Hotel Management System

Project Report

Submitted by

Sanjay Aaditya
Rick Brenton
Rohit Saravana

18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY

DEPARTMENT OF COMPUTING TECHNOLOGIES
FACULTY OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR - 603 203

NOVEMBER 2022

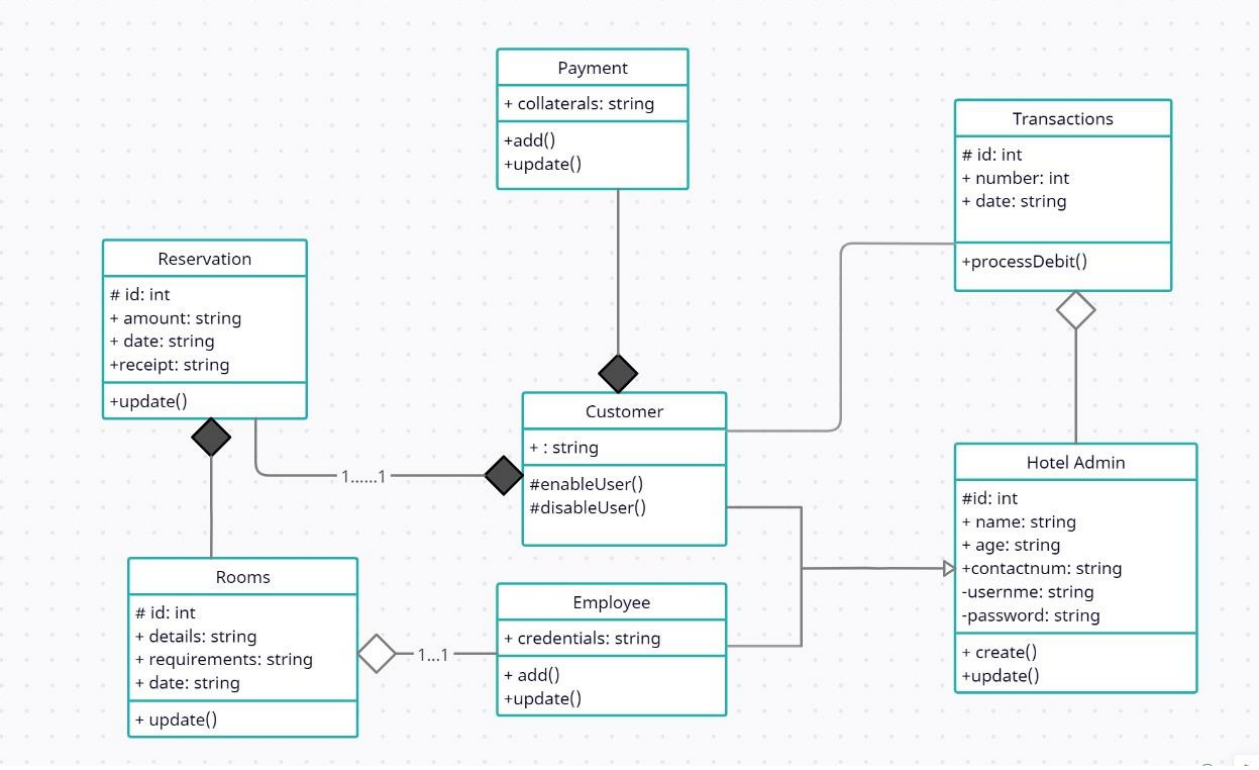
Hotel Management System

Hotel Management System: Here the customer opens the website of the hotel to reserve a room. The system takes in the request and checks the database for available vacancy. The customer is then given a Yes or No depending on the availability of the requested room. The Customer can then enter the payment details and confirm the reservation. The system verifies the payment information and outputs a Confirmation Status.

