System Requirements Specification

for

<Hotel Management System>

Version 1.0

Prepared by

<K.Greeshma 2203A51052>

<D.Hasini 2203A51044>

<G.Abhi 2203A51046>

<Ch.Sri Charan 2203A51041>

Group Code

**<Group Code> − SA1**

B.Tech. CSE, Section – C,

LAB. Batch – 9

SR UNIVERSITY

<20-02-2024 >

Table of Contents

1.0 Introduction

1.1 Purpose

1.2 Scope of Product

1.3 Definitions, Acronyms, and Abbreviations

1.4 References

1.5 Overview

2.0 General Description

2.1 Product Perspective

2.2 Product Functions

2.3 User Characteristics

2.4 General Constraints

2.5 Assumptions and Dependencies

3.0 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interface

3.1.2 Software Interfaces

3.1.3 Hardware interfaces

3.1.4 Communication Interfaces

3.2 Functional Requirements

3.2.1 Registration

3.2.2 Logging in

3.2.3 Reservation

3.2.4 Receptionist Access

3.2.5 Manager Access

3.2.6 Payment management System

3.3 Performance Requirements

3.4 Security Requirements

3.5 Safety Requirements

3.6 Capacity Requirements

3.7 Availability Requirements

3.8 Software System Atrributes

3.9 Requirements Traceability Matrix

APPENDIXES

Data Flow Diagram

Use Case diagram

Class Diagram

Object Diagram

State Chart Diagram

Activity Diagram

Sequence Diagram

Component Diagram

**1.0 Introduction**

**1.1 Purpose:**

The current hotel management system lacks flexibility in pricing, using a uniform rate for all room bookings regardless of the stay duration. This approach overlooks the needs of guests with shorter stays, creating dissatisfaction. To address this, a dynamic pricing model is needed, considering the length of stay as a key factor. This model aims to provide fair pricing by proportionally reducing costs for shorter stays, enhancing customer satisfaction, transparency, and overall competitiveness. The objective is to attract a diverse guest base and improve the efficiency of the hotel's pricing structure.

**1.2 Scope of Product:**

The HMS project is intended for the reservations for room that can be made through online. It will be able to automate the various operations of the Hotel. Our Hotel Management System will have three end users: Customer, Receptionist, and Hotel Manager. Hotel Management System will consists of Booking Management System, DBMS Server, and Report Generator. Customers will Hotel Management System 2 | P a g e be able to check for room’s availability, select the rooms, and pay for the room. Receptionist will have access to update or modify booking details. Manager will able to view the financial report and able to update room information such as cost and category. The main goal of this introduced automated HMS software is to simplify every day process of hotel. Day to day Hotels are increasing and they need to automate to provide customer ease of access. It will be able to take care of services to customer in a quick manner. This automation will be able to replace the drawbacks of large customer information physical files which were difficult to handle. Secure Transaction, quick retrieval of information, ease of use, quick recovery of errors, fault tolerance are some of the benefits that development team will be working on to achieve end user satisfaction.

**1.3 Definitions, Acronyms, and Abbreviations:**

SRS- Software Requirement Specifications

HMS -Hotel Management System

DBMS -Database Management System

Blueprint - A design technical plan

JDBC- Java Database Connectivity

HTTP/HTTPS- Hyper Text Transfer Protocol/Secure

EJB- Enterprise Java Beans Hotel Management System

API- Application Interface

OS -Operating System

JSP- Java Server Pages

RTM -Requirement Traceability Matrix

FR -Functional Requirement

NFR- Non Functional Requirement

**1.4 References:**

[1] Software Engineering 9th Edition, Ian Sommerville

[2] Fundamentals of Database System, 6th Edition, Ramez Elmasri, Shamkant B. Navathe

[3]ER DiagramTutorial:

<https://www.tutorialspoint.com/dbms/er_diagram_re>presentation.html

[4] Requirement Engineering: <http://morse.inf.unideb.hu/valseg/gybitt/07/ch02.html>

[5] Hotel Management System: https://www.scribd.com/doc/63824633/Hotel-ManagementSystem Hotel Management System

[6] Case Study: https://www.scribd.com/doc/27927992/Hotel-Management-Case-Study

[7] Data Flow Diagram: <http://myyee.tripod.com/cs457/dfd.htm>

[8] Requirement Engineering: <https://en.wikipedia.org/wiki/Requirements_engineering>

**1.5 Overview:**

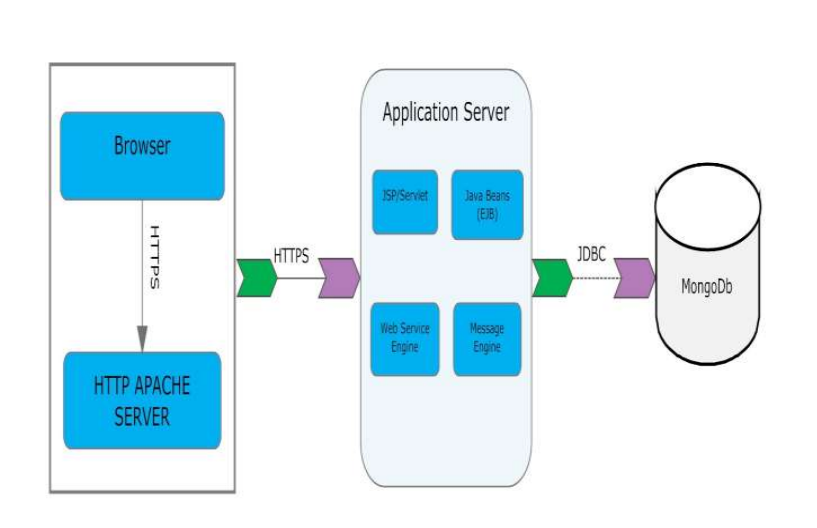
The remaining sections of this documentations describes the overall descriptions which includes product perspective and functions, characteristics of users. It also consists of Assumptions, and Constraints. Overall description is listed in section 2. Section 3 includes Specific Requirements which consists of Functional and Non-functional requirements, External Interface Requirements, Software System Attributes, Performance Requirements, Capacity Requirements, Availability Requirements, Safety Requirements and Requirement Traceability Matrix.

**2.0 General Description**

Presents the environment in which the application is expected to operate, provides an overview of the system requirements, describes assumptions about possible users of the application, possible constraints on the project, and the underlying assumptions that on which the requirements analysis is based. **Specific Requirements:**

The specification of requirements. Contains a complete description of the application's requirements, both functional and non-functional.

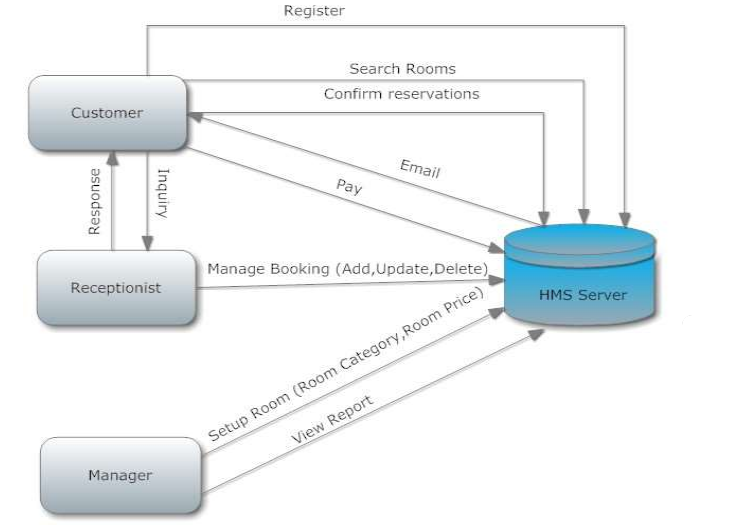
**2.1 Product Perspective:**



**2.2 Product Functions:**

Our Product General functions are:

* Customer Registration
* Check for Availability Of Rooms
* Display the Rate
* Confirmation Of Booking
* Email Notification
* Payment
* Set Room Details
* Manage Booking Details
* Generate Report
* Customer Service



**2.3 User Characteristics**

There are 3 user Levels in our Hotel Management System:

* Hotel Manager
* Receptionist
* Customers Hotel Management System

**Hotel Manager :**

Manager have every access to the hotel system. Manager is solely responsible for managing hotel resources and staffs. Manager can view any report such as financial report, customer information, booking information, and room information, analyze them and take the decision accordingly. Manger is required to have experience on managing hotel previously, and have base knowledge of database and application server.

**Receptionist:**

Hotel Receptionist sole purpose is to provide the quality customer service. She have least access than manager. She can manage the booking details. She can search for availability of rooms, add the customer, confirm the booking, and update the booking details. Manager of hotel would probably want the receptionist who have good communication skills and command over English language. She should have basic IT Knowledge.

**Customer:**

Customer are vital part of the system. Customer have access to view the vacant room information and price range. They should be able to confirm the booking and cancel it if necessary. Customers have access to customer service desk portal to forward their inquiry. Customer should at least be capable to use the web UI interface.

**2.4 General Constraints:**

* Memory: System will have only 10GB space of data server.
* Language Requirement: Software must be only in English.
* Budget Constraint: Due to limited budget, HMS is intended to very simple and just for
* basic functionalities. UI is going to be very simple.
* Implementation Constraint: Application should be based on Java only.
* Reliability Requirements: System should sync frequently to backup server in order to
* avoid the data loss during failure, so it can be recovered.

**2.5 Assumptions and Dependencies:**

It is assumed that system developed will work perfectly that’s going to be developed under the Windows OS, and Apache Server with Mongo DB database. If incase of any difficulties, SRS should be flexible enough to change accordingly.

**CASE TOOLS:**

**INTRODUCTION:**

CASE tools known as Computer-aided software engineering tools is a kind of component-based development which allows its users to rapidly develop information systems. The main goal of case technology is the automation of the entire information systems development life cycle process using a set of integrated software tools, such as modeling, methodology and automatic code generation. Component based manufacturing has several advantages over custom development. The main advantages are the availability of high quality, defect free products at low cost and at a faster time. The prefabricated components are customized as per the requirements of the customers. The components used are pre-built, ready-tested and add value and differentiation by rapid customization to the targeted customers. However the products we get from case tools are only a skeleton of the final product required and a lot of programming must be done by hand to get a fully finished, good product.

**CHARACTERISTICS OF CASE:**

Some of the characteristics of case tools that make it better than customized development are;

* It is a graphic oriented tool.
* It supports decomposition of process.

Some typical CASE tools are:

* Unified Modeling Language
* Data modeling tools, and
* Source code generation tools

**INTRODUCTION TO UML (UNIFIED MODELING LANGUAGE):**

The UML is a language for specifying, constructing, visualizing, and documenting the software system and its components. The UML is a graphical language with sets of rules and semantics. The rules and semantics of a model are expressed in English in a form known as OCL (Object Constraint Language). OCL uses simple logic for specifying the properties of a system. The UML is not intended to be a visual programming language. However it has a much closer mapping to object-oriented programming languages, so that the best of both can be obtained. The UML is much simpler than other methods preceding it. UML is appropriate for modeling systems, ranging from enterprise information system to distributed web based application and even to real time embedded system. It is a very expensive language addressing all views needed to develop and then to display system even though understand to use. Learning to apply UML effectively starts forming a conceptual mode of languages which requires learning.

Three major language elements:

* + UML basic building blocks
  + Rules that dictate how this building blocks put together
  + Some common mechanism that apply throughout the language

The primary goals in the design of UML are:

1. Provides users ready to use, expressive visual modeling language as well so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts.
7. Integrate best practices and methodologies.

Every complex system is best approached through a small set of nearly independent views of a model. Every model can be expressed at different levels of fidelity. The best models are connected to reality. The UML defines nine graphical diagrams:

1. Class diagram
2. Use-case diagram
3. Behavior diagram
   1. Interaction diagram
      1. sequence diagram
      2. collaboration diagram

3.2. state chart diagram

3.3. activity diagram

4. Implementation diagram

4.1. component diagram

4.2. deployment diagram

**1. UML class diagram:**

The UML class diagram is also known as object modeling. It is a static analysis diagram. These diagrams show the static structure of the model. A class diagram is a connection of static model elements, such as classes and their relationships, connected as a graph to each other and to their contents.

**2. Use-case diagram:**

The functionality of a system can be described in a number of different use-cases, each of which represents a specific flow of events in a system. It is a graph of actors, a set of use-cases enclosed in a boundary, communication, associations between the actors and the use-cases, and generalization among the use-cases.

**3. Behavior diagram:**

It is a dynamic model unlike all the others mentioned before. The objects of an object oriented system are not static and are not easily understood by static diagrams. The behavior of the class’s instance (an object) is represented in this diagram. Every use-case of the system has an associated behavior diagram that indicates the behavior of the object. In conjunction with the use-case diagram we may provide a script or interaction diagram to show a time line of events. It consists of sequence and collaboration diagrams.

**4. Interaction diagram**:

It is the combination of sequence and collaboration diagram. It is used to depict the flow of events in the system over a timeline. The interaction diagram is a dynamic model which shows how the system behaves during dynamic execution.

**5. State chart diagram:**

It consists of state, events and activities. State diagrams are a familiar technique to describe the behavior of a system. They describe all of the possible states that a particular object can get into and how the object's state changes as a result of events that reach the object. In most OO techniques, state diagrams are drawn for a single class to show the lifetime behavior of a single object.

**6. Activity diagram**:

It shows organization and their dependence among the set of components. These diagrams are particularly useful in connection with workflow and in describing behavior that has a lot of parallel processing. An activity is a state of doing something: either a real-world process, or the execution of a software routine.

**7. Implementation diagram:**

It shows the implementation phase of the systems development, such as the source code structure and the run-time implementation structure. These are relatively simple high level diagrams compared to the others seen so far. They are of two sub-diagrams, the component diagram and the deployment diagram.

**8. Component diagram:**

These are organizational parts of a UML model. These are boxes to which a model can be decomposed. They show the structure of the code itself. They model the physical components such as source code, user interface in a design. It is similar to the concept of packages.

**9. Deployment diagram:**

The deployment diagram shows the structure of the runtime system. It shows the configuration of runtime processing elements and the software components that live in them. They are usually used in conjunction with deployment diagrams to show how physical modules of code are distributed on the system.

**NOTATION ELEMENTS:**

These are explanatory parts of UML model. They are boxes which may apply to describe and remark about any element in the model. They provide the information for understanding the necessary details of the diagrams.

**Relations in the UML:**

These are four kinds of relationships used in an UML diagram, they are:

* + Dependency
  + Association
  + Generalization
  + Realization

**Dependency:** It is a semantic relationship between two things in which a change one thing affects the semantics of other things. Graphically a dependency is represented by a non-continuous line.

**Association:** It is a structural relationship that describes asset of links. A link is being connected among objects. Graphically association is represented as a solid line possibly including label.

**Generalization:** It is a specialized relationship in which the specialized elements are substitutable for object of the generalized element. Graphically it is a solid line with hollow arrow head parent.

**Realization:** It is a semantic relation between classifiers. Graphically it is represented as a cross between generalization and dependency relationship.

**Where UML can be used:**

UML is not limited to modeling software. In fact it is expressive to model non-software such as to show in structure and behavior of health case system and to design the hardware of the system.

**Conceptual model be UML:**

UML you need to form the conceptual model of UML. This requires three major elements:

* + UML basic building blocks.
  + Rules that dictate how this building blocks are put together.
  + Some common mechanism that apply throughout the language.

Once you have grasped these ideas, you may be able to read. UML create some basic ones. As you gain more experience in applying conceptual model using more advanced features of this language.

**Building blocks of the UML:**

The vocabulary of UML encompasses these kinds of building blocks.

**Use CASE definition:**

**Description:** A use case is a set of scenarios tied together by a common user goal. A use case is a behavioral diagram that shows a set of use case actions and their relationships.

**Purpose:** The purpose of use case is login and exchange messages between sender and receiver (Email client).

**Main flow:** First, the sender gives his id and enters his login. Now, he enters the message to the receiver id.

**Alternate flow**: If the username and id by the sender or receiver is not valid, the administrator will not allow entering and “Invalid password” message is displayed.

**Pre-condition:** A person has to register himself to obtain a login ID. Post-condition: The user is not allowed to enter if the password or user name is not valid.

**Class diagram:**

**Description:**

* + A class diagram describes the type of objects in system and various kinds of relationships that exists among them.
  + Class diagrams and collaboration diagrams are alternate representations of object models.

During analysis, we use class diagram to show roles and responsibilities of entities that provide email client system behaviors design. We use to capture the structure of classes that form the email client system architecture.

**A class diagram is represented as**:

<<Class name>>

<<Attribute 1>>

<<Attribute n>>

<<Operation ()>>

**Relationship used:**

A change in one element affects the other.

**Generalization:** It is a kind of relationship

**State chart:**

**Description:**

* The state chart diagram made the dynamic behavior of individual classes.
* State chart shows the sequences of states that an object goes through events and state transitions.
* A state chart contains one state ‘start’ and multiple ‘end’ states.

The important objectives are:

**Decision:** It represents a specific location state chart diagram where the work flow may branch based upon guard conditions.

**Synchronization:** It gives a simultaneous workflow in a state chart diagram. They visually define forks and joints representing parallel workflow.

**Forks and joins:**

* A fork construct is used to model a single flow of control.
* Every work must be followed by a corresponding join.
* Joints have two or more flow that unit into a single flow.

**State:**

A state is a condition or situation during a life of an object in which it satisfies condition or waits for some events.

**Transition:** It is a relationship between two activities and between states and activities.

**Start state:** A start state shows the beginning of a workflow or beginning of a state machine on a state chart diagram.

**End state:** It is a final or terminal state.

**Activity diagram** :

**Description:**

Activity diagram provides a way to model the workflow of a development process. We can also model this code specific information such as class operation using activity diagram. Activity diagrams can model different types of diagrams. There are various tools involved in the activity diagram.

**Activity:** An activity represents the performance of a task on duty. It may also represent the execution of a statement in a procedure.

**Decision:** A decision represents a condition on situation during the life of an object, which it satisfies some condition or waits for an event.

**Start state:** It represents the condition explicitly the beginning of a workflow on an activity.

**Object flow:** An object on an activity diagram represents the relationship between activity and object that creates or uses it.

**Synchronization**: It enables us to see a simultaneous workflow in an activity.

**End state:** An end state represents a final or terminal state on an activity diagram or state chart diagram.

**Sequence diagram:**

**Description:**

A sequence diagram is a graphical view of scenario that shows object interaction in a time based sequence what happens first what happens next. Sequence diagrams are closely related to collaboration diagram.

The main difference between sequence and collaboration diagram is that sequence diagram show time based interaction while collaboration diagram shows objects associated with each other.

The sequence diagram for the e-mail client system consists of the following objectives:

**Object:**

An object has state, behavior and identity. An object is not based is referred to as an instance.

The various objects in e-mail client system are:

* + User
  + Website
  + Login
  + Groups

**Message icon:** A message icon represents the communication between objects indicating that an action will follow. The message icon is the horizontal solid arrow connecting lifelines together.

**Collaboration diagram:**

**Description:**

Collaboration diagram and sequence diagrams are alternate representations of an interaction. A collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. Collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction.

Collaboration diagram shows object s, their links and their messages. They can also contain simple class instances and class utility instances.

During, analysis indicates the semantics of the primary and secondary interactions. Design, shows the semantics of mechanisms in the logical design of system.

Toggling between the sequence and collaboration diagrams When we work in either a sequence or collaboration diagram, it is possible to view the corresponding diagram by pressing F5 key.

CONCLUSION:

Thus the study for case tools was done.

**3.0 Specific Requirements:**

**3.1 Functional Requirements:**

**3.1.1 User Interface:**

The user interface requirements are concerned with the user interface and how information is presented to the user.

**3.1.2 Software Interfaces:**

Web Server :

• Apache Tomcat Server , OS (Windows)

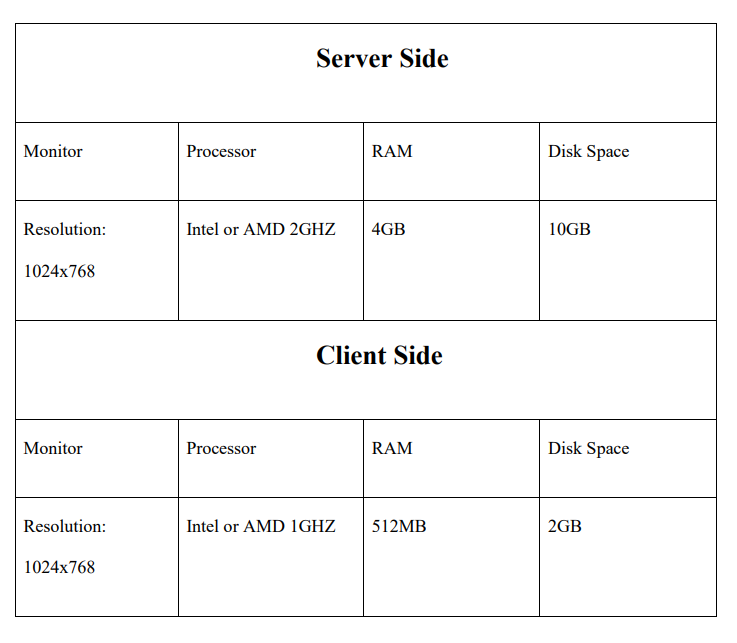
Database Server :

• Mongo DB, OS (Windows)

Development End :

• J2EE,Java,JSP,Servlet,HTML,XML,JavaScript, OS(Windows)

**3.1.3 Hardware Interfaces:**



**3.1.4 Communication Interfaces:**

The System shall be using HTTP/HTTPS for communication over Internet and for intranet communications, it shall use TCP/IP protocol.

**3.2 Functional Requirements:**

**3.2.1 Registration**:

FR1. The Customer should be able to register with their details

FR2. The system should record following customer details into member database.

* Name
* Email
* Password
* Address
* DOB

FR3. The system shall send verification message to email

**3.2.2 Logging In:**

FR4. The system should verify the customer email & password against the member database when logging in

FR5. After login, member should be directed to Home screen Hotel Management System

**3.2.3 Reservation:**

FR6. The system should enable customer to check for availability of rooms

FR7. The system should display rate for all rooms

FR8. The system should allow customer to confirm or cancel the booking

FR9. The system should record booking details into database

**3.2.4 Receptionist Access:**

FR10. The system should allow Receptionist to update, add or delete booking information

FR11.The system should provide customer desk portal access to receptionist for providing response to customer inquiry

**3.2.5 Manager Access:**

FR12. The system should generate financial and customer report for manager

FR13. The system should enable manager full modification access to customer, booking and room information

**3.2.6 Payment Management System:**

FR14. The system should allow customer to pay bill via online using credit or debit card

**3.3 Performance Requirements**

**NF1.** Data in database should be updated within 2 seconds.

**NF2.** Query results must return results within 5 seconds

**NF3.** Load time of UI Should not take more than 2 seconds

**NF4.** Login Validation should be done within 3 seconds

**NF5.** Response to customer inquiry must be done within 5 minutes.

**3.4 Security Requirements**

**NF6.** All external communications between the data’s server and client must be encrypted.

**NF7.** All data must be stored, protected or protectively marked. NF8. Payment Process should use HTTP over Secure protocol to secure the payment transactions.

**3.5 Safety Requirements**

**NF9.** Database should be backed up every hour.

**NF10.** Under failure, system should be able to come back at normal operation under an hour.

**3.6 Capacity Requirements**

**NF11.** Not more than 10,000 members to be registered

**NF12.** System need to handle at least 20 transactions during peak hours.

**3.7 Availability Requirements**

**NF13.** Report should be generated automatically every day for manager and anytime upon request.

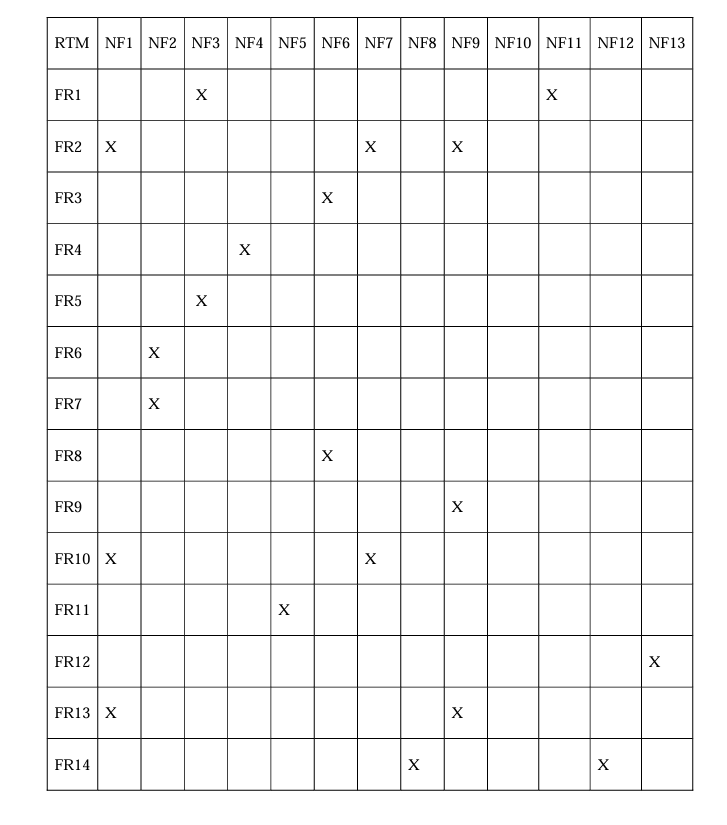
**3.8 Software System Attributes**

* **Correctness:** This system should satisfy the normal regular Hotel Management operations precisely to fulfill the end user objectives
* **Efficiency:** Enough resources to be implemented to achieve the particular task efficiently without any hassle.
* **Flexibility:** System should be flexible enough to provide space to add new features and to handle them conveniently
* **Integrity:** System should focus on securing the customer information and avoid data losses as much as possible
* **Portability:** The system should run in any Microsoft windows environment.
* **Usability:** The system should provide user manual to every level of users.
* **Testability:** The system should be able to be tested to confirm the performance and clients specifications.
* **Maintainability:** The system should be maintainable.

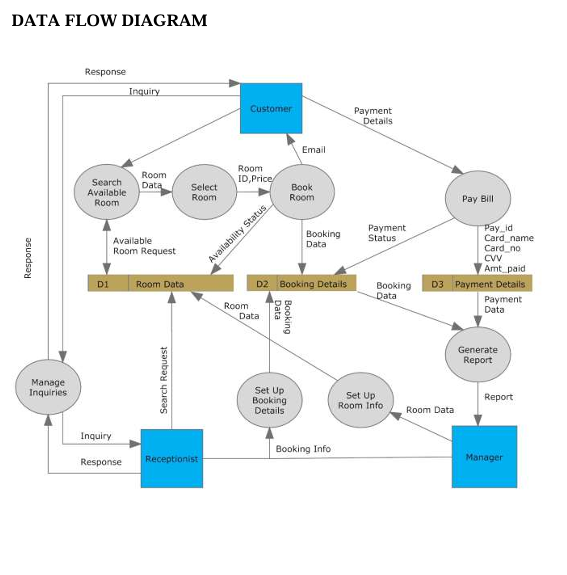
**3.9 Requirement Traceability Matrix**

The Requirement Traceability Matrix (RTM) reflects the correlation between Non Functional Requirements (NFR) and Functional Requirements (FR). The RTM is a documentation that associates the requirements entirely throughout the validation process. Traceability is regarded to be one of the most important considerations for tracing the requirements.

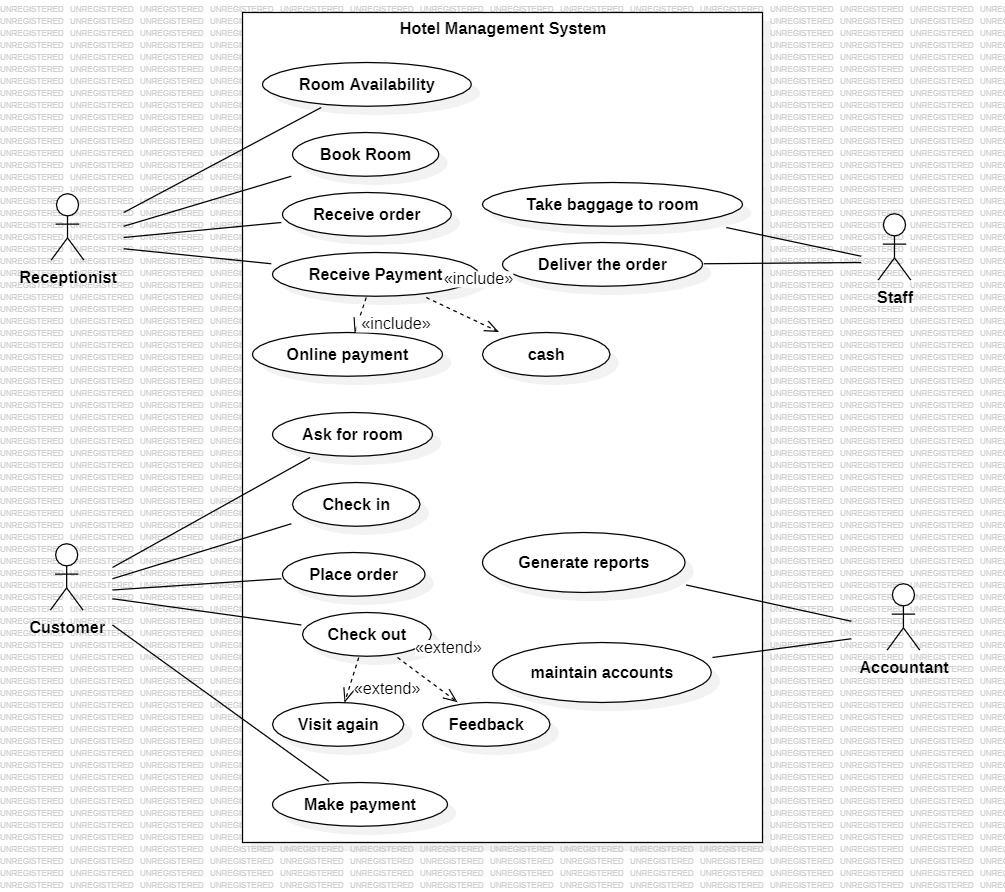
In the table below we will be tracing the relation between Functional Requirements and Non Functional Requirements.



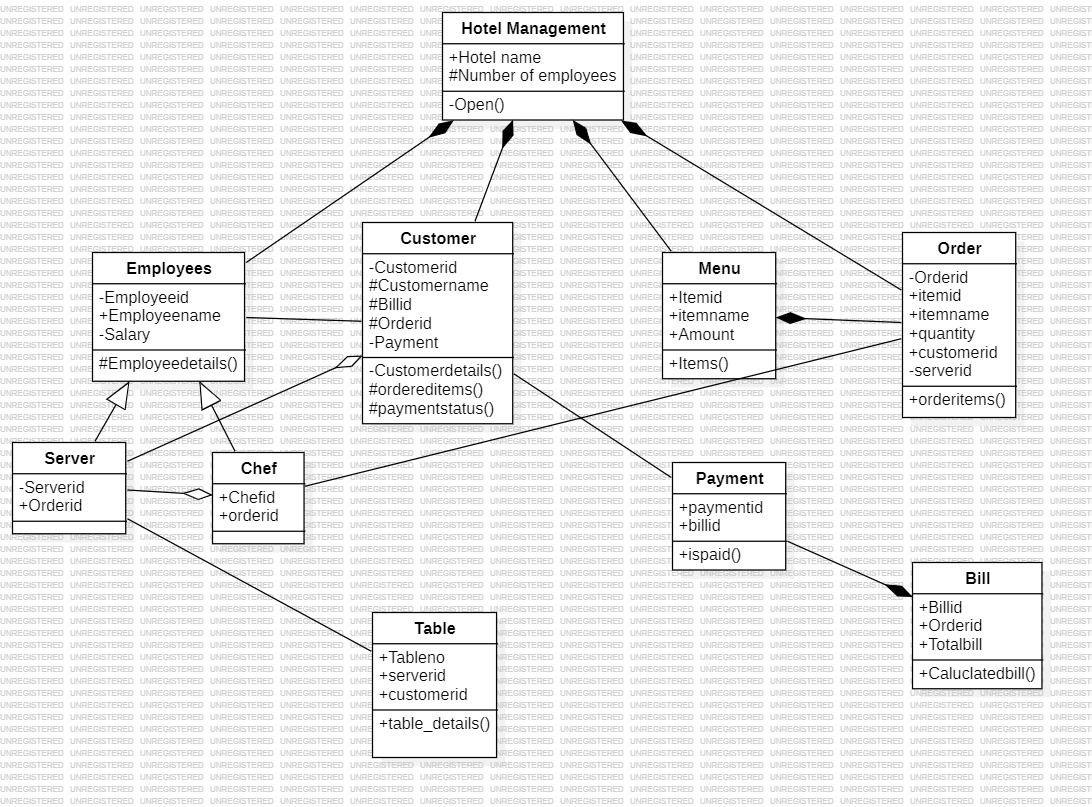
**Appendixes**

****

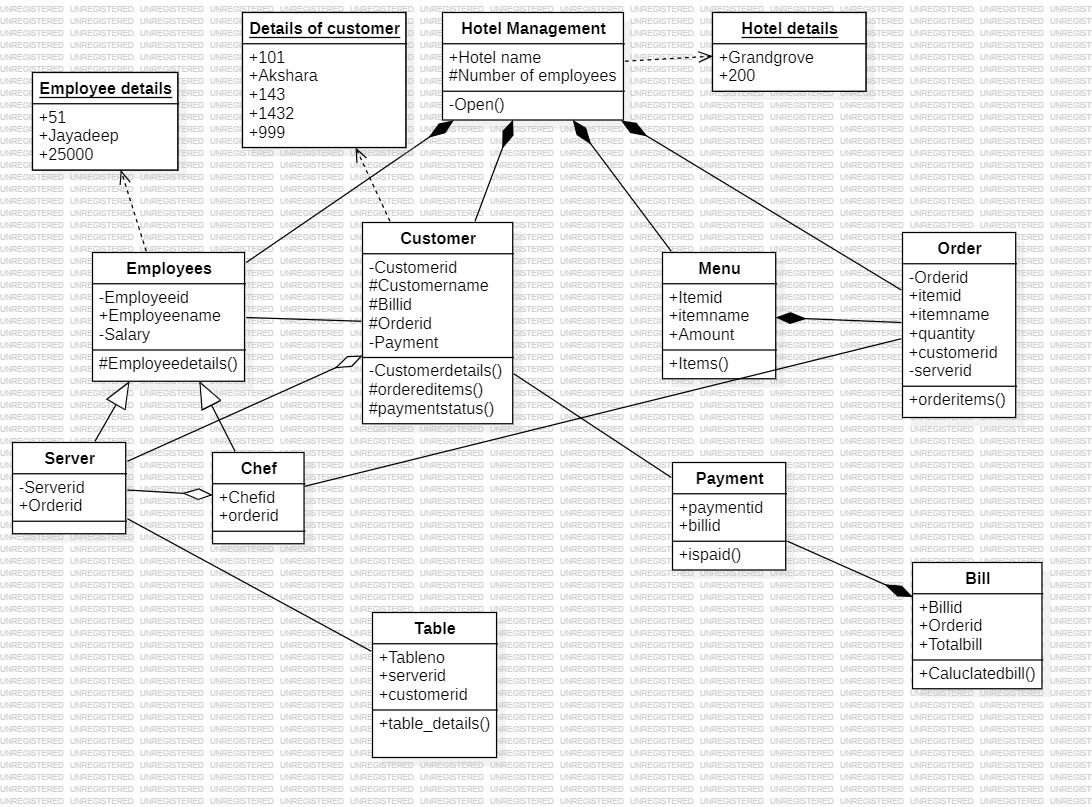
**Use Case Diagram:**



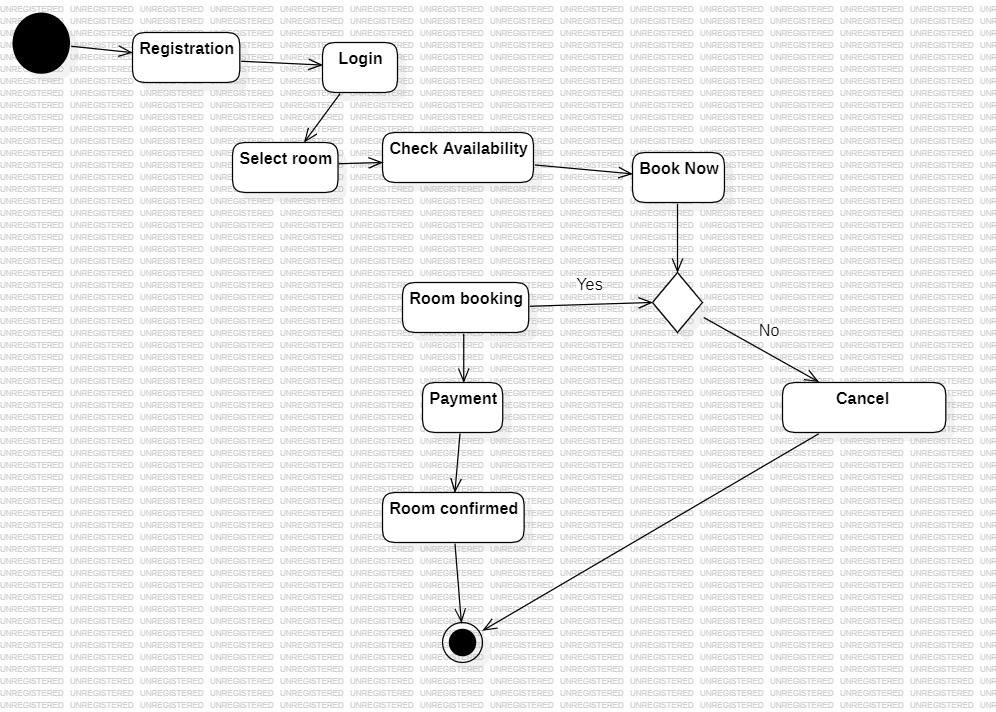
**Class diagram:**



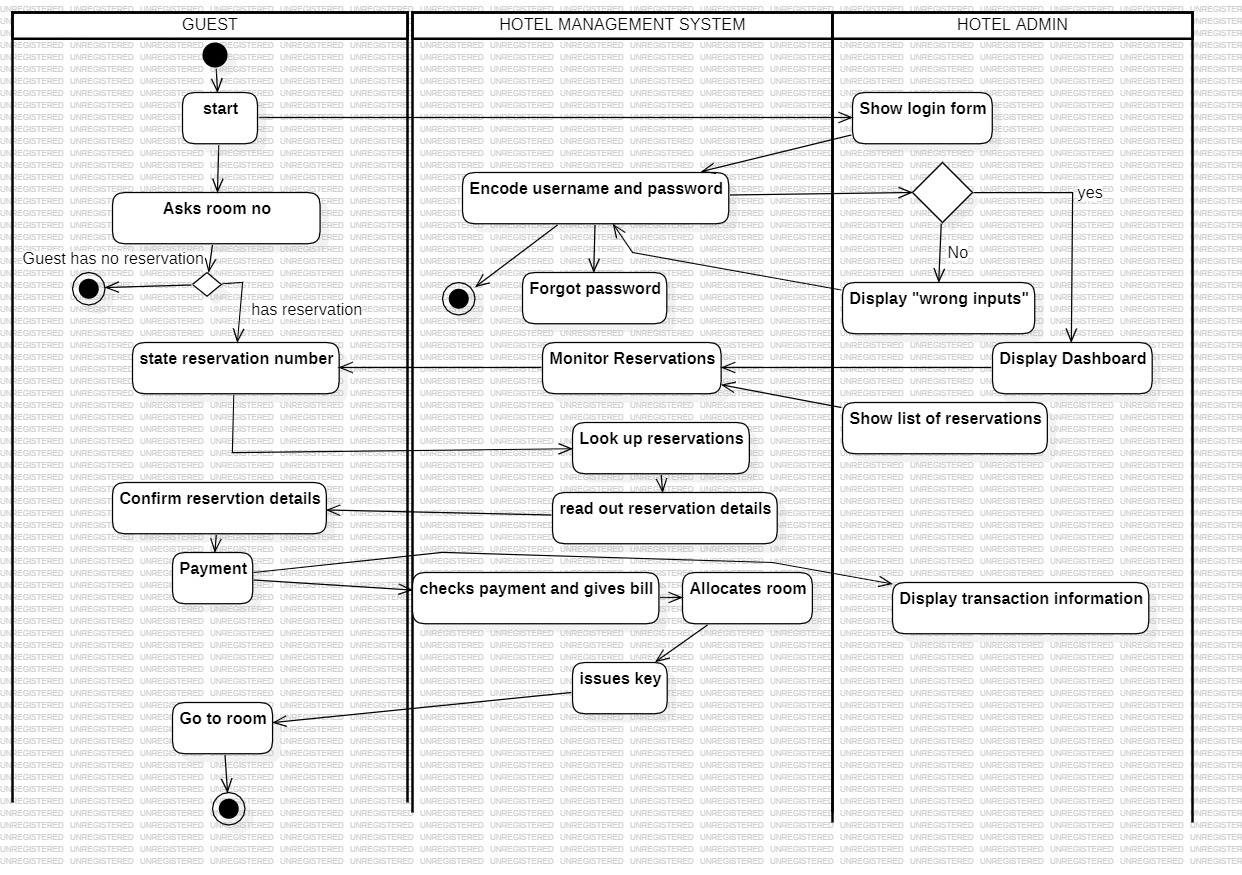
**Object diagram:**



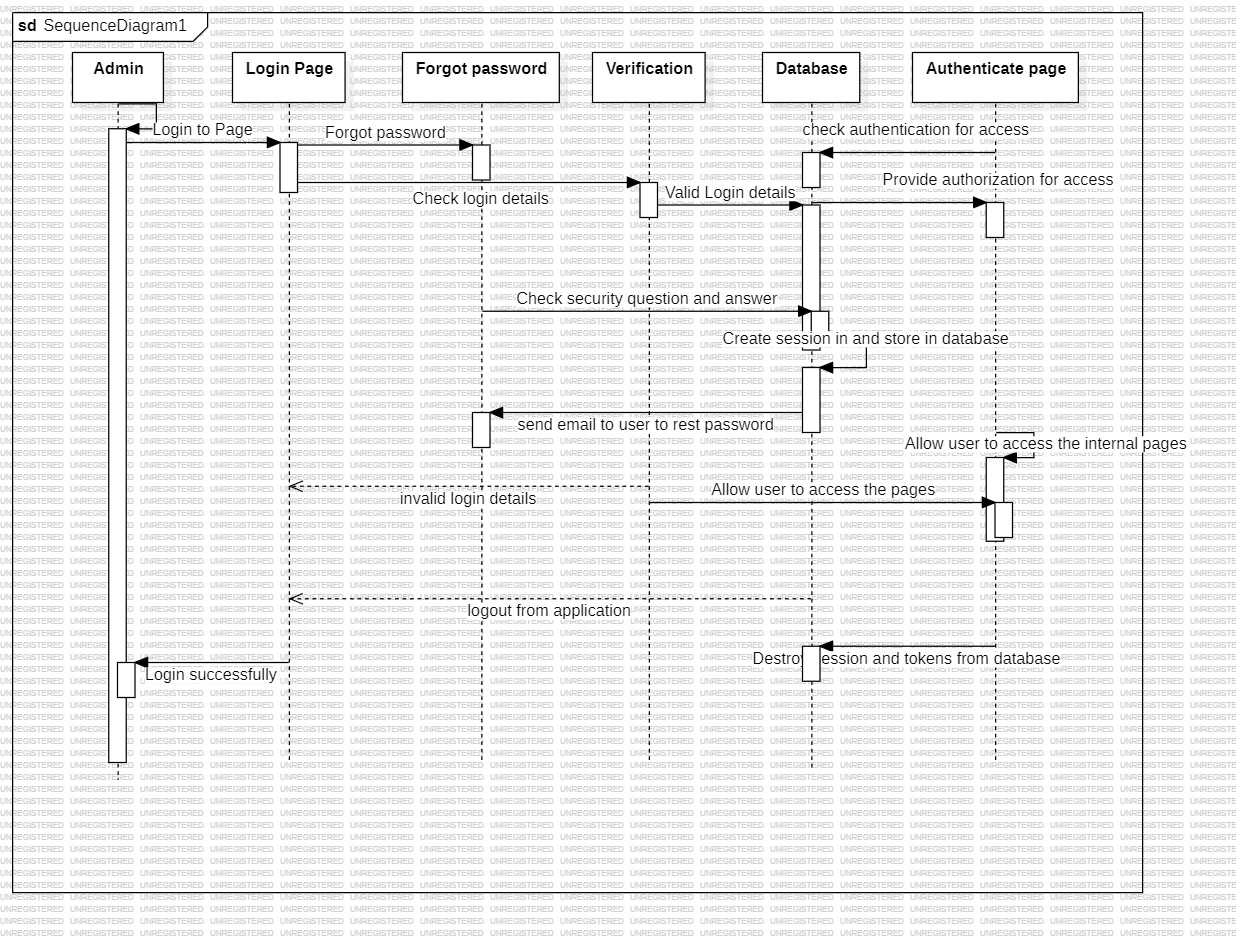
**State chart diagram:**



**Activity Diagram:**



**Sequence diagram:**



**Component Diagram:**

