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

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

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

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

[1.4. Project Scope](#_heading=h.4d34og8) 5

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

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

[1.7. 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

[*2.2.11. Housekeeping and Maintenance*](#_heading=h.1pxezwc) 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. Network*](#_heading=h.xvir7l) 17

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

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

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

# 

# Introduction:

The aim of this project is to develop a Personnel Management System using C++ through a menu-driven program that allows us to add, update, delete records and project details of an employee working in an Marspro organization. The project personnel management system stores employee ID, name, department, project details and salary of the employee. Initially, it has no data. Thus, we have to add employee records choosing appropriate options in this program.

After adding the data we can perform various operations such as calculating the salary, generating various reports such as pay report, query report and employee report. This system does not allow wrong inputs as we have provided validations for various inputs such as employee name, contact number, salary, employee id, project id, department.

## 1.1 Intended Audience

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

## 1.2 Project Purpose

The project aims at the development of the online personnel management system to assist organization to keep the records of the employee along with the project they are working on and to calculate their pay stub in a more robust and efficient manner. Along with that it also provides the list of employees who are working on more than one project and able to give the details of working hours on a project for an employee.

## 1.3 Key Project Objectives:

The key purpose of this project is to manage the employee and provide project details as per the requirement from the organisation.

**1.4 Project Scop****e:**

The project aims at the development of the online personnel management system to assist organization to keep the records of the employee along with the project they are working on and to calculate their pay stub in a more robust and efficient manner. Along with that it also provides the list of employees who are working on more than one project and able to give the details of working hours on a project for an employee.

## 1.6 Functional Overview:

* **Employee Details:**

We can track all the information related to employee like adding the employee details, editing the employee detais, viewing the employee details and deleting the employee details and choosing the appropriate options the program.

* **Project Details:**

We can track all the information related to project details on which employee is working on and we can add the project details, edit the project details, delete the project details, giving the pay stub according to the working hours of the employee on the particular project and employee should be able to view the number of hours he spent on a particular project and the application should also provide the report for the employees who are working on more than one project.

**1.6.1 Functions**

**Employee Details:**

1.6.1.1 Emp\_01 => add\_employee ()

This feature adds the employee record to the file. It asks for employee\_id, employee\_name, department, salary, contact number whose record is to be created.

1.6.1.2 Emp\_02 => edit\_employee ()

This feature edits the employee record in the file. It asks for the employee\_id and asks the user to enter new employee\_name, department , salary, contact number and the list of projects done by the new employee by adding project\_id, project\_name, number of hours worked . Whose record is to be edited Upon successful editing of employee details into the employee database it will return to the main menu.

1.6.1.3 Emp\_03 =>view\_employee ()

This feature displays all the employee details present in the employee database. It displays employee\_id, employee\_name, department, contact number and list of projects done by the employee with project\_id, project name, number of hours worked.

1.6.1.4 Emp\_04 => delete\_employee ()

This feature deletes the employee details from the employee database. The user needs to provide the employee\_id to be deleted from the employee database. Upon successful deletion of employee details from the employee database it will return to the main menu.

**Project Details:**

1.6.1.5 Pro\_05 => add\_project()

This feature adds the new project details into the project database. It asks for project\_id, project name and number hours to be worked on that project. Upon successful addition of a new project details into the project database then it returns to the main menu.

1.6.1.6 Pro\_06 => edit\_project ()

This feature edits the project's details from the project database. It asks for project\_id and asks the user to enter a new project name and number of hours to be extended to the project. Upon successful editing of project details into the project database then it returns to the main menu.

1.6.1.7 Pro\_07 => view\_project ()

This feature displays the project details in the project database. It displays project\_id, project name, number of hours to be worked on that project.

1.6.1.8 Pro\_08 => delete\_project ()

This feature deletes the project details from the project database. The user needs to provide the project\_id to be deleted from the project database. Upon successful deletion of project details from the employee database it will return to the main menu.

1.6.1.9 Pro\_09 => query()

This feature presents a query on top projects should show the project on which the highest number of employees are working. If the number is more than one, then all those projects should be displayed.

1.6.1.10 Pro\_10 => pay\_report ()

This function calls salary\_view() member function of SALARY class which displays the pay report for all the employees (id, name, basic salary, bonus, deduction and final salary) according to the criteria set by the Marspro organization.

1.6.1.11 Pro\_10 => calc\_fi\_salary ()

This feature generates the pay report for a particular employee (id, name, basic salary, bonus, deduction and final salary) according to the criteria set by the Marspro organization

1.6.1.12 Pro\_11 => emp\_project report()

This feature allows an employee to view how many hours he has already spent on which project through a menu option. This option requires him to enter his employee\_id, which is validated, and the details are shown.

1.6.1.13 Pro\_12 => emp\_list\_with\_3\_projects ()

This feature shows the details of all employees who are working on 3 projects at time.

## 1.7 Assumptions, Dependencies & Constraints:

* Operating System: Linux
* Platform: Ubuntu
* Compiler: gcc

## 1.8 Risks

No Risk(As it is for educational purpose)

# Design Overview

## 1. Start

## This is the start block which indicates the start of the program. which will accept the employee and project details employee working on. On validation the system will allow the employee to further communicate else if any details are wrong.

## 2. Adding Employee details

## Once the admin adds the valid employee details then it will connect to the database then the user will now be required to check whether the details correct or not in this module.

## 3. Adding Project details

Once the admin adds the valid project details then it will connect to the database then the user will now be required to check whether the details correct or not and the project details will be added in this module.

## 4. View employee report

## Once the employee details are added then the user can view the employee report by entering employee id.

## 5. Edit details of employee

## Once the employee details is added we can edit the details of the employee from database in this module.

## 6. Delete details of employee

In this we can delete the details of employee from the employee database by entering the employee id.

## 7. Edit details of project

We can edit the details of the project which the employee working on in the project database by entering the project id.

## 8. Delete details of project

We can delete the details of project which the employee working on in the project database by entering the project id.

9.View pay report

In this we can view the pay report of the employees.

10. View query report

In this we can view the query report of the employees.

## 11. End

## This ensures that the program has terminated.

## 2.1 Design Objectives

### 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

#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

#include<string.h>

### 2.2.3 Creation of New Common Services/Utilities

NA

### 2.2.4 User Interface Paradigms

Command Line Interface: Terminal

### 2.2.11 Housekeeping and Maintenance

NA

### 2.2.5 System Interface Paradigms

Command Line Interface: Terminal

### 2.2.6 Error Detection / Exceptional Handling

Error detection :

1. Invalid Domain Name
2. Database 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

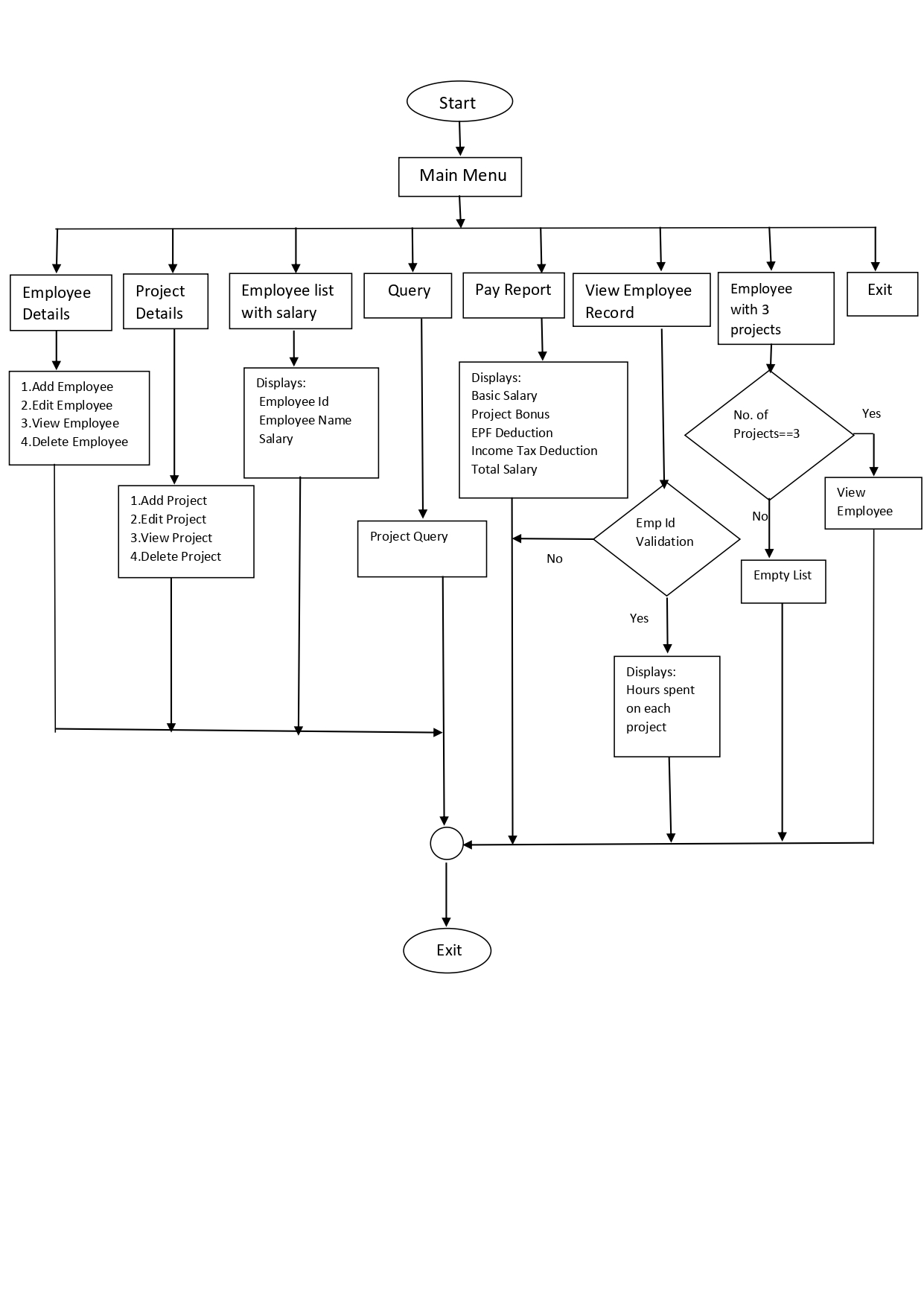
For security purposes the system asks for login credentials from server and client.