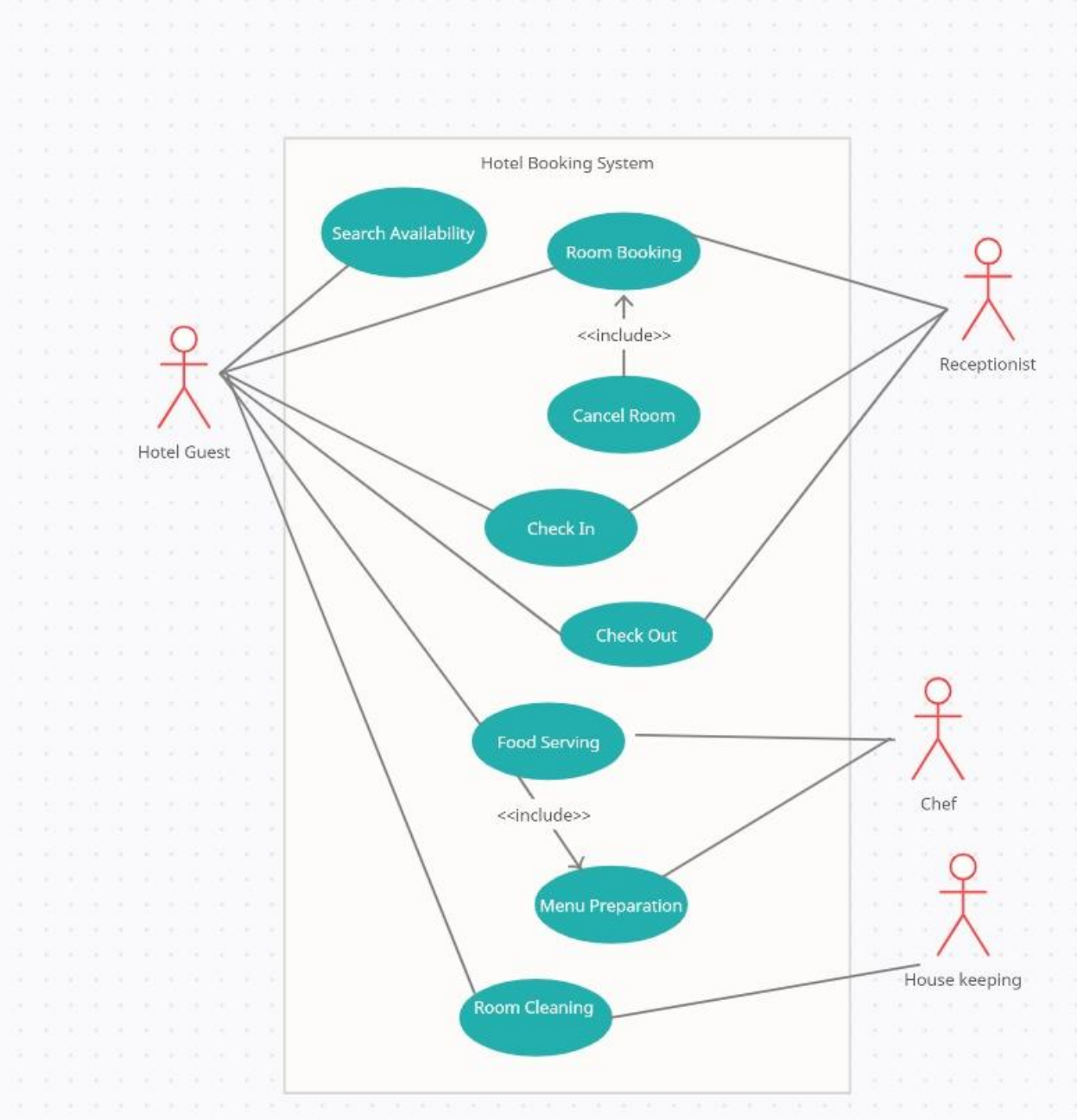
Required UML Diagrams:

- Class Diagram
- Use Case Diagram
- Sequence Diagram
- Collaboration Diagram
- State chart Diagram
- Activity Diagram
- Deployment Diagram
- Component Diagram
- Package Diagram

Class Diagram: Class diagrams describe the static structure of a system, or how it is structured rather than how it behaves.
These diagrams contain the following elements:
1. Classes, which represent entities with common characteristics or features. These features include attributes, operations, and associations.
2. Associations, which represent relationships that relate two or more other classes, Where the relationships have common characteristics or features. These features include attributes and operations.

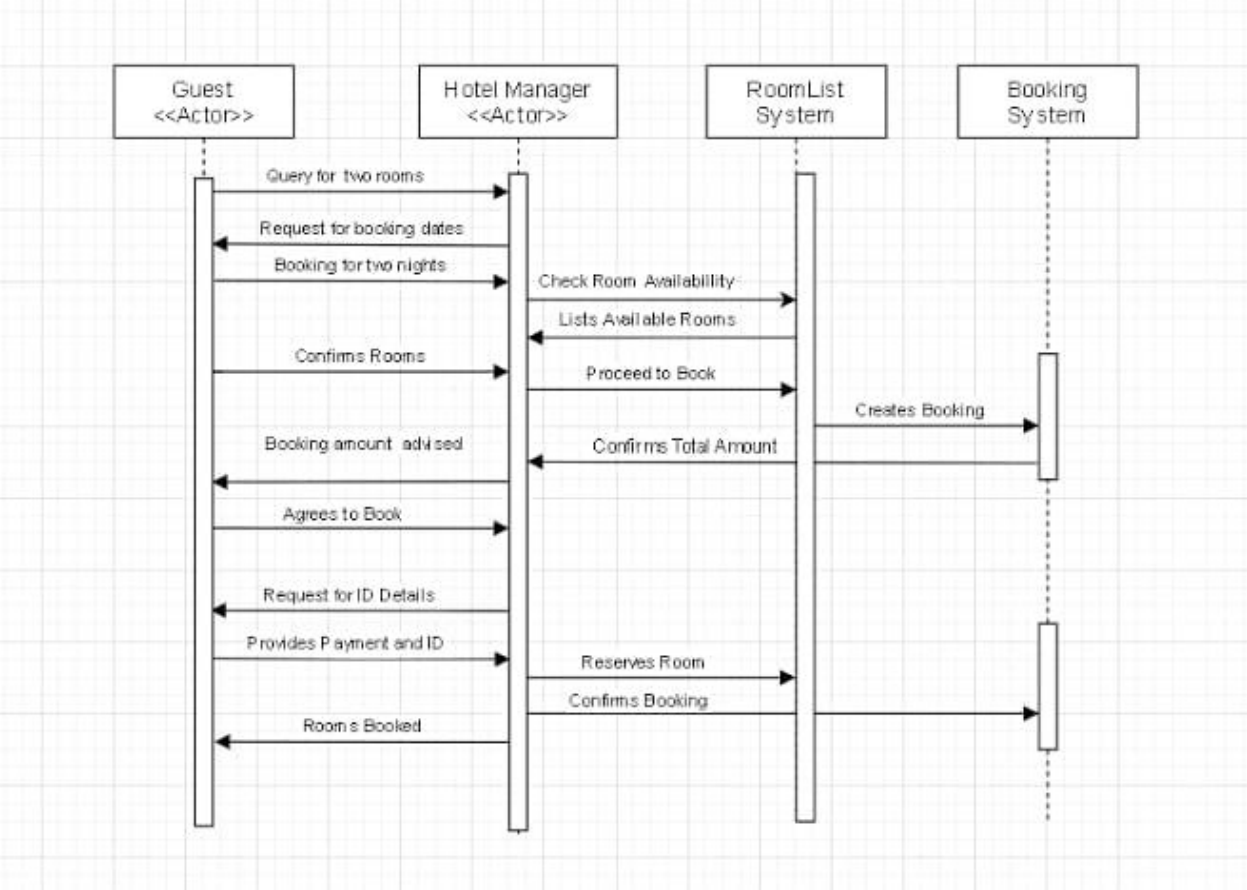


Use Case Diagram: Use case diagrams describe the functionality of a system and users of the system. They contain the
following elements:
1. Actors, which represent users of a system, including human users and other systems
2. Use cases, which represent functionality or services provided by a system to users

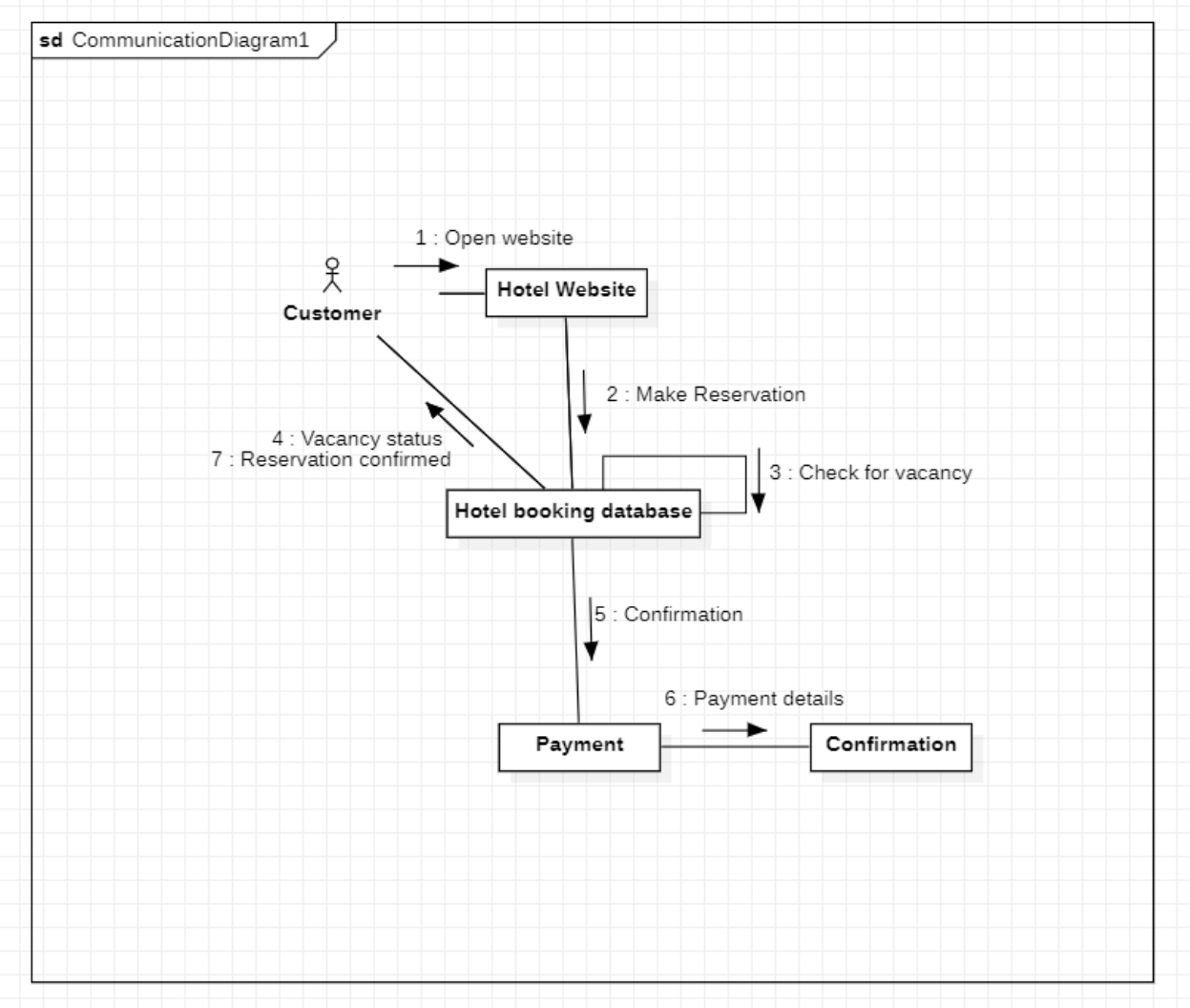

Hotel Booking System
Search Availability
Room Booking
<<include>>
Cancel Room
Check In
Check Out
Food Serving
<<include>>
Menu Preparation
Room Cleaning
Hotel Guest
Receptionist
Chef
House keeping

Sequence Diagram: Sequence diagrams typically show the flow of functionality through a use case, and consist of the following components:

1. Actors, involved in the functionality
2. Objects, that a system needs to provide the functionality
3. Messages, which represent communication between objects

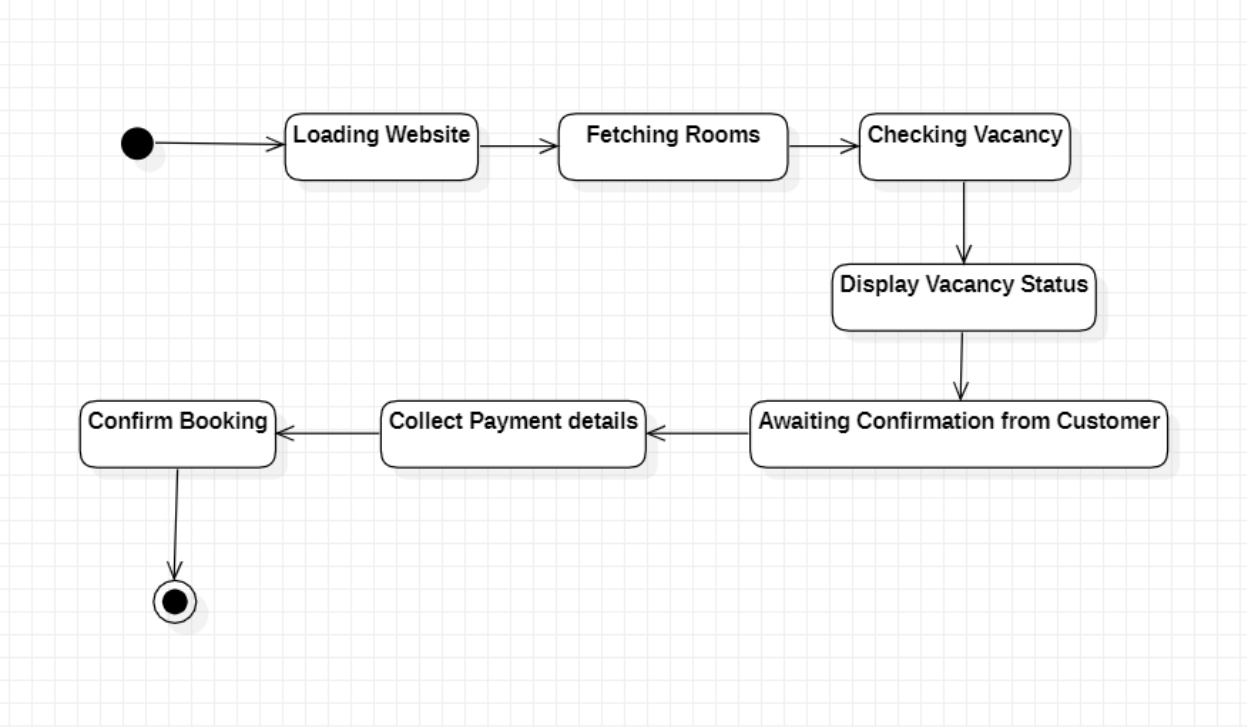


Communication/Collaboration Diagrams: A Communication or Collaboration diagram, as shown is a directed graph that uses objects and actors as graph nodes. The focus of the collaboration diagram is on the roles of the objects as they interact to realize a system function. Directional links are used to indicate communication between objects. These links are labelled using appropriate messages. Each message is prefixed with a sequence number indicating the time ordering needed to realize the system function.



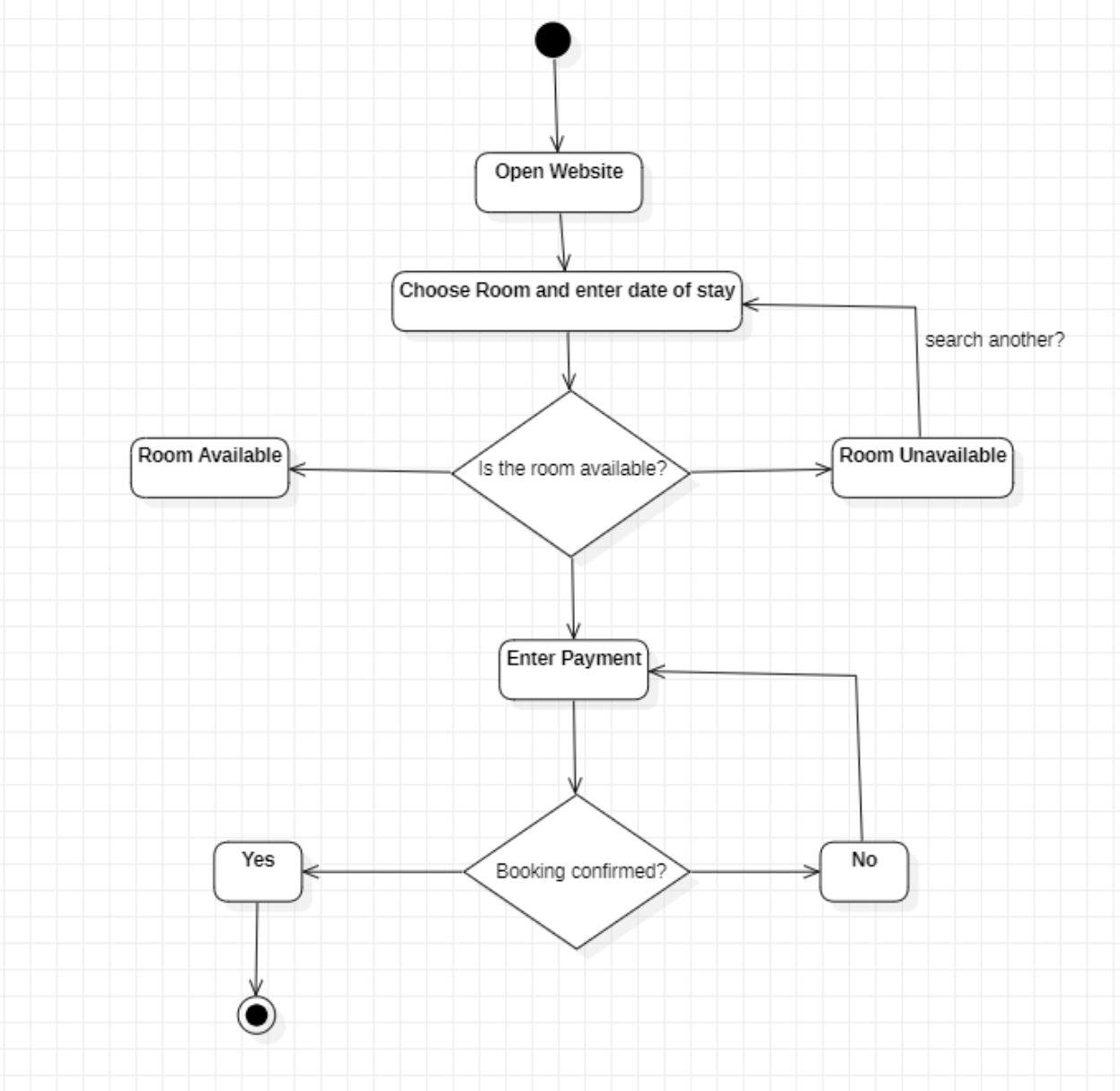
State Diagram: State transition diagrams provide a way to model the various states in which an object can exist. While the class diagram shows a static picture of the classes and their relationships, state transition diagrams model the dynamic behaviour of a system in response to external events (stimuli). State transition diagrams consist of the following:

1. States, which show the possible situations in which an object can find itself
2. Transitions, which show the different events which cause a change in the state of an object.



