**Error! Unknown document property name.**

**Error! Unknown document property name.**

High Level Design & Low Level Design

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

**Document Control :**

| **Project Revision History** | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  |  |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | | **Approver Signature** | |
| 19-12-2022 | 0.1 | 19-12-2022\_LINUX17NOVB1\_SPRINT-1\_F | Initial Draft on Introduction and Designing | | | |  | |
| 20-12-2022 | 0.2 | 19-12-2022\_LINUX17NOVB1\_SPRINT-1\_F | Modification on Designing and System Architecture | | | |  | |
| 21-12-2022 | 0.3 | 19-12-2022\_LINUX17NOVB1\_SPRINT-1\_F | Modification on System Design and Environment | | | |  | |

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

[1.1. Intended Audience 3](#_heading=h.2et92p0)

[1.2. Acronyms/Abbreviations 3](#_heading=h.tyjcwt)

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

[1.4. Key Project Objectives 3](#_heading=h.1t3h5sf)

[1.5. Project Scope and Limitation 3](#_heading=h.4d34og8)

[*1.5.1. In Scope 3*](#_heading=h.2s8eyo1)

[*1.5.2. Out of scope 3*](#_heading=h.17dp8vu)

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

[1.7. Assumptions, Dependencies & Constraints 3](#_heading=h.26in1rg)

[1.8. Risks 3](#_heading=h.lnxbz9)

[**2. Design Overview 3**](#_heading=h.35nkun2)

[2.1. Design Objectives 3](#_heading=h.1ksv4uv)

[*2.1.1. Recommended Architecture 3*](#_heading=h.44sinio)

[2.2. Architectural Strategies 3](#_heading=h.2jxsxqh)

[*2.2.1. Design Alternative 3*](#_heading=h.z337ya)

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

[*2.2.3. Creation of New Common Services/Utilities 3*](#_heading=h.1y810tw)

[*2.2.4. User Interface Paradigms 3*](#_heading=h.4i7ojhp)

[*2.2.5. System Interface Paradigms 3*](#_heading=h.2xcytpi)

[*2.2.6. Error Detection / Exceptional Handling 3*](#_heading=h.1ci93xb)

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

[*2.2.8. Performance 3*](#_heading=h.2bn6wsx)

[*2.2.9. Security 3*](#_heading=h.qsh70q)

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

[*2.2.11. Housekeeping and Maintenance 3*](#_heading=h.1pxezwc)

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

[3.1. System Architecture Diagram. (Not Necessary) 3](#_heading=h.2p2csry)

[3.2. System Use-Cases 3](#_heading=h.147n2zr)

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

[3.4. System Interfaces 3](#_heading=h.23ckvvd)

[*3.4.1. Internal Interfaces 3*](#_heading=h.ihv636)

[*3.4.2. External Interfaces 3*](#_heading=h.32hioqz)

[**4. Detailed System Design 3**](#_heading=h.1hmsyys)

[4.1. Key Entities 3](#_heading=h.41mghml)

[4.2. Detailed-Level Database Design 3](#_heading=h.2grqrue)

[*4.2.1. Data Mapping Information 3*](#_heading=h.vx1227)

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

[4.3. Archival and retention requirements 3](#_heading=h.1v1yuxt)

[4.4. Disaster and Failure Recovery 3](#_heading=h.4f1mdlm)

[4.5. Business Process workflow 3](#_heading=h.2u6wntf)

[4.6. Business Process Modeling and Management (as applicable) 3](#_heading=h.19c6y18)

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

[4.8. Variables 3](#_heading=h.28h4qwu)

[4.9. Activity / Class Diagrams (as applicable) 3](#_heading=h.nmf14n)

[4.10. Data Migration 3](#_heading=h.37m2jsg)

[*4.10.1. Architectural Representation 3*](#_heading=h.1mrcu09)

[*4.10.2. Architectural Goals and Constraints 3*](#_heading=h.46r0co2)

[*4.10.3. Logical View 3*](#_heading=h.2lwamvv)

[*4.10.4. Architecturally Significant Design Packages 3*](#_heading=h.111kx3o)

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

[*4.10.6. Deployment View 3*](#_heading=h.1egqt2p)

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

[5.1. Time Zone Support 3](#_heading=h.2dlolyb)

[5.2. Language Support 3](#_heading=h.sqyw64)

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

[5.4. Server-Side Requirements 3](#_heading=h.1rvwp1q)

[*5.4.1. Deployment Considerations 3*](#_heading=h.4bvk7pj)

[*5.4.2. Application Server Disk Space 3*](#_heading=h.2r0uhxc)

[*5.4.3. Database Server Disk Space 3*](#_heading=h.1664s55)

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

[*5.4.5. Jobs 3*](#_heading=h.25b2l0r)

[*5.4.6. Network 3*](#_heading=h.kgcv8k)

[*5.4.7. Others 3*](#_heading=h.34g0dwd)

[5.5. Configuration 3](#_heading=h.1jlao46)

[*5.5.1. Operating System 3*](#_heading=h.43ky6rz)

[*5.5.2. Database 3*](#_heading=h.2iq8gzs)

[*5.5.3. Network 3*](#_heading=h.xvir7l)

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

[**6. References 3**](#_heading=h.1x0gk37)

[**7. Appendix 3**](#_heading=h.4h042r0)

# 

# Introduction

A search engine is a system that allows users to search files, through all files in the local file system and list out all the files where a match is found. They search the files in a systematic way for particular information. Search engines are able to return results quickly-even with millions of websites by scanning the internet continuously and indexing every page they find.

When a user enters a query into a search engine, the engine scans its index of files to find those that are relevant to the user’s query.It looks at the files titles, contents and keywords it has indexed and uses algorithms(step-by-step operation) to produce a list of result. If the user inputs wrong data or there is no result of the query of the user then it will display the appropriate error message.

The information can be a mix of links to web pages,articles,research papers, and other types of files. There is no restriction related to a number of searches,so all users can spend a lot of time searching for valuable content to fulfill their requirements.

## 

## Intended Audience

| User | Searching file |
| --- | --- |

## Acronyms/Abbreviations

| LSR | Local Search Engine |
| --- | --- |

## Project Purpose

The main aim of the project is to develop LSR that helps people find the information they are looking for using keywords and phrases. It provides the user with the most relevant results. The functions of search engines are collecting information about files, categorizing those files, and creating an algorithm that makes it easy for people to find relevant files.

## Key Project Objectives

The key objectives of this project are:

* To develop a LSR that helps people to find information they are looking for.
* To develop a search engine that is able to return the results quickly.
* To develop a search engine that collects information about files,categorizing those files and creating an algorithm.
* To eliminate the need to find information manually.
* To search for queries using the file system..

## Project Scope and Limitation

Our project aims for a LSR that helps people to find the information they are looking for and we have tried to computerize various processes of search engines. This project is targeted to the small or medium organization only.

### In Scope

It may help in collecting perfect details of the searched element. In a short period the collection of data will be obvious, simple and sensible. The data files shall be loaded in the database, and all operations shall happen in the file system only. On the demand of the user, the data shall be processed.

LSR shall comprise of at least following fields:

* search file
* search keyword in files
* search file by path

### Out of scope

* In the system the person has to specify their requirements according to the system’s choices.
* In the computer system there is less risk of making errors than doing it manually.
* To utilize resources in an efficient manner.
* The system generates the type of information that can be used for various purposes.
* Easy to operate and understand by the user.
* Is expandable.

## Functional Overview

*To create this software we use 5 function that is:*

* *path()- LSE should also be able to search for a particular file, when the absolute path of the file is provided and display the contents of that file.*
* *files()-LSE should be able to search file through the file system and display the content of that file.*
* *keyword()-LSE should search for a given word/sentence/string through all the files in the local file system and list out all the files where a match was found.*
* *error()-LSE should display appropriate error messages as and when a search fails.*
* *menu()-LSE should provide an appropriate menu for searching a file.*

## Assumptions, Dependencies & Constraints

With the use of c language and Linux we can create a local search engine where the user finds the information they are looking for. The information can be searched using keywords and phrases.

## Risks

* The sites that appear through a search engine are only the index of keywords by software and they are hence not evaluated for any relevance. The software only displays these sites even though they are not relevant to your search
* One also needs to be very specific during the selection of the key words because there are so many sites on the internet. There are also so many results which can result through one indexing or search and these can prove to be very tiresome to evaluate.
* The search outcome displayed may not be arranged in a sequence and hence one may be required to scroll up to the next page to get the relevant information that he/she was searching for*.*

# Design Overview

The project is designed by dividing the program into three files i.e., Search file, Search Keyword, Search file through path.

These files contain different functions and declarations for different cases. These functions are called based on the user inputs.

In the user search file we perform search related operations. Cases such as search file,

search keyword, search file through path are in it.

## Design Objectives

* **User search**

**This file contains 3 cases which are as follows -**

**1. Search files**

**2. Search keyword in files**

**3. Search files by path**

**4. Exit**

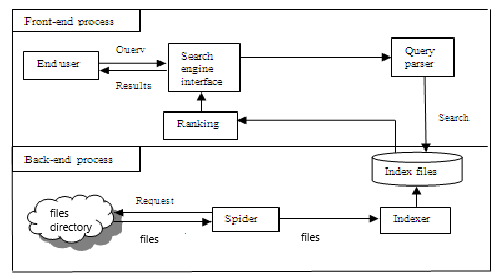
**Each case has a member function that is declared inside the block. The member**

**function is defined inside the program. Based on the user’s inputs each case calls**

**the function that it is mapped with.**

**When case 4 is chosen by the user then the program exits and a thank you message gets displayed on the screen.**

### Recommended Architecture



## Architectural Strategies

*We can follow the TOP DOWN STRATEGY in our project.*

*The top down strategy uses the modular approach to develop the design of the system. It is called so because it starts from the top or the higher-level moves towards the lowest level modules.*

*In this technique, the main module for developing the system is identified. The main module is divided into several small submodules based on the task performed by each module.*

*Like in our project firstly the user goes to the search page which is the main module then he will get three options (search file, search text file and search keyword) ,these three are sub-modules.*

*This process of dividing each module into several submodules continues till lowest level modules, which cannot be further subdivided, are not identified.*

### Design Alternative

Not Applicable

### Reuse of Existing Common Services/Utilities

In LSE is are using existing linux commands for searching the files , which are:-

* grep
* cat
* cd
* ls

### 

### Creation of New Common Services/Utilities

*In LSE we are creating a menu function for users so that users can search files according to his/her preferences.*

*In LSE we are also creating an error function so that if any searched file is not available the error message will be displayed.*

### User Interface Paradigms

*This system allows users to easily search files through all the files in the local file system and display the contents of the searched file. Just with a click, user can view all the data which is there inside the file. We have made a very simple and easy to understand interface.*

### System Interface Paradigms

*Good design creates good projects. If the system has a good interface and it satisfies user requirements, then the software can reach new heights. The LSE offers basic searching functionality. The user can search the file by word/sentence/string or by proper file name.*

### Error Detection / Exceptional Handling

*These elements are the main factors for cleanliness or quality in a code:*

*maintainability: Allows us to easily find and fix new bugs, without the fear of breaking current functionality.*

*extensibility: Allows us to easily add to our code base, implementing new or changed requirements without breaking existing functionality. Extensibility provides flexibility and enables a high level of reusability for our code base.*

*readability: Allows us to easily read the code and discover its purpose without spending too much time digging.*

### Memory Management

### For memory management we will use install valgrind.

Valgrind will analyze the program as it runs and report any errors or issues it finds, such

as memory leaks or accesses to uninitialized memory.

* Compile your program with debugging symbols: Make sure to compile your program with debugging symbols enabled. This will allow Valgrind to provide more detailed information about the errors it finds.
* Run your program through Valgrind: Use the following command to run your program through Valgrind.
* **valgrind --leak-check=full ./myprogram**

### Performance

* When the user searches the file with a path then the content of the file is displayed.
* When the user searches the file with a word,sentence or string then the search engine list out all the files where a match was found.
* When the user searches the file then the content of the file is displayed.

### Security

*A File Security System is a console application that can be used to prevent unauthorized access to a file.*

### Concurrency and Synchronization

Not Applicable

### Housekeeping and Maintenance

Not Applicable

# System Architecture

To create this LSE we use 3 case that is:

1. path()

If user choose to search file by providing the path of the file, then the particular

function get called. LSE should also be able to search for a particular file, when the

absolute path of the file is provided and displays the contents of that file.

2. files()

If user choose to search file by providing the name of the file, then the particular

function get called. LSE should be able to search file through the file system and display

the content of that file.

3. keyword()

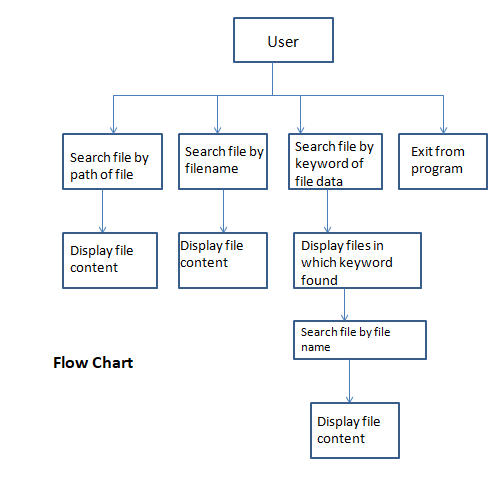
If user choose to search file by providing the keyword of the file content, then the

a particular function get called. LSE should search for a given word/sentence/string

through all the files in the local file system and list out all the files where a match

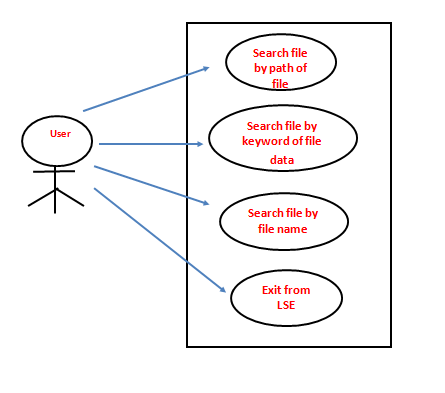
was found.

## System Architecture Diagram.



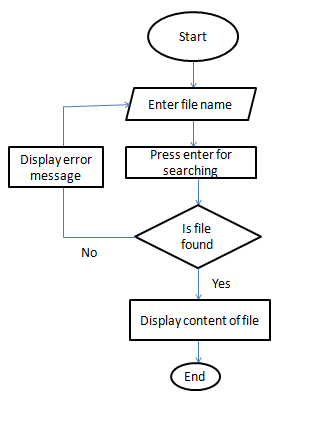
## 

## System Use-Cases

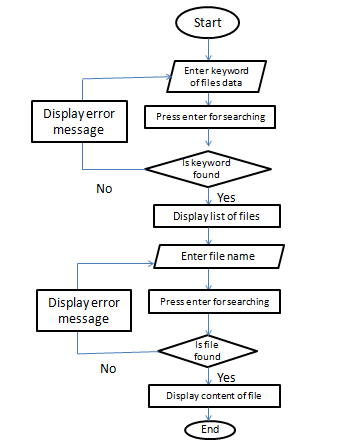
**

## Subsystem Architecture

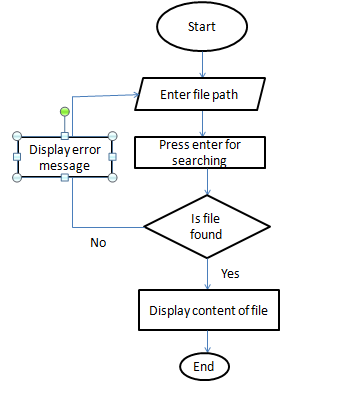
**3.3.1 Search files**

****

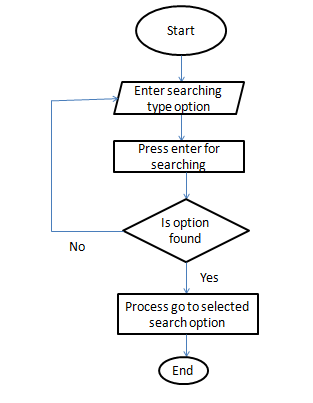
**3.3.2 Search keyword in files**

****

**3.3.3 Search files by path**

****

**3.3.4 Menu**

****

## System Interfaces

The user interface is responsible for all the interactions with the users.User

interface always effects the user mind because how easy and how functional

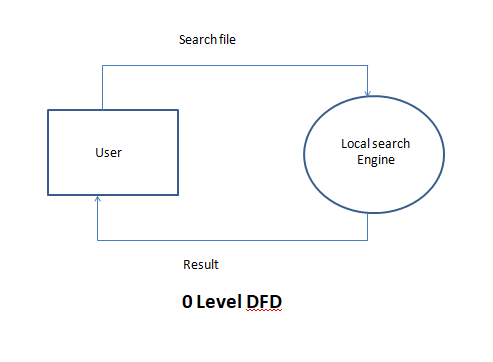
the project is depends on the user point of view.

### Internal Interfaces

As an Internal Interface we are using Ubuntu Linux distribution. It is an operating system that is made up of a collection of software based on Linux kernel or you can say distribution contains the Linux kernel and supporting libraries and software.

### External Interfaces

Input from the user will be via keyword. The user will navigate the LSE by entering input via keyword and then press enter. Then LSE will give appropriate responses to the given input.

**

# Detailed System Design

A LSE is a system that allows users to search files through all the files in the local file

system and list out all the files where a match was found.

It is the best search engine that helps user to find out the files which user wants and

display the contents of the file when the user chooses a particular file from the listing

of the files provided.

The project is designed by dividing the program into three functions:-

**1. Search files**

If user wants to search file by filename, user needs to choose the option search file.

Under this user needs to providing the name of the file, then the file function get

called. LSE should be able to search file through the file system and display the

content of that file.

**2. Search keyword in files**

If user didn’t remember the file name and wants to search by words,sentence,string,

user needs to choose the option search keyword. Under this user needs to providing

the keyword, then the keyword function get called. LSE should search for a given

word/sentence/string through all the files in the local file system and list out all the

files where a was match was found.

After that user needs to provide the file name from the list of file and LSE will display

the content of the file.

**3. Search files by path**

If user wants to search file by providing the path of the file, user needs to choose the

option search file by path.

Under this user needs to providing the path of the file, then the path function get

called. LSE should be able to search file through the file system and display the

content of that file.

## Key Entities

**file()**

Entity used in this function are:

Ø file\_name

**keyword()**

Entity used in this function are:

Ø keyword

Ø file\_name

**path()**

Entity used in this function are:

Ø path

## Detailed-Level Database Design

## Not Applicable

### Data Mapping Information

Not Applicable

### Data Conversion

Not Applicable

## Archival and retention requirements

Not Applicable

## Disaster and Failure Recovery

Not Applicable

## Business Process workflow

## Not Applicable

## Business Process Modeling and Management (as applicable)

Not Applicable

## Business Logic

Not Applicable

## Variables

Not Applicable

## Activity / Class Diagrams (as applicable)

Not Applicable

## Data Migration

Not Applicable

### Architectural Representation

Not Applicable

### Architectural Goals and Constraints

Not Applicable

### Logical View

Not Applicable

### Architecturally Significant Design Packages

Not Applicable

### Data model

Legacy system data model

Proposed system data model

Interface data model

### Deployment View

Not Applicable

# Environment Description

The complete details of the System Environment we can provide here

## Time Zone Support

The number of seconds of time difference between the local time zone and Coordinated Universal Time[UTC].

## Language Support

C language is used in this project. It was created in the 1970s by Dennis Ritchie, and remains very widely used and influential.

## User Desktop Requirements

Windows: 7 or above

Processor:Minimum 1GHz and more

Hard Drive:32GB and more

Memory(RAM):Minimum 1GB and more.

## Server-Side Requirements

Not Applicable

### Deployment Considerations

Not Applicable

### Application Server Disk Space

### Not Applicable

### Database Server Disk Space

Not Applicable

### Integration Requirements

Not Applicable

### Jobs

Not Applicable

### Network

Not Applicable

### Others

Not Applicable

## Configuration

For making LSE we need windows version above 7 and Ubuntu Linux Distribution

in our system.

### Operating System

Windows: 7 or above

Processor:Minimum 1GHz and more

Hard Drive:32GB and more

Memory(RAM):Minimum 1GB and more.

### Database

Not Applicable

### Network

Not Applicable

### Desktop

Ubuntu Linux

# References

[Build yourself a Mini Search Engine (toronto.edu)](https://www.cs.toronto.edu/~muuo/blog/build-yourself-a-mini-search-engine/)

[Creating Your Own Search Engine for C/C++ Code Samples | CodeGuru](https://www.codeguru.com/cplusplus/creating-your-own-search-engine-for-c-c-code-samples/)

[Making a Search Engine - CodeProject](https://www.codeproject.com/articles/563869/making-a-search-engine)

# Appendix

Linux commands

C functions

**Change Log**

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