### 2.2.10 Concurrency and Synchronization

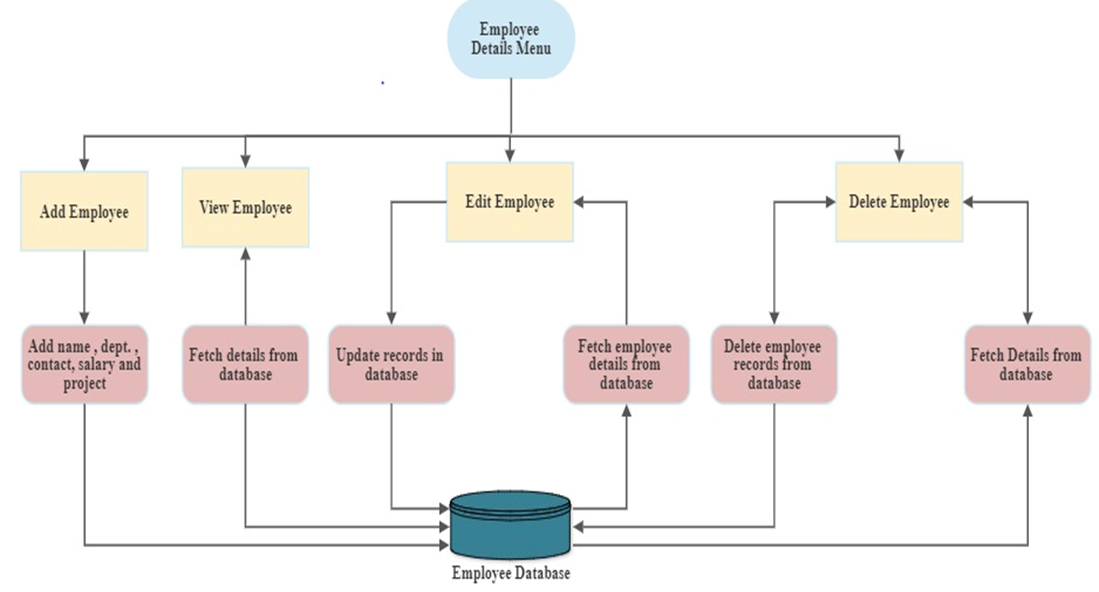
NA

**3. System Architecture**

3.1 System Architecture Diagram



## 3.2 E-R-Diagram

**

## 3.3 Subsystem Architecture

NA

## 3.4 System Interfaces

NA

### 3.4.1 Internal Interfaces

NA

### 3.4.2 External Interfaces

NA

# Detailed System Design

The purpose of this project is to manage the employee and provide project details as per the requirement from the organisation. With the help of this project, we can track all information related to Employee(add employee, edit employee, view employee and delete employee (if needed)), Project details on which employee is working (add project, edit project, delete project(if needed),query an employee, giving the pay stub according to the working hours of the employee on the particular project, employee should be able to view the number of hours he spent on a particular project and the application should also provide the report for the employees who are working on more than one project.

## 4.1 Key Entities

* emp\_id
* emp\_name
* dept
* salary
* contact
* project counter
* project id
* no. of hours
* fi\_salary
* project\_id
* project\_name
* team members

## 4.2 Detailed-Level Database Design

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

## 4.10 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

*[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: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** | |
| 28-May-2015 | 1.0 | QA Team | | Initial Version | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |