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** | |
| 07/01/2023 | HLD\_LLD DESIGN DOCUMENT-Version 0.1 | 1.Chinta Jayasree | Functional requirements that are required for different modules. | | | | Prasanth | |
| 07/01/2023 | HLD\_LLD DESIGN DOCUMENT-Version 0.2 | 2.Potnuru Ramya | Design the System Architecture(Flow chart, Usecase Diagram) | | | | Prasanth | |
| 14/01/2023 | HLD\_LLD DESIGN DOCUMENT-Version 0.3 | 3.Rohini Gokake | Design the Sequence diagram | | | | Prasanth | |
| 14/01/2023 | HLD\_LLD DESIGN DOCUMENT-Version 0.4 | 4.Palla Mounika | Made changes in Environment Description | | | | Prasanth | |
| 17/01/2023 | HLD\_LLD DESIGN DOCUMENT-Version 0.5 | 5. Sasapu Chandrika kumari | Changes in Pseudo code | | | | Prasanth | |

[1. Introduction 5](#_Toc124611532)

1.1Intended Audience……………………………………………………………………………………..5

[1.2 Acronyms/Abbreviations ………………………………………………………………………...5](#_Toc124611534)

1.3Project Purpose…………………………………………………………………………………………5

[1.4 Key Project Objectives ………………………………………………………………………...5](#_Toc124611536)

[1.5 In Scope …………………………………………………………………………..6](#_Toc124611537)

[1.6 Functional Overview ………………………………………………………………………….6](#_Toc124611538)

[1.7 Assumptions, Dependencies & Constraints ………………………………………………………6](#_Toc124611539)

[1.8 Risks ..…………………………………………………………………………6](#_Toc124611540)

[2. Design Overview ………………………………………………………………………… 7](#_Toc124611541)

[2.2.1 Design Alternative 8](#_Toc124611544)

[2.2.2 Reuse of Existing Common Services/Utilities 8](#_Toc124611545)

[2.2.3 Creation of New Common Services/Utilities 9](#_Toc124611546)

[2.2.4 User Interface Paradigms 9](#_Toc124611547)

[2.2.5 Housekeeping and Maintenance 9](#_Toc124611548)

[2.2.6 System Interface Paradigms 9](#_Toc124611549)

[2.2.7 Error Detection / Exceptional Handling 9](#_Toc124611550)

[2.2.8 Memory Management 9](#_Toc124611551)

[2.2.9 Performance 9](#_Toc124611552)

[2.2.10 Security 9](#_Toc124611553)

[2.2.11 Concurrency and Synchronization 9](#_Toc124611554)

[3. System Architecture 9](#_Toc124611555)

[System Architecture Diagram ………………………………………………………………………...10](#_Toc124611556)

[3.2 System Use-Cases ………………………………………………………………………………….11](#_Toc124611557)

[3.3 Subsystem Architecture ………………………………………………………………………...12](#_Toc124611558)

[3.4 System Interfaces ………………………………………………………………………………….13](#_Toc124611559)

[3.4.1 Internal Interfaces 13](#_Toc124611560)

[3.4.2 External Interfaces 13](#_Toc124611561)

[4. Detailed System Design 13](#_Toc124611562)

[4.1 Key Entities ………………………………………………………………………………….13](#_Toc124611563)

[4.2 Detailed-Level Database Design ………………………………………………………………13](#_Toc124611564)

[4.2.1 Data Mapping Information 13](#_Toc124611565)

[4.2.2 Data Conversion 13](#_Toc124611566)

[4.3 Archival and retention requirements ………………………………………………………………13](#_Toc124611567)

[4.4 Disaster and Failure Recovery ………………………………………………………………14](#_Toc124611568)

[4.5 Business Process workflow ………………………………………………………………………...14](#_Toc124611569)

[4.6 Business Process Modeling and Management (as applicable) ………………………………….14](#_Toc124611570)

[4.7 Business Logic ………………………………………………………………………………….14](#_Toc124611571)

[4.8 Variables ……………………………………………………………………………………………14](#_Toc124611572)

[4.9 Pseudo code…………………………………………………………………………………………15](#_Toc124611573)

[4.10 Data Migration ………………………………………………………………………………….22](#_Toc124611574)

[4.10.1 Architectural Representation 22](#_Toc124611575)

[4.10.2 Architectural Goals and Constraints 22](#_Toc124611576)

[4.10.3 Logical View 22](#_Toc124611577)

[4.10.4 Architecturally Significant Design Packages 22](#_Toc124611578)

[4.10.5 Data model 22](#_Toc124611579)

[4.10.6 Deployment View 23](#_Toc124611580)

[5. Environment Description 23](#_Toc124611581)

[5.1 Time Zone Support …………………………………………………………………………………23](#_Toc124611582)

[5.3 User Desktop Requirements ……………………………………………………………………..…](#_Toc124611584)23

[5.4 Server-Side Requirements](#_Toc124611585) ………………………………………………………………………..23

[5.4.1 Deployment Considerations 23](#_Toc124611586)

[5.4.2 Application Server Disk Space 23](#_Toc124611587)

[5.4.3 Database Server Disk Space 23](#_Toc124611588)

[5.4.4 Integration Requirements 24](#_Toc124611589)

[5.4.5 Jobs 24](#_Toc124611590)

[5.4.6 Network 24](#_Toc124611591)

[5.4.7 Others 24](#_Toc124611592)

[5.5 Configuration ………………………………………………………………………………](#_Toc124611593)….24

[5.5.1 Operating System 24](#_Toc124611594)

[5.5.2 Database 24](#_Toc124611595)

[5.5.3 Network 24](#_Toc124611596)

[5.5.4 Desktop 24](#_Toc124611597)

[6. References 25](#_Toc124611598)

# 

# 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.

**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.

## 1.6 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 username 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 Sequence Diagram

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, and vectors storing the data of movies and customers and members. Vectors are used in our application dynamically store the data at runtime and have random access to retrieve the values.

### 

### 2.2.7 Error Detection / Exceptional Handling

* By using Exception 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.

### 2.2.10 Security

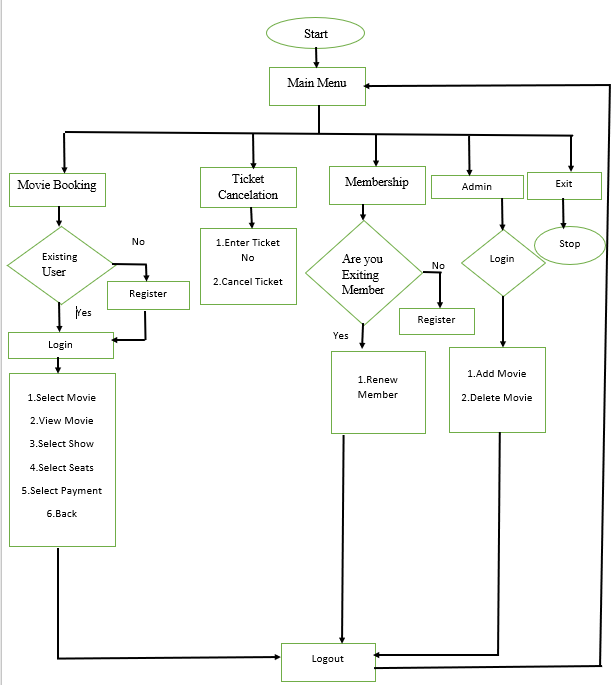
* All data in the Customer Database should be backed up in case the of data loss.
* Admin must use secure connection while logging in.
* Admin must not disclose his password to anybody.

# 2.2.11 Concurrency and Synchronization

NA

## System Architecture Diagram

**FLOW CHART**



## 3.2 USE CASE DIAGRAM

Diagram

Description automatically generated

**SEQUENCE DIAGRAM**

**Graphical user interface

Description automatically generated**

# 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

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

There are variables for member are name, email, phone number, Account number, IC number,

## 4.9 Pseudocode: Movie Booking

Start:

Create a class Movie

class Movie

Declare variables string name, day, time1, time2, time3

Create class Customer

class Customer

Declare variables string name, email, phone\_ number

function of setname()

print enter name

function of setemail()

print enter email

function of setphone\_number ()

print enter phone number

getname()

getemail()

getphone\_number()

Create a class Member inherited from class Customer

class Member: public Customer

Declaration of variables icno, acc\_num, password, valid\_month, valid\_year

void MemberInitialize()

read data of name, icno, email, phone\_number, acc\_num, password, month, year

void setIC()

print enter ic number without -

void setAccNum()

assign random Account number

void setMonth()

assign current month using ctime header take pointer assign time to it

void setPassword()

print create password

void setYear()

set year using ctime header file

int getYear()

get year

string getAccNum()

return account number

string getIC()

return Ic

int getMonth()

return month

string getPassword()

return password

//movies maximum count

using MOVIE = Movie[MAXMOVIES]

//read movies from movies file

int ReadFile()

// display available movies from file

void showmovielist()

//checking whether time is correct or not

void time\_checking()

//assign ticket number

void showTicket()

//payment function

void payment(float pay)

//booking a movie

void movie\_booking()

// function for cancelling movie

void ticket\_cancel()

//member registration

void member\_reg()

//Function to extend years

void member\_renew()

//user membership

void membership()

void management()

// user/member login checking

string member\_login()

Main function

void menu ()

call title page to set title page

(A) MOVIE BOOKING

(B) TICKET CANCELLATION

(C) MEMBERSHIP

(D) ADMIN

(E) QUIT

select from above option

if it is A

then call movie booking function

else if B

then call Ticket cancellation function

else if C

call membership Function

else if D

call admin/employee function

else

quit

// reading movies list from file

int ReadFile(MOVIE& movies, int& movieCount)

read from file movies name and other details like timings and day available

count increment

// show movie list from file

void showmovielist(MOVIE& movies, const int movieCount)

print s.no available movies, day, timings

write movies available to movies.txt using streams

//check for valid time

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

print enter timings

//show the ticket number

void showTicket()

assign random ticket number and print this is your ticket number

print thank you

//payment

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 if C

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

read from user

//Member registration

void member\_reg()

setName()

setPhonenumber()

setIC()

setEmail()

setMonth()

setYear()

print you are member now

//void Member Renew

void member\_renew()

call member to login

print how many years you want extend your account 1 year, 2year, 3year

read from user

if 1 year

payment (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 successfully extend user selected year

//membership function

void membership()

print A new User

Print B membership

read input from user

if A

call member\_reg ()

else

call member\_renew ()

//function for movie booking

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 phone\_number

call showlist

print please select movie from available movie

read from user

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

read from user

print select number of seats

read from user

if seat number is greater than 10

print wrong choice select between 1 and 10

repeat above two steps

else

do

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

read from user input

if user input is not between A and E

print please select between A and E

print enter column number

read from user

if user input is greater than 5

print please select from between 1 to 5

for (int variable =0; variable <user input for row and column; variable++)

if (seat\_row[variable] == seat\_row[user input for row] && seat\_column[variable]== seat\_column[user input for column])

then print seat is not available please select another input

read from user

repeat selection of seats

while seat is less than no of seats

print here are two types of seats available:

(A) DELUXE (RM 200)

(B) NORMAL (RM 100)

print select from above

read from user input

open a booking.txt 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

go to 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

print enter your response

if input is A

call add movies

else

call delete movies

//Add movies by admin

void addmovies()

enter movies name

input from user input add timings for 3 show timings by calling time checking

open movie file and add the movies

//Delete movies by admin

void delete\_movie()

read movies.txt file

print enter number of movies you want to delete from 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.txt and temp.txt

//ticket cancel by customer

void ticket\_cancel ()

print enter your ticket number

read data from user

open booking.txt file

check whether entered user ticket number present in booking.txt

if present

then show all details of his booking system by reading from a file

else

store all details from temp.txt

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 model49

**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 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 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.

**6. References**

* [**https://www.geeksforgeeks.org/vector-in-cpp-stl/**](https://www.geeksforgeeks.org/vector-in-cpp-stl/)
* [**https://www.javatpoint.com/cpp-templates**](https://www.javatpoint.com/cpp-templates)
* **https://www.geeksforgeeks.org/implementation-of-singleton-class-in-cpp/**