingredient to the list.

* ELSE

Display normal menu

* ENDIF.
* IF remove button is pressed

THEN remove particular ingredient from the list.

* ENDIF.
* IF update

Then cost of the food item or name of the item will be done.

* BEGIN if he is a User
* It displays the locations.
* IF User ok with the displayed locations then he enters the valid location.
* ELSE

The user uses the Customer Support

* BEGIN

User get the Manager Response that “How can we help?”

* IF User wants to text to the Manager regarding the location queries then User will able to text to the Manager in the Chat Box and User gets the Response from the manager and the chat continues Until the User sends thank you.

THEN chat box ends

* ELSE

User directly goes to the chat box ends

* Select the locations after that choose the respected restaurant in that location.
* According to the restaurant selection the menu will be displayed.
* IF User is satisfied with Menu list then the User ordered the items.

THEN ask for the quantity of the food.

THEN show the total cost of the item.

* Show the order placed successfully.
* User enters into the Feedback Form.
* ELSE IF

User not satisfied with Menu list then the User can search the items from Search Menu.

* ELSE

The user uses the Customer Support

* BEGIN

User get the Manager Response that “How can we help?”

* IF User wants to text to the Manager regarding the Menu list queries then User will able to text to the Manager in the Chat Box and User gets the Response from the manager and the chat continues Until the User sends thank you.

THEN chat box ends

* ELSE

User directly goes to the chat box ends

* ENDIF

## 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

**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 editions does also exist.

**C++ LANGUAGE**: C++ is a cross-platform language that can be used to create high-performance applications. It was developed by Bjarne Stroustrup, as an extension to the C language. It gives programmers a high level of control over system resources and memory. The language was updated 4 major times in 2011, 2014, 2017, and 2020 to C++11, C++14, C++17, C++20.

C++ is one of the world's most popular programming languages. It is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs. It is close to C,C# and Java, it makes it easy for programmers to switch to C++ or vice versa.

**MAKEFILE**: Makefile is a program building tool which runs on Unix, Linux, and their flavors. It aids in simplifying building program executables that may need various modules. To determine how the modules need to be compiled or recompiled together, make takes the help of user-defined makefiles.

**Steps**: First Write Make. Then enter the target file name like (./food)

**GDB:** gdb is the acronym for GNU Debugger. This tool helps to debug the programs written in C, C++, Ada, Fortran, etc. The console can be opened using the gdb command on terminal.

## 5.1 Time Zone Support

India Standard Time, is the time zone observed throughout India, with a time offset of UTC+05:30.

## 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 4 GB RAM- A GUI-based LINUX system must be used

### 5.5.2 Database

NA

### 5.5.3 Network

*NA*

### 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.

# 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>

# Appendix

**Change Log**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **QMS Template Version Control (Maintained by QA)** | | | | | |
|  |  |  |  |  |  |
| **Date** | **Version** | **Author** | | **Description** | |
| 15-11-2022 | 1.0 |  | | Initial Version | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |