**Bus Reservation 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  V0.1 | 1.Gorantla Saranya |  | | | | | Prasanth | |
|  | HLD\_LLD  Design Document  V0.2 | 2.Potta Lakshmi Ravalli |  | | | | | Prasanth | |
|  | HLD\_LLD  Design Document  V0.3 | 3.Om Sai Amrutha  Varshini Alla |  | | | | | Prasanth | |
|  | HLD\_LLD  Design Document  V0.4 | 4.Vudumula Tejaswi |  | | | | | Prasanth | |
|  | HLD\_LLD  Design Document  V0.5 | 5.Vanapalli Bhargavi |  | | | | | Prasanth | |
|  | HLD\_LLD  Design Document  V0.6 | 6. Jyothi Swarupa Repalle |  | | | | Prasanth | | |

[1. 1 Introduction……………………………………………………………………………….5](#_Toc122382953)

[1.1 Intended Audience 5](#_Toc122382954)

[1.2 Acronyms/Abbreviations 5](#_Toc122382955)

[1.3 Project Purpose 6](#_Toc122382956)

[1.4 Key Project Objective 6](#_Toc122382957)

1.5 Project Scope and Limitations…………………………………………………………………………………………………….6

1.5.1 In Scope……………………………………………………………………………………………………………………………….6

[1.6 Functional Overview 7](#_Toc122382966)

[1.7 Assumptions, Dependencies & Constraints 7](#_Toc122382967)

[1.8 Risks 7](#_Toc122382968)

[2 Design Overview 8](#_Toc122382969)

[The system consists of three entities: admin , coordinator and client. 8](#_Toc122382970)

[2.1 Design Objectives 8](#_Toc122382971)

[2.1.1 Recommended Architecture 9](#_Toc122382972)

[2.2. Architectural Strategies 9](#_Toc122382973)

[2.2.1 Reuse of Existing Common Services/Utilities 9](#_Toc122382975)

[2.2.2 User Interface Paradigms 9](#_Toc122382977)

[2.2.3 System Interface Paradigms 9](#_Toc122382979)

*2.2.4 Error Detection/ Exceptional Handling………………………………………………………………10*

*2.2.5 Memory Management……………………………………………………………………………………10*

[2.2.6 Performance 10](#_Toc122382980)

[2.2.7 Security 10](#_Toc122382981)

[2.2.8 Housekeeping and Maintenance 10](#_Toc122382983)

[3.QUIZ GAME FLOW CHART AND ER DAIGRAM 11](#_Toc122382984)

[3.1. Flow Chart Diagram 11](#_Toc122382985)

[3.2. UML Diagram 12](#_Toc122382986)

[3.2.1. Internal Interfaces 13](#_Toc122382989)

[3.2.2. External Interfaces 13](#_Toc122382990)

[4. Detailed System Design 13](#_Toc122382991)

[4.1. Key Entities 13](#_Toc122382992)

[5. Environment Description 14](#_Toc122382995)

[5.1. Time Zone Support 14](#_Toc122382996)

[5.2. Language Support 14](#_Toc122382997)

[5.3. User Desktop Requirements 14](#_Toc122382998)

[5.3.1. Deployment Considerations 14](#_Toc122383000)

[5.3.2. Application Server Disk Space 15](#_Toc122383001)

5.4 Pseudo code……………………………………………………………………………………..16

[6 Configuration 15](#_Toc122383010)

[6.1 Operating System 15](#_Toc122383006)

[6.2 Database 15](#_Toc122383007)

[6.3 Desktop 16](#_Toc122383009)

**7 REFERENCES………………………………………………………………………………...**16

[8 Conclusion 16](#_Toc122383011)

# **1.Introduction**

As the name suggests Bus Reservation System is software that handles the entire booking data of the bus. It is fully based on the concept of reserving bus tickets for various destinations. Previously the task of handling the tickets at a time was very difficult, so there was a need for software that can handle all bus reservation systems.

* Therefore, the Bus Reservation System was designed. It’ll reduce the stress and workload of the employee.
* Now it hardly takes 10-15 minutes to book a ticket wherever the passenger is in this bus reservation system, you can store the bus details and user details, we can view the bus details according to the destination or using PNR number or using the name of the bus.
* Only admin can add buses and do the reservation.
* User can login to the portal and he can book a ticket, view ticket and will be able to check the availability of seats particular bus, whether seats are reserved or not. Users are able to see the status of available buses.

## 1.1 Intended Audience

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

## 1.2 Acronyms/Abbreviations

|  |  |
| --- | --- |
| BRS | Bus reservation system |

|  |  |
| --- | --- |
| UML | Unified Modeling Language |

## 

## 1.3 Project Purpose

* The sole intention behind the consideration of this project is to book their tickets in online and they can select seat as per there wish.

## Key Project Objective

## To reduce the delay in processing time.

## To reduce the delay in records updating.

## To provide the user-friendliness in all possible ways.

## To provide greater flexibility.

## Project Scope and Limitation

The Bus Reservation System helps users to reserving bus tickets for various destination.

**Limitation**:

* User must need the internet access. Without internet user cannot book tickets.
  + 1. **In Scope**

The scope of this project should be used in as a bus transportation system a facility which is used to reserve seats, cancellation of reservation and different types of route enquiries used on securing reservation.

## Functional Overview

The functions that are used for admin part is mentioned below:

* *Admin ():* This function definition is used to login as a admin. He’ll manage the bus and booking functionalities. They are:

admin can add the bus, view all buses and details of bus, edit the bus, delete the bus, and admin will also work on ticket booking he can book the ticket, cancel the ticket, edit the ticket.

* *User ():* This function is used to login as a user and to book tickets or cancel tickets.

## Assumptions, Dependencies & Constraints

Assumption, dependency & constraints - system require

network connection. We must store the credentials of the users

in a file safely. And we also must store the website list in a separate file.

## Risk

## No Risk (As it is for educational purpose)

# **Design Overview**

# 

# 1.start

This the start blocks which indicates the star of the program. which will allow the user and user can login to the portal and he can book a ticket, view ticket and will be able to check the availability of seats particular bus, whether seats are reserved or not. Users can be able to see the status of available buses.

The entire activity is to automate the ticket booking process of day activities of system library like:

* Ticket booking.
* Assign a bus Ticket according to customer’s demand.
* Advance booking.
* Bus ticket cancellation.
* View bookings.

## 2.1 Design Objectives

* + The Bus Reservation enables admin to manage the details of add bus, delete bus, edit bus, view busses and manages to book tickets, cancellation tickets and provides a user-friendly interface.
  + The user can start the program whenever they want.
  + The user can login this using internet only.

### 2.1.1Recommended Architecture

**Client-side hardware interface**:

* Ubuntu/Linux machine
* Terminal

**Tools to be used:**

* G++ compiler
* Valgrind.
* Splint.
* Unit testing
* IT testing

## 2.2. Architectural Strategies

* The architectural strategy used in this project is an admin -admin can manages the details of the bus like add bus, delete bus, edit bus etc...
* User can manage the details of ticket i.e. book ticket, view ticket and cancel ticket. User strategy. This strategy consists of two entities, the admin and the user.

### 2.2.1 Reuse of Existing Common Services/Utilities

* Bus Reservation System by using ubuntu/Linux.

### 2.2.2 User Interface Paradigms

* Linux/Ubuntu machine
* gcc complier
  + 1. **System Interface Paradigms**
* Operating system: LINUX/Ubuntu
* Linux Kernel version: GNU/Linux 5.10.16.3-microsoft-standard-WSL2 x86\_64
* Bash shell: 5.0.16(1)-release

### 2.2.4 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.

* We have integrated the project again and again to resolve the errors. We must manage exceptional handling for bus and ticket functionalities. We have used the validation for exception handling.

### 2.2.5 Memory Management

N/A

### 2.2.6 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.

**2.2.7 Security**

* For security purpose Admin will store the all-user information in a file no one can see. we must secure the data in the directory

# **Bus Reservation System Flow Chart:**

## Flow Chart Diagram

Flow Chart Diagram is a diagrammatic representation of data movement through a system-manual or automated from inputs to outputs through processing. Flow chart helps in the analysis of the flow of data through a system and thus help identifying the system requirements.



## UML [Unified Modeling Language] Diagram

The unified modeling language [UML] is a pictorial language used to make software blueprints. UML can be described as a general-purpose visual modeling language to visualize, specify, construct & document software system.

### Internal Interface

The internal interfaces comprise interfaces through which the system interacts with the clients/users through which it provides them services.

### External Interfaces

The external interface comprises interfaces through which the users interact with the system.

* Desktop or Linux Machine

# **4.Detailed System Design**

The Bus Reservation System web portal that enables customers make an online bus tickets add () will be used to admin will add buses

and add the bus ticket, delete () can delete the bus and delete the tickets, edit () will be used to edit bus or edit ticket, generating of reports etc.

## Key Entities

Key Entities are associated with the systems are: -

**Admin: -**

The admin can manage adding bus, view buses, delete the bus, edit the bus, view bus ticket.

**User: -**

The user can Book the tickets, and he/she can view bookings and also, they can cancel the ticket.

# **5.Environment Description**

* G++: In Linux, the G++ 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.
* 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

We are using C++ programming language is a high-level general-purpose programming language created by Danish computer scientist Bjarne Stroustrup as an extension of the C Programming language. The C++ programming language was initially standardized in 1988 as ISO/IEC 14882:1998. There is different version in C++ they are C++03, C++11, C++14, C++17 and C++20.This is current version.

## 5.3 User Desktop Requirements

Linux/ Ubuntu.

### 5.3.1 Deployment Considerations

Deployment considerations are,

* 500Mhz Processor
* 120GB HDD CPU
* 4GB RAM

### 5.3.2Application Server Disk Space

Disk Space - Less space is required.

* 1. **Activity / class Diagram (as applicable)**

Pseudocode: Bus Reservation System

//Pseudocode for main

Start:

In this we have functions

function of welcomescreen():

print welcome to bus reservation management system

function of mainMenu():

declare int choice;

print main menu

print 1.admin menu

print 2.usermenu

print 3.Exit

print Enter your choice

read choice

By using switch case select the choice

If choice is set to 1

Calls function adminLogin()

declare string name, string password ;

print"ADMIN LOGIN";

read adminname and password;

check if(admin name and password are correct)

calls adminmenu();

else

invalid adminname and password

calls function usermenu():

else if choice is set to 2

print usermenu;

print 1.Book Ticket

print 2.view Ticket

print 3.Cancel Ticket

print 4.Exit

print Enter your choice

read choice

By using switch case enter your choice

function adminMenu():

declare bus b, ticket t;

declare int choice;

print "ADMIN PORTAL"

print 1.Add Bus\n2.view buses\n3.Book Ticket\n4.EditTicket

5.cancel Ticket\n6.view bookings\n7.view bus details\n8.edit bus\n9.delete bus\n10.back;

print enter your choice

read choice

by using switch case enter your choice

function viewbookingsmenu():

declare ticket t;

declare int choice;

print "VIEW BOOKINGS"

print 1.By PNR\n2.By name\n3.by bus\n4.by source\n5.by destination\n6.all bookings\n7.back

print enter your choice;

read choice

by using switch case enter your choice

choice is not present choose valid option

//Pseudocode: bus headerfile

class Bus

private:

declare int maxseats, bookedseats, Double busfare, char busno, source, destination, sourcetime, destination time;

public:

bus()//constructor

busno,maxseats,bookedseats,busfare,source,destination,sourcetime,destiation time;

function name addbus();

function name showall buses();

function name viewbus details();

function name delete bus();

function name edit bus();

declare getter and setter functions for all attributes;

Pseudocode: bus definition file:

class Bus :: function addbus()

file open

print "ADD BUS"

print all the details of bus like bus number, source, destination etc..

Read all the above details

file close

print "BUS ADDD SUCCESSFULLY";

class Bus :: Showbusdetails()

print all the bus details like busno, source, destination, fare, time;

class Bus :: function showallbus()

open the file fstream

check file (if file is not found)

else

print "BUSES"

open the busfilestream

read the data

while file found

display busdetails

close the file;

class Bus :: function viewbusdetails()

declare bno;

int ch;

open the file

print "VIEW BUS INFO"

check if(file is opened or not)

if(can't open file)

else

open the file and read the characters

check if (busno==0)

show bus details

else

bus not found

close the file

class Bus :: function edit bus()

declare bno, int ch;

open the file fstream bus and temp

print"EDIT BUS"

print bus number

read busnumber

check(if bustream file opened or not)

if(can't open file)

else

open the file

if(compare the getter function busno and bno)

print"EDIT BUS"

display show busdetails();

declare

char source, destination, time;

double fare;

print and read all the above details;

by using set method write the data into tempfilestream

else

open the two files

if(ch==1)

print bus updated successfully

else

bus not found

close the read and write files

remove buses data and rename the temp and buses data

class Bus :: function deletebus()

declare char bno, int ch;

open the file bus and temp;

print"DELETE BUS"

print busnumber;

read bus number;

open to file busstream

check the file is opened or not

if(file not opened)

else

temp file is opened and bus file will read

compare the getter busno and busno

if(!=0)

open the tempfile to write the data

else

ch=1;

open the busfile to read

if(ch==0)

print bus not found

else

bus deleted

close the bus and temp files

remove the bus data and rename the temp and bus data

end;

class Ticket

private:

declare char name, PNR no, date, bus;

public:

display the functions:

void generate ticket():

void display ticket():

void book ticket():

void cancel ticket():

void edit ticket():

void show ticketby PNR():

void show ticketby name():

void show ticket by bus():

void show ticket by source():

void show ticket by destination():

void show alltickets():

declare all getter and setter functions for private;

//definition file for ticket booking

in the main function

//display ticket details

void ticket :: display\_ticket();

{

print book tickets

if (busfilestream.fail())

print can't open file

else

{

call set source, set destination

print available buses

print please select bus from available buses

read from user

}

if user input is equal to zero

{

print no buses found

else

{

print enter bus number 10 and get bus number 10

}

if bus number is equal to source and destination getbookseats

if getbookedseats is greater then equal to 32

{

print seats not available

else

{

ask user to fill details for booking ticket

print heading booking details

}

}

// display ticket

display();

print ticket booked successfully

//function for cancel ticket

void ticket :: cancel\_ticket();

{

print enter PNR number

if ticketfilestream is fail

print can not open the file

else

close the ticketstream

if (check == 0)

print ticket not found

else

print ticket cancelled

}

//void ticket :: edit\_ticket

print enter PNR number

if ticketfilestream is fail

{

print can not open file

else

{

while (!ticketfilestream.eof())

if PNR number is equal to zero

print edit ticket enter passenger name

print updated successfully

}

else

print ticket not found

}

//function for show ticket by PNR

void ticket :: showticketsbypnr();

printheading show bookings by phone number

print enter PNR number

cin.getline PNR number must be 10 digits

if ticketfilestream is fail

{

print can not open file

else

{

read the file from ticketfilestream.read((char\*)this, size of(\*this));

print no bookings....

}

}

//show ticket by name

void ticket :: showticketbyname()

{

class Ticket

private:

declare char name, PNR no, date, bus;

public:

display the functions:

void generate ticket():

void display ticket():

void book ticket():

void cancel ticket():

void edit ticket():

void show ticketby PNR():

void show ticketby name():

void show ticket by bus():

void show ticket by source():

void show ticket by destination():

void show alltickets():

declare all getter and setter functions for private

//show ticket by name

void ticket :: showticketbyname()

{

class Ticket

private:

declare char name, PNR no, date, bus;

public:

display the functions:

void generate ticket():

void display ticket():

void book ticket():

void cancel ticket():

void edit ticket():

void show ticketby PNR():

void show ticketby name():

void show ticket by bus():

void show ticket by source():

void show ticket by destination():

void show alltickets():

declare all getter and setter functions for private;

//definition file for ticket booking

in the main function

//display ticket details

void ticket::display\_ticket();

{

print book tickets

if (busfilestram.fail())

print can't open file

else

{

call set source, set destination

print available buses

print please select bus from available buses

read from user

}

if user input is equal to zero

{

print no buses found

else

{

print enter bus number 10 and get bus number 10

}

if bus number is equal to source and destination getbookseats

if getbookedseats is greater then equal to 32

{

print seats not available

else

{

ask user to fill details for booking ticket

printheading booking details

}

}

// display ticket

display();

print ticket booked successfully

//function for cancel ticket

void ticket :: cancel\_ticket();

{

print enter PNR number

if ticketfilestream is fail

print can not open the file

else

close the ticketstream

if (check == 0)

print ticket not found

else

print ticket cancelled

}

//void ticket :: edit\_ticket

print enter PNR number

if ticketfilestream is fail

{

print can not open file

else

{

while (!ticketfilestream.eof())

if PNR number is equal to zero

print edit ticket enter passenger name

print updated successfully

}

else

print ticket not found

}

//show ticket by name

void ticket::showticketbyname()

{

printheading show booking by name

print enter passenger name

ticketfilestream.open("ticket.date", ios::on | ios::app | ios::binary);

if tickekfilestream is fail

print this file can not open

else

get name is equal to zero

print no bookings

}

//function for show ticket by bus

void ticket :: showticketbybus()

printheading showticketbybus

ticketfilestram.open(""ticket.date", ios::in | ios::app | ios::binary);

if ticketfilestream is fail

{

print can not open file

}

else

{

print no bookings...

}

//show ticket by source

void ticket :: showticketbysource()

printheading showticketbysource

print enter source

ticketfilestram.open(""ticket.date", ios::in | ios::app | ios::binary);

if ticketfilestream is fail

{

print file can not open

}

else

{

if bus get source is equal to zero

print no booking

}

//show ticket by destination

void ticket :: showticketbydestination()

printheading showticketbydestination

print enter destination

ticketfilestram.open(""ticket.date", ios::in | ios::app | ios::binary);

if ticketfilestream is fail

{

print file can not open

}

else

{

if bus get destination is equal to zero

print no booking

}

//show all booking

void ticket :: showalltickets()

printheading booking

ticketfilestram.open(""ticket.date ",ios::in | ios::app | ios::binary);

if ticketfilestream is fail

{

print file can not open

}

else

{

ticketFileStream.read((char \*)this, sizeof(\*this));

displayticket

ticketfilestream closed

}

### 6. Configuration

NA

### Operating System

Linux desktop editions with 8 GB RAM- A GUI-based LINUX system must be used

* 1. **Database**

File handling is used which refers to the method of storing data in the C++ program in the form of an output or input that might have been generated while running a C++ program in a data file.

**6.3 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

# **7 References**

**1.**[**Bus\_reservation\_system GeeksforGeeks**](https://www.geeksforgeeks.org/traceroute-in-network-layer/)

**2.https://www.geeksforgeeks.org/bus-reservation-system in-e-commerce/**

**3.Git hub**

# **8.CONCLUSION**

* Finally, in Bus Reservation System, we have developed a secure, user friendly Bus Reservation System.
* This project basically provides bus information. First of all on our website any user or visitor view our system and search the bus and how many seats are available in our buses.
* The user can also register their own seats on the bus on this website.