HOTEL MANAGEMENT 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** | |
| 14/01/2023 | HLD\_LLD Version 0.1 | Geethanjali Goddumarri |  | | | | Prashanth | |
| 14/01/2023 | HLD\_LLD Version 0.2 | Shridevi Prabhu S |  | | | | Prashanth | |
| 14/01/2023 | HLD\_LLD Version 0.3 | Rachala Manisha |  | | | | Prashanth | |
| 14/01/2023 | HLD\_LLD Version 0.4 | Aishwarya Palled |  | | | | Prashanth | |
| 14/01/2023 | HLD\_LLD Version 0.5 | B.R. Swathi |  | | | | Prashanth | |
| 14/01/2023 | HLD\_LLD Version 0.6 | Mallela Reddypriya |  | | | | Prashanth | |

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

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

[1.2. Acronyms/Abbreviations](#_heading=h.tyjcwt) 5

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

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

[1.5. Project Scope and Limitation](#_heading=h.4d34og8) 6

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

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

[1.8. Risks](#_heading=h.lnxbz9) 6

[**2. Design Overview**](#_heading=h.35nkun2) **7**

2[.1 Design Objectives](#_heading=h.1ksv4uv) 8

[*2.1.1. Recommended Architecture*](#_heading=h.44sinio) 9

[2.2. Architectural Strategies](#_heading=h.2jxsxqh) 10

[*2.2.1. Design Alternative*](#_heading=h.z337ya) 10

[*2.2.2. Reuse of Existing Common Services/Utilities*](#_heading=h.3j2qqm3) 10

[*2.2.3. Creation of New Common Services/Utilities*](#_heading=h.1y810tw) 10

[*2.2.4. User Interface Paradigms*](#_heading=h.4i7ojhp) 10

[*2.2.5. System Interface Paradigms*](#_heading=h.2xcytpi) 10

[*2.2.6. Error Detection / Exceptional Handling*](#_heading=h.1ci93xb) 10

[*2.2.7. Memory Management*](#_heading=h.3whwml4) 10

[*2.2.8. Performance*](#_heading=h.2bn6wsx) 11

[*2.2.9. Security*](#_heading=h.qsh70q) 11

[*2.2.10. Concurrency and Synchronization*](#_heading=h.3as4poj) 11

[*2.2.11. Housekeeping and Maintenance*](#_heading=h.1pxezwc) 11

[**3. System Architecture**](#_heading=h.49x2ik5) **12**

[3.1. System Architecture Diagram](#_heading=h.2p2csry) 15

[3.2. System Use-Cases](#_heading=h.147n2zr) 16

[3.3. Subsystem Architecture](#_heading=h.3o7alnk) 16

[3.4. System Interfaces](#_heading=h.23ckvvd) 16

[*3.4.1. Internal Interfaces*](#_heading=h.ihv636) 16

[*3.4.2. External Interfaces*](#_heading=h.32hioqz) 16

[**4. Detailed System Design**](#_heading=h.1hmsyys) **16**

[4.1. Key Entities](#_heading=h.41mghml) 16

[4.2. Detailed-Level Database Design](#_heading=h.2grqrue) 16

[*4.2.1. Data Mapping Information 1*](#_heading=h.vx1227)6

[*4.2.2. Data Conversion 1*](#_heading=h.3fwokq0)6

[4.3. Archival and retention requirements](#_heading=h.1v1yuxt) 17

[4.4. Disaster and Failure Recovery](#_heading=h.4f1mdlm) 17

[4.5. Business Process workflow](#_heading=h.2u6wntf) 17

[4.6. Business Process Modeling and Management (as applicable)](#_heading=h.19c6y18) 17

[4.7. Business Logic](#_heading=h.3tbugp1) 17

[4.8. Variables](#_heading=h.28h4qwu) 17

[4.9. Activity / Class Diagrams (as applicable)](#_heading=h.nmf14n) 17

4.10.*Pseudocode*  17

[4.11. Data Migration](#_heading=h.37m2jsg) 20

[*4.11.1. Architectural Representation*](#_heading=h.1mrcu09) 20

[*4.11.2. Architectural Goals and Constraints*](#_heading=h.46r0co2) 20

[*4.11.3. Logical View*](#_heading=h.2lwamvv) 20

[*4.11.4. Architecturally Significant Design Packages*](#_heading=h.111kx3o) 20

[*4.11.5. Data model*](#_heading=h.3l18frh) 21

[*4.11.6. Deployment View*](#_heading=h.1egqt2p) 21

[**5. Environment Description**](#_heading=h.3ygebqi) **21**

[5.1. Time Zone Support](#_heading=h.2dlolyb) 21

[5.2. Language Support](#_heading=h.sqyw64) 21

[5.3. User Desktop Requirements](#_heading=h.3cqmetx) 21

[5.4. Server-Side Requirements](#_heading=h.1rvwp1q) 21

[*5.4.1. Deployment Considerations*](#_heading=h.4bvk7pj) 22

[*5.4.2. Application Server Disk Space*](#_heading=h.2r0uhxc) 22

[*5.4.3. Database Server Disk Space*](#_heading=h.1664s55) 22

[*5.4.4. Integration Requirements*](#_heading=h.3q5sasy) 22

[*5.4.5. Jobs*](#_heading=h.25b2l0r) 22

[*5.4.6. Network*](#_heading=h.kgcv8k) 22

[*5.4.7. Others*](#_heading=h.34g0dwd) 22

[5.5. Configuration](#_heading=h.1jlao46) 22

[*5.5.1. Operating System*](#_heading=h.43ky6rz) 22

[*5.5.2. Database*](#_heading=h.2iq8gzs) 22

[*5.5.3. Network*](#_heading=h.xvir7l) 22

[*5.5.4. Desktop*](#_heading=h.3hv69ve) 22

[**6. References**](#_heading=h.1x0gk37) **23**

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

# 

# 

# 1. Introduction

The Hotel management system is a system that aids the management of the customer’s data, customer’s registration, customer accommodation or allocation into specific rooms. The main objective of the Hotel management information system is to manage the details of Customer and Bookings. It manages all the information about Users. The project is totally built at administrative end and thus only the administrator is guaranteed the access.

* The users can track the information related to the available hotels and rooms with great ease through this application. This system will include all the necessary fields which are required to book rooms. This system will be easy to use and can be used by any person. The basic idea behind this project is to save data, time and burden which are being faced by their customers. Customer can register by providing personal details and book a room at the Hotel.

## 1.1 Intended Audience

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

## 1.2 Acronyms/Abbreviations

|  |  |  |
| --- | --- | --- |
| HLD | High Level Design | |
| LLD | Low-Level Design | |
| HMS | Hotel Management System | |
|  | |

## 1.3 Project Purpose

The main purpose of this project is to build an application program to reduce the manual work for managing hotel room Bookings. This application will help in accessing the information regarding the available hotel rooms. This application gives admin the power and flexibility to manage the entire system from a single system. This hotel management system makes booking easy as it is simple.

## 1.4 Key Project Objectives

The objective is designed to create a platform that allows both the admin and customer to keep track of transactions like room booking. It seeks to improve efficiency and Operational process performance.

* To be able to get work faster
* To be able to manage fast access to the customer details
* To be able to easy update of the customer record
* To be able to reduce paper use at the hotel

**1.5 Project Scope and Limitation**

The Hotel Management System helps Customers to book the specific rooms by giving their personal information like Name, Mobile number, Government ID proof and Address.

LIMITATION:

* Customers are not allowed change data information such as room types, admin password.
* Retrieval of Customer records are difficult.

### 1.5.1 In Scope

* The Hotel Management System is a tool for booking rooms of Hotel through this application.
* The HMS project is intended for the booking for room that can be made through application, it will be able to automate the various operations of the Hotel.
* Hotel Management System will consist of Booking Management System server and report generator Customer will be able to check for room’s availability, select the rooms and pay for the room.

## 1.6 Functional Overview

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

* Admin (): This function is used to login as admin. He will manage the Hotel and booking functionalities are:

add\_room(), display\_room(),modify\_room(),search\_customer(), view\_customer(),exit()

* Customer (): This function is used in customer module are:

register\_customer(), book\_room(), search\_room(), checkout\_room(),exit()

## 1.7 Assumptions, Dependencies & Constraints

OPERATING SYSTEMS:

Operating environment for implementing are:

* Client/server system
* Operating system: Linux
* Platform: Ubuntu

## 1.8 Risks

No Risk (As it is for educational purpose).

# 

# Design Overview

1. **Start**

* This is the start block which indicates the start of the program.

1. **Login Credentials**

* In this module the credentials entered by the user are then validated by the system. If the user or admin enters valid credentials, then it will move to the further step else the system will ask user to register.

1. **Customer module functions:**

* Register of new customer

Here user must register by giving details like name, address, phone number.

* Book a Room

Once the user has registered then, user will now be connected to the main menu. Once connected to the main menu the user will now be able to book the room as per his requirement.

* Search Room

User can search the room which are available in the hotel.

* Check Out Room

It is the process of customer leaving the hotel by paying the bill and

returning keys to the hotel.

* Exit

This ensures that the customer module program has terminated.

1. **Admin module functions:**

* Add room: In this functionality admin can add the rooms.
* Display available rooms: Here admin can display all available rooms.
* Modify the rooms: In this functionality admin can modify any rooms.
* View Customer: Admin can view all the registered customers.
* Search Customer: In this functionality admin can search any specific customer.
* Exit: In this system admin will come out from the system after his task.

1. **End**

* This will ensure the program has terminated.

## 2.1 Design Objectives

Create login credential page for both server and client

Take domain name as an input from the client after successful login

Server will check whether domain names exist or not.

If exists, server will send IP address to client

* + 1. **Recommended Architecture**

**Client-based hardware interface:**

* Ubuntu/Linux machine
* Terminal

**Tools to be used:**

* g++ compiler
* Valgrind
* Splint
* Unit testing
* IT testing

## 2.2 Architectural Strategies

* Header files
* Structures
* Macros

### 2.2.1 Design Alternative

NA

### 2.2.2 Reuse of Existing Common Services/Utilities

#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

#include<string.h>

### 

### 2.2.3 Creation of New Common Services/Utilities

NA

**2.2.4 User Interface Paradigms**

Command Line Interface: Terminal

**2.2.5 Housekeeping and Maintenance**

NA

**2.2.6 System Interface Paradigms**

Command Line Interface: Terminal

**2.2.7 Error Detection / Exceptional Handling**

Admin needs to enter the required login credentials else an error will occur and admin has to re-login.

Phone number should contain the 10digits.

**2.2.8** **Memory Management**

NA

* + 1. **Performance**

NA

**2.2.10** **Security**

For security purposes the system asks for login credentials from admin.

* + 1. **Concurrency and Synchronization**

NA

## 

## 3.1 System Architecture

## ER Diagram:

Diagram

Description automatically generated

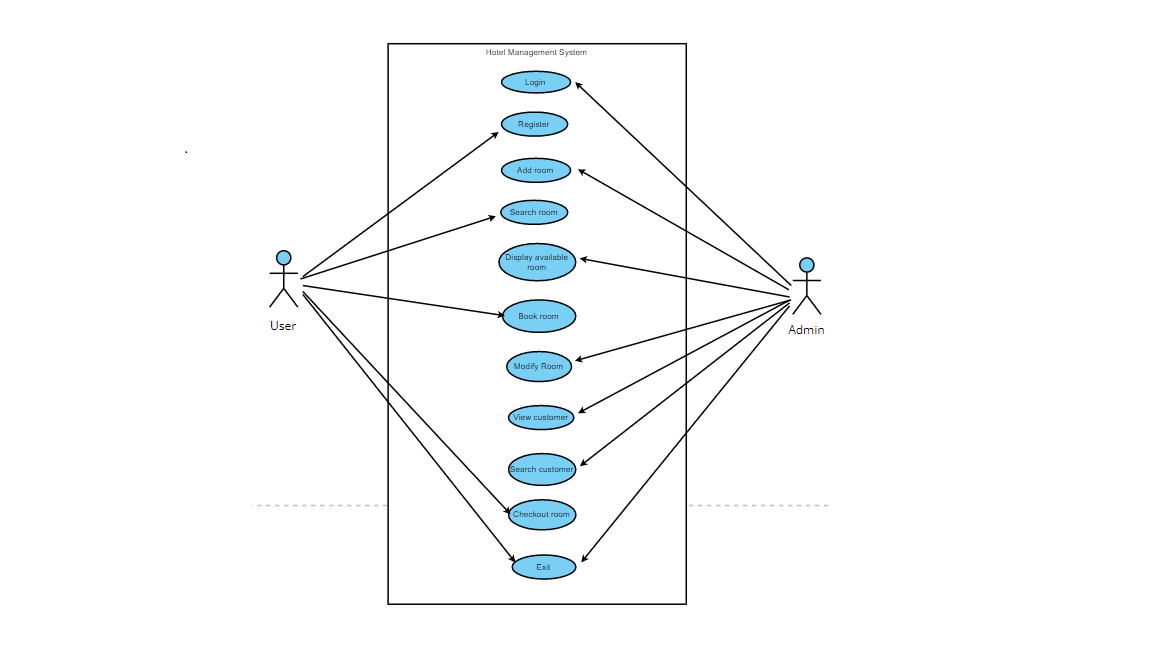
## 

## Flowchart

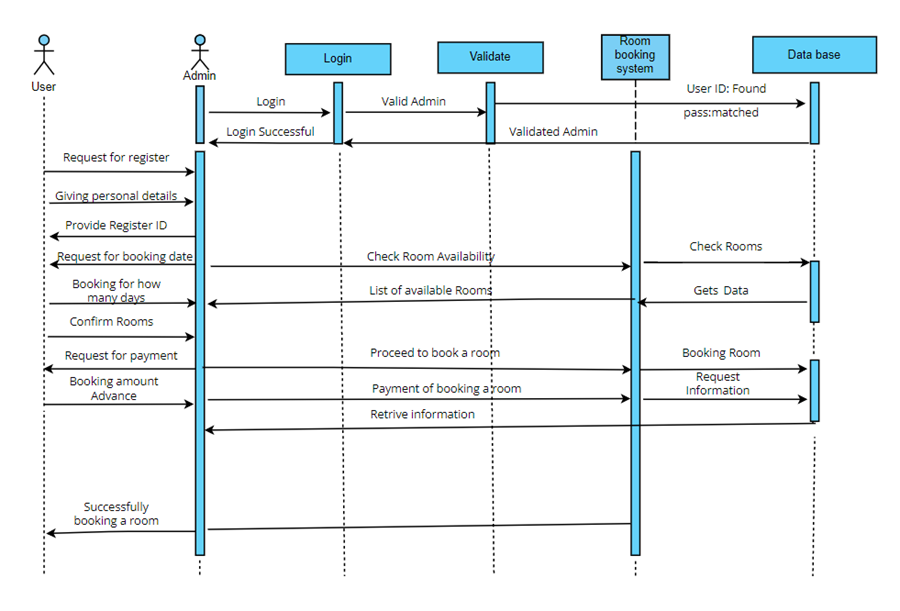
Graphical user interface

Description automatically generated with medium confidence

* **Use case diagram:**

****

* **Sequence diagram:**



## 3.2 System Use-Cases

NA

**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 declaring admin login function as admin can login to the system by providing required login credentials like username and password. And admin can perform the operations such as add room, display available rooms, modify the room, view customer, search customer.

## 4.1 Key Entities

* Admin login
* Customer
* add room, display available rooms, modify the room, view customer, search customer.

## 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

## 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)

NA

**4.10.Pseudocode:**

Admin Module

1. First we must check with the login credentials

2. Admin must enter the password

3. Check for the condition

4. If (ID) and (Password) is valid then display the admin functions

5. Else

6. Display the message “ID and Password doesn’t match”

7. Create a function for view customer

8. Declare a variable integer i

9. Set found=1

10. Display Customer details

11. print customer name, address, phone number, booking id

12. Define function for search customer

13. set count=0 and found=0

14. if entered customer id is found

15. print customer name, address, phone number, booking\_ id

16. Define function for guest summary report

17. set count=0

18. if count is equal to zero

19. Display details of customer

20. check for the condition if the room status=1

21. if the condition is true

22. Display customer name, room number, address, phone number

23. end if

Customer Module Pseudocode

1.Create class Customer

2.Declar class variables character name, character address, integer phone, character

from date, character to date, float payment advance, integer booking id.

3.create classroom

4.Declare class variables character type, character stype, character ac, integer

roomnumber, integer rent, integer status;

5.Declare function for register\_customer(), book\_room(), search\_room, checkout\_room(), exit()

6.In register function customer is registering using his details

7.In book room function the customer is booking the room

8.In search\_room function customer is searching for room by providing room number

9.In checkout room customer making payment

10.write menu to perform operation using switch case

11.Declare variable integer choice

12.print

1.Register of new customer

2.Book room

3.Search room

4.Checkout\_room

5.Exit

13.print enter the choice

14.passing choice to switch case

15.In case 1: calling function register\_customer()

16.In case 2: calling function book\_room()

17.In case 3: calling function search\_room ()

18.In case 4: calling function Checkout\_customer ()

20.In case 6: calling function exit()

21.In case default: print invalid choice.

## 4.11. Data Migration

NA

### 4.11.1 Architectural Representation

NA

### 4.11.2 Architectural Goals and Constraints

This project is a tool that is meant to make hotel booking system easier.

### 4.11.3 Logical View

NA

### 4.11.4 Architecturally Significant Design Packages

NA

### 

### 4.11.5 Data model

NA

**Legacy system data model**

**Proposed system data model**

**Interface data model**

**4.11.6 Deployment View**

NA

# Environment Description

GCC: In Linux, the GCC stands for GNU Compiler Collection. It is a compiler system for the various programming languages. It is mainly used to compile the C and C++ programs.

UBUNTU: Ubuntu is an open-source operating system (OS) based on the Debian GNU/Linux distribution. Ubuntu incorporates all the features of a Unix OS with an added customizable GUI, which makes it popular in universities and research organizations. Ubuntu is primarily designed to be used on personal computers, although a server edition does also exist.

GITHUB: GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. This tutorial teaches you GitHub essentials like repositories, branches, commits, and pull requests.

## 5.1 Time Zone Support

NA

## 5.2 Language Support

NA

## 5.3 User Desktop Requirements

Linux, Ubuntu

## 

## 5.4 Server-Side Requirements

Linux, Ubuntu

### 5.4.1 Deployment Considerations

NA

### 5.4.2 Application Server Disk Space

NA

### 5.4.3 Database Server Disk Space

NA

### 5.4.4 Integration Requirements

NA

### 5.4.5 Jobs

NA

### 5.4.6 Network

NA

### 5.4.7 Others

NA

## 5.5 Configuration

NA

### 5.5.1 Operating System

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

used

### 5.5.2 Database

NA

### 5.5.3 Network

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.cppbuzz.com/projects/c++/c++-project-on-hotel-management
* https://code-projects.org/simple-hotel-management-system-in-c-with-sourcecode/
* https://www.geeksforgeeks.org/hotel-management-project-in-python/