ONLINE MOVIE TICKET BOOKING

SYSTEM

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** | |
|  | HLD\_LLD DESIGN DOCUMENT-Version 0.1 | 1.Chinta Jayasree |  | | | | Prasanth | |
|  | HLD\_LLD DESIGN DOCUMENT-Version 0.2 | 2.Potnuru Ramya |  | | | | Prasanth | |
|  | HLD\_LLD DESIGN DOCUMENT-Version 0.3 | 3.Sasapu Chandrika kumari |  | | | | Prasanth | |
|  | HLD\_LLD DESIGN DOCUMENT-Version 0.4 | 4.Palla Mounika |  | | | | Prasanth | |
|  | HLD\_LLD DESIGN DOCUMENT-Version 0.5 | 5. Gokake Rohini |  | | | | Prasanth | |

[1. Introduction ……………………………………………………………………………………...4](#_Toc124797212)

[1.1 Intended Audience ………………………………………………………………………..5](#_Toc124797213)

[1.2 Acronyms/Abbreviations ………………………………………………………………………..5](#_Toc124797214)

[1.3 Project Purpose ………………………………………………………………………..5](#_Toc124797215)

[1.4 Key Project Objectives ………………………………………………………………………..5](#_Toc124797217)

1.5 Project scope and limitations ……………………………………………………………………..5

[1.5.1 In Scope ……………………………………………………………………………………...6](#_Toc124797218)

[1.6 Functional Overview ……………………………………………………………………………...6](#_Toc124797219)

[1.7 Assumptions, Dependencies & Constraints ……………………………………………………..6](#_Toc124797220)

[1.8 Risks ………………………………………………………………………………………………6](#_Toc124797221)

[2. Design Overview …………………………………………………………………………………….7](#_Toc124797222)

2.1 Design Objectives ……………………………………………………………………………….8

[2.2.1 Design Alternativ …………………………………………………………………………...8](#_Toc124797224)

[2.2.2 Reuse of Existing Common Services/Utilities ……………………………………………….8](#_Toc124797225)

[2.2.3 Creation of New Common Services/Utilities ………………………………………………...8](#_Toc124797226)

[2.2.4 User Interface Paradigms ……………………………………………………………………..8](#_Toc124797227)

[2.2.5 Housekeeping and Maintenance ……………………………………………………………...8](#_Toc124797228)

[2.2.6 System Interface Paradigms …………………………………………………………………..8](#_Toc124797229)

[2.2.7 Error Detection / Exceptional Handling …………………………………………………….9](#_Toc124797230)

[2.2.8 Memory Management ………………………………………………………………………..9](#_Toc124797231)

[2.2.9 Performance ………………………………………………………………………………….9](#_Toc124797232)

[2.2.10 Security …………………………………………………………………………………….9](#_Toc124797233)

[2.2.11 Concurrency and Synchronizatio …………………………………………………………..9](#_Toc124797234)

[3.1 System Architecture Diagram ………………………………………………………………….10](#_Toc124797235)

[3.2 System Use-Cases ……………………………………………………………………………….11](#_Toc124797236)

[3.3 Subsystem Architecture ………………………………………………….…………………….11](#_Toc124797237)

[3.4 System Interfaces ……………………………………………………………………………...11](#_Toc124797238)

[3.4.1 Internal Interfaces …………………………………………………………………………11](#_Toc124797239)

[3.4.2 External Interfaces ………………………………………………………………………..11](#_Toc124797240)

[4.Detailed System Design ……………………………………………………………………………..12](#_Toc124797241)

[4.1 Key Entities ……………………………………………………………………………………12](#_Toc124797242)

[4.2 Detailed-Level Database Design ……………………………………………………………..12](#_Toc124797243)

[4.2.1 Data Mapping Information ………………………………………………………………..12](#_Toc124797244)

[4.2.2 Data Conversion …………………………………………………………………………...12](#_Toc124797245)

[4.3 Archival and retention requirements …………………………………………………………..12](#_Toc124797246)

[4.4 Disaster and Failure Recovery ……………………………………………………………..12](#_Toc124797247)

[4.5 Business Process workflow …………………………………………………………………..12](#_Toc124797248)

[4.6 Business Process Modeling and Management (as applicable) ……………………………….13](#_Toc124797249)

[4.7 Business Logic ……………………………………………………………………………...…13](#_Toc124797250)

[4.8 Variables ……………………………………………………………………………………….13](#_Toc124797251)

[4.9 Activity / Class Diagrams (as applicable) ……………………………………………………13](#_Toc124797252)

[4.10.1 Architectural Representation ………………………………………………………………23](#_Toc124797253)

[4.10.2 Architectural Goals and Constraints …………………………………………………….23](#_Toc124797254)

[4.10.3 Logical View ……………………………………………………………………………...23](#_Toc124797255)

[4.10.5 Data model ……………………………………………………………………………….23](#_Toc124797256)

4.10.6[Legacy system data mode Proposed system data model Interface data model ……….…..23](#_Toc124797257)

[5. Environment Description …………………………………………………………………………...23](#_Toc124797258)

[5.1 Time Zone Support ……………………………………………………………………………..24](#_Toc124797259)

[5.2 Language Support ……………………………………………………………………………...24](#_Toc124797260)

[5.3 User Desktop Requirements …………………………………………………………………….24](#_Toc124797261)

[5.4 Server-Side Requirements ……………………………………………………………………24](#_Toc124797262)

5. 4.1 Deployment Considerations ……………………………………………………………………...…….24

[5.4.2 Application Server Disk Space ………………………………………………………..…..24](#_Toc124797263)

[5.4.3 Database Server Disk Spacec ……………………………………………………………..24](#_Toc124797264)

[5.4.4 Integration Requirements ………………………………………………………………….24](#_Toc124797265)

[5.4.5 Jobs …………………………………………………………………………………………24](#_Toc124797266)

[5.4.6 Network ……………………………………………………………………………………24](#_Toc124797267)

[5.4.7 Others ……………………………………………………………………………………...25](#_Toc124797268)

[5.5 Configuration ………………………………………………………………………………..25](#_Toc124797269)

[5.5.1 Operating System …………………………………………………………………………..25](#_Toc124797270)

[5.5.2 Database ………………………………………………………………………………….25](#_Toc124797271)

[5.5.3 Network …………………………………………………………………………………….25](#_Toc124797272)

[5.5.4 Desktop …………………………………………………………………………………..25](#_Toc124797273)

[References ……………………………………………………………………………………………..25](#_Toc124797274)

# 

# Introduction

Online movie ticket booking system is basically made for providing the customers an anytime and anywhere service for booking cinema tickets and providing information about the movies and their schedule online.

* Admin can use Online Movie Ticket Booking System Project to insert and delete data such as movie description, movie schedule which will update the related webpage and will be accessible by the customers.
* Online Movie Ticket Booking System provide another way for the customers to buy cinema ticket. This system reduces workload on customers, it is an automatic ticket booking system.
* This system is basically aimed to provide complete information of the movie and schedule to the customer, according to which he can book the tickets.

## Intended Audience

This document is for demo purpose this is created for educational purpose.

## 1.2 Acronyms/Abbreviations

|  |  |
| --- | --- |
| MTBS | Movie Ticket Booking System |
| UML | Unified Modeling Language |

## Project Purpose

## The purpose of the project to build an application program to reduce to manual work for managing the Seats, Booking, Movie, Customer. It tracks all the details about the Customer, Payment, Shows.

## 1.4 Key Project Objectives

The main purpose of Online Movie Ticket Booking system project is to provide an automated system of buying movie ticket. Now customer can get to know movie show timing and buy tickets online via internet 24×7. Admin has full control over all modules of this application. Admin can

**1.5 Project Scope and Limitation**

SCOPE:

* This is a web application for buying movie tickets online.
* This system is developed keeping in view of the current multiplex working pattern. Schedule for many screens can be programmed in this application.
* Customers can see a graphical view of the seat availability and choose their desired seat.
* They can pay ticket amount online via credit card etc.

LIMITATION:

* You need internet access.
* You need to be ready for an influx of new customers.
* Not all online booking systems are created equal.
* Avoid booking systems that don’t bring you knew quality customers.

### 1.5.1 In Scope

* The scope of this project very broad in terms of gaining and sharing knowledge among worlds.
* Few of them are:
* Can be used anywhere any time as it is a web-based application.
* This application will be used in educational institutions.

## Functional Overview

An online ticket booking platform facilities selection, reservation and purchase of tickets for the movies. This type of platform primarily contains:

* Information such as show movie schedule, and price of the tickets.
* Check and post reviews and ratings of the movies.

## 1.7 Assumptions, Dependencies & Constraints

Let us assume that this is a distributed PVR database system and it is used in the following application:

* A request for booking/cancellation of tickets from any source to any theatre or show
* Calculation fares and calculating appropriate.
* Reward points for these who booked the tickets.

## 1.8 Risks

The traditional way of booking the ticket for the movie is the customer need to go to the specific theatre where the desired movie was playing and need to stand in queue and buy the ticket for the movie this will become more difficult for a person in order to overcome this problem The project gives real life understanding of online movie ticket booking system and activities performed by various roles in the supply chain. Here we provide automation for movie ticket booking system through internet. Online movie ticket booking system project captures activities performed by different roles in real life ticket booking which provides enhanced techniques for maintaining the required information up to date, which results in efficiency. The project gives real life understanding of online movie ticket booking system and activities performed by various roles in the supply chain.

# 2. Design Overview

1.Start

This is the start block which indicates the start of the program.  
 It will allows to enter the main menu .

2.Main Menu

It consists of Movie booking, Ticket Cancellation, Membership, Admin, exit.

3.Movie Booking:

This is the module used for the booking the movies from the available list of the movies.

After selecting the movie, it shows select timings of the movie and select number of seats

Select row and column and choose payment option.

4.Ticket cancellation:

In this module the ticket will be cancelled for the selected movie with ticket number.

5.Membership/user login:

If the member is new user has to enter name, email-id, phone number, account number,

and to create password. If the enter details are valid the new user become a member.

In this module the member can renew the membership by entering account number and password, if he is existing user and extend 1,2,3 years.

6.Admin:

In this module admin can login with validate user name and password.

Admin can able to modify the list of movies by adding or deleting the movies and can update the movies by adding show timings and movie name.

7.Exit:

This ensures that the program has terminated.

## 2.1 Design Objectives

The Movie ticket booking system follow a series of steps in order to complete the design system.

1. Requirement Gathering

2. Creation of Use-Case Diagram

3. Create Class Diagrams

4. Coding the System

5. Future Scopes

### 2.2.1 Design Alternative

NA

### 2.2.2 Reuse of Existing Common Services/Utilities

#include<iostream>

#include<string.h>

|  |
| --- |
|  |
|  |
|  | #include <fstream> |
|  | #include <iomanip> |
|  | #include <ctime> |
|  |  |
|  | #include <vector> |
|  | #include <algorithm> |
|  |  |
|  |  |

### 2.2.3 Creation of New Common Services/Utilities

NA

### 2.2.4 User Interface Paradigms

* Linux/Ubuntu machine
* g++ complier

### 2.2.5 Housekeeping and Maintenance

NA

### 2.2.6 System Interface Paradigms

By using file handling, we are detecting the errors and handle it using the conditional statements. We have used Valgrind for error detection and exceptional handling.

### 2.2.7 Error Detection / Exceptional Handling

* By using file handling, we are detecting the errors and handle it using the conditional statements. We have used Valgrind for error detection and exceptional handling.

### 2.2.8 Memory Management

NA

### 2.2.9 Performance

To make the application run faster and smoother the code has to be written in optimized manner. The performance is based on the configuration of the system

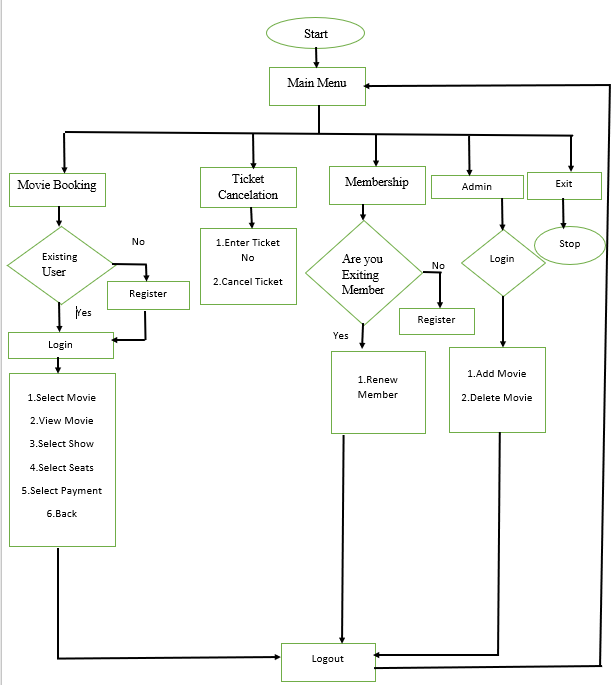
### 2.2.10Security

* All data in the Customer Database should be backed up in case the of data loss.
* Admin must use secure connection while logging in.
* Admin must not disclose his password to anybody.

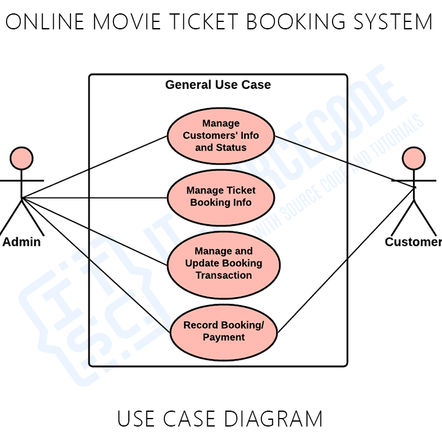
# 2.2.11 Concurrency and Synchronization

NA

## 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

# 4.Detailed System Design

The code starts by home page. For designing Home page we used switch cases. After that if customer selects the Movie Booking then Movie Booking functions will open. In Movie Booking Guest login(),Select Seats(),Compare Names(), Show ticket(),Member login(),Show Movie List(),Time Checking(),Payment() these functions are available. All Movie Booking process is done in this module. If customer selects a Membership module then user can do two functions one is Member Registration() and Renew Membership().

In this system there is Admin module also present. In that only Add movie() and Delete Movie() Functions are available. In this system if customer wants to cancel tickets then there is separate module is available. In that customer can cancel booked tickets. If customer wants exit from system there is exit module to exit from system

# 4.1 Key Entities

* Validation

## 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

## 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: movie booking

Start:

class Movie{

declare string name,day,time1,time2,time3;

};

class Customer{

protected:

declare string name, email, phone\_number;

public:

function of setname():

print enter name

function of setemail():

print enter email

chack if (email has @ or not)

print valid

else

invalid repeat set email

function of setphone\_number():

print enter phone number

check for validation if(phonenumber is greater than 10)then

print valid

else

invalid repeat process

getname()

getemail()

getphone\_number()

};

class Member : public Customer {

protected:

declaration of string icno, acc\_num;

char password[50];

int valid\_month, valid\_year;

get the current year

void MemberInitialize(string name,string icno,string email, string phone\_number, string acc\_num, string password, int valid\_month, int valid\_year)

{

read all above data

}

void setIC(){

print enter ic number without -

if(ic is greater than 10)

print valid

create file membership and read all data i.e name,icnum,eamil,phone\_number,acc num,month year

if(enterded ic number equal to ic number which is read from membership.txt)

then icfound

else

icnot found

else

print valid IC number

repeat setIc

}

void setAccNum(){

assignn random Account number

}

void setMonth(){

assigne current month using ctime headet take pointer assign time to it

}

void setPassword(){

print create password

print reenter your passowrd

if(created password is equal to reenter password)

print succesfully pasword created

else

print create password

}

void setYear(){

set year using ctime header file

}

int getYear(){

get year

}

void getPassword(){

return password}

string getAccNum(){retrun account number}

string getIC(){return Ic}

int getMonth(){return month;}

string getPassword(){retrun password}

};

Member \*MemberProg = new Member;

using MOVIE = Movie[MAXMOVIES];

void title();

bool CompareNames(Movie movie1, Movie movie2);

int ReadFile(MOVIE& movies, int& movieCount);

void showmovielist(MOVIE& movies, const int movieCount);

void time\_checking(int time[], int i);

void showTicket();

void payment(float pay);

void movie\_booking();

void ticket\_cancel();

void member\_reg();

void member\_renew();

void membership();

void add\_movie();

void delete\_movie();

void update\_movie();

void management();

string member\_login();

in the main function{

main menu(){

call title page to set title page

(A) MOVIE BOOKING

(B) TICKET CANCELLATIONG

(C) MEMBERSHIP

(D) EMPLOYEES

(E) QUIT

select from above option

if it is A then call movie booking function

else if B then call membership function

else if C call membership Function

else if D call admin/employee function

else quit

print do want want mamin menu enter b

}

void title {

set title page by running two loops

}

bool CompareNames(Movie movie1, Movie movie2)

{ true

if(movie 1 name equal to movie 2 name )

false

}

int ReadFile(MOVIE& movies, int& movieCount)

{

read from file movies name and other details lo=ike timings and day available

count increment

}

void showmovielist(MOVIE& movies, const int movieCount)

{

print s.no avaialble movies ,day,timings

write movies availble to moviess.txt using streams

}

void time\_checking(int time[], int i)

{

print enter timings

while(time is lessthan 0 and time greater than 2400)

print enter valid time

read from user time

}

print successful

}

void showTicket(){

assign random ticket number

print this is your ticket number

print thank you

}

void payment(float pay)

{

print (A) CASH

(B) DEBIT CARD

(C) CREDIT CARD

print enter your payment method

if A

print You have selected to pay on cash

print The amount you need to pay is : pay

else if B

print You have selected to pay on Debit card

print enter cvv number last 3 digits

print The amount you need to pay is : pay

else

else if B

print You have selected to pay on Credit card

print enter cvv number last 3 digits

print The amount you need to pay is : pay

print predd b go to main menu

read from user

}

void member\_reg(){

setName();

setPhonenumber();

setIC();

setEmail();

setMonth();

setYear();

print do you want change data or process

if yes

repeat above set statement

else

set account number and password

print you are member now

}

void member\_renew(){

call member login to login

print how many years you e=want extend your account

print 1year

print 2 year

print 3 year

read from user

if 1 year

paymnet(float 100.00);

else if(2 years)

payment(float 200.00)

else

payment(float 300.00)

open file membership.txt file update extended year

print you have succefully extend user selected year

}

void membership()

{

print A new User

Print B membership

read input from user

if A

call member\_reg();

else

call member\_renew();

}

void movie\_booking()

{

print A member login

print B Guest login

read input from user

if A

call member login

else

call set name,set email,set phonen\_number

call showlist

print please selct movie from available movie

read from user

print which show you want select(show 1 ,show 2 and show 3)

read from user

if user input is greater than 3

print wrong selection please select right one repeat above two steps

else

print you have selcted (user)timings

print selct number of seats

read from user

if seat number is greater than 10

print wrong choice selct between 1 and 10

repeat above two steps

else

do{

print enter row u want [A,B,C,D]

read from user input

if user input is not between A and E

print

plaese select between A and E

print enter coloumn number

read from user

if user input is greater than 5

print please select from between 1 to 5

for (int varaible =0; varaiabel <user input for row and coloun ;variable++)

if (seat\_row[variable] == seat\_row[userinout for row] && seat\_column[varible] == seat\_column[userinput for coloum]) {

then print seat is not vailable please selcet another input

read from user

repeat selection of seats

while seat is less than no of selats

}

print here are two types of seats available:

(A) DELUXE (RM 200)

(B) NORMAL (RM 100)

print select from above

read from user input

if user input is Deluxe then

payment(200.00)

else

payment(100.00)

open a booikng.text file and store the booing recording

print do you wish to continue then press p and b back and r reselect

if user input p

receipt

elseif user input r

goto seat selection

else

back to main menu

}

void management()

{

print enter name and password

if correct then

print please select following options

A add movie

B delete movie

C change timings

print senter your response

else

if login attempt is 3

login failed

read input response from user

if input os A

call addmovies

else if input is B

call delete movies

else

call change showtimings

}

void addmovies()

{

enter movies name

input from user input

add timings for 3 showtimings by calling timechecking

open movie file and uodate the movies

}

void delete\_movie(){

read movies.txt file

print enter number of movie you e=wamt to delete feom available movies

take user input

copy the data into temp.txt then delete movie

print the deletion of movie is done

close two files i.e movies.tex and temp.txt

}

void update\_movie(){

print select movies you want update timings from availble movies

read from user input

open movies .text file read movie detials

add first show timing and check timecheck condition

add second show timing and check timecheck condition

add third show timing and check timecheck condition

update movie.text file by opening and

update in temp.txt

}

void ticket\_cancel(){

print enter your ticket number

read data from user

open booking.txt file

check whether entered user tickekt number present in booking.txt

if present

then show all detilas of his booking sytem by reading from a file

else

store all details from temp.tex

if yes

cancellation done

else

not done main menu

**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

### Legacy system data mode Proposed system data model Interface data model

**4.10.6 Deployment View**

NA

# 5. 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.

* Movie Ticket Booking Programming: This system mainly for movie ticket booking. In those four modules are there Ticket booking, Ticket cancelation, Membership and Admin. The system will work on the user’s terminal. The performance shall depend upon hardware components of the Movie Ticket Booking Application and the internet connection. The responsiveness of the application shall be high, and the application shall behave as per the user action.
* 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

It will support time zones as per Indian standard time(IST) in (GMT +5:30) and UST standard

## 5.2 Language Support

English

## 5.3 User Desktop Requirements

Linux, Ubuntu

## 5.4 Server-Side Requirements

Linux, Ubuntu

**5.4.1 Deployment Considerations**

* Local storage is used
* No network latency to consider
* To scale buy a bigger CPU, more memory, larger hard drive, or additional hardware

### 5.4.2 Application Server Disk Space

Disk Space - Less space is required. Local Operating System is required and three txt files to store the records of processes.

# 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

**5.5.1**. Operating System: - Linux environment

### 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

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://[www.programiz.com/dsa/linked-list](http://www.programiz.com/dsa/linked-list)
* ://[www.javatpoint.com/file-handling-in-c](http://www.javatpoint.com/file-handling-in-c)
* Git hub