Activity Diagram: Activity diagrams describe the activities of a class. They are similar to state transition diagrams and use similar conventions, but activity diagrams describe the behaviour/states of a class in response to internal processing rather than external events. They contain the following elements:

1. Swim lanes, which delegate specific actions to objects within an overall activity
2. Action States, which represent uninterruptible actions of entities, or steps in the execution of an algorithm
3. Action Flows, which represent relationships between the different action states on an entity
4. Object Flows, which represent utilization of objects by action states, or influence of action states on objects.

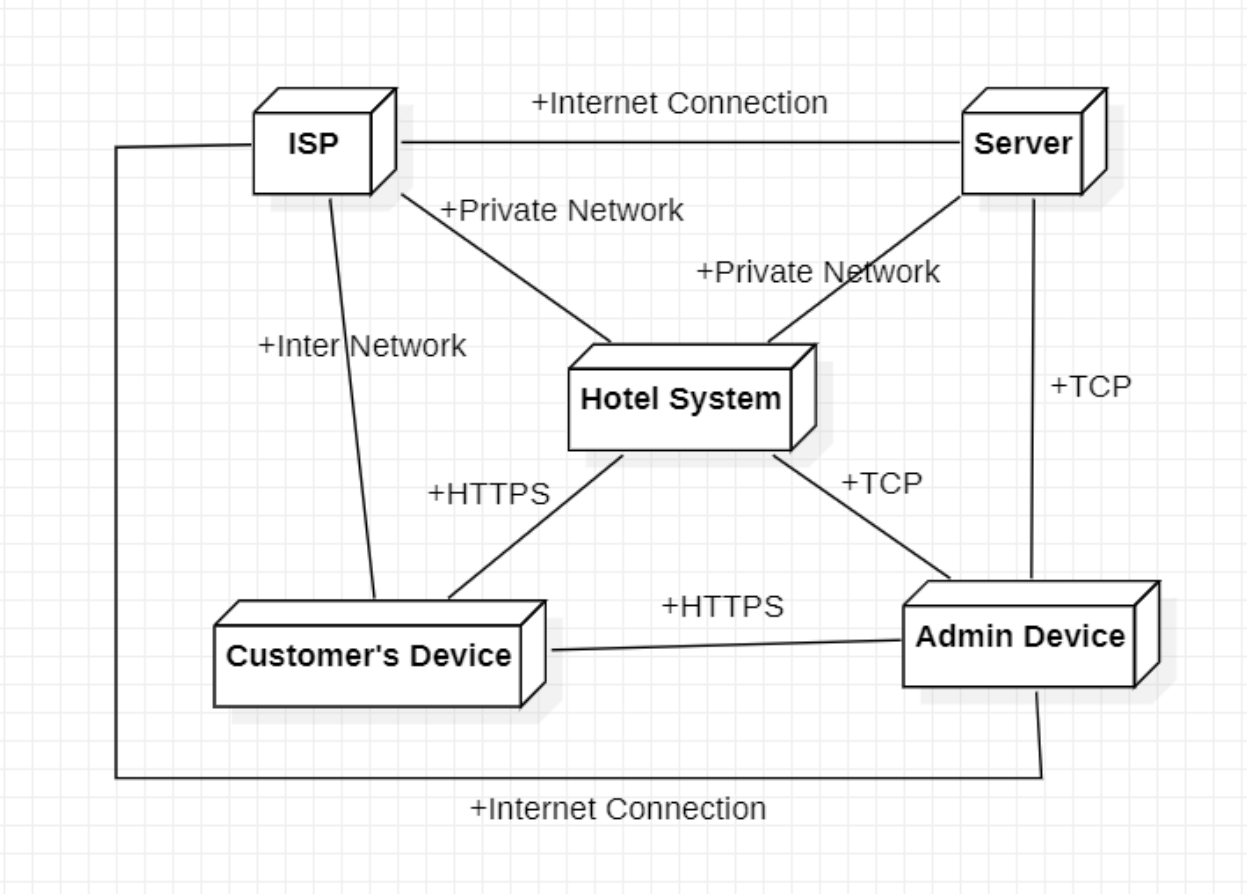


Deployment Diagram: The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships.

It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.

The deployment diagram consists of the following notations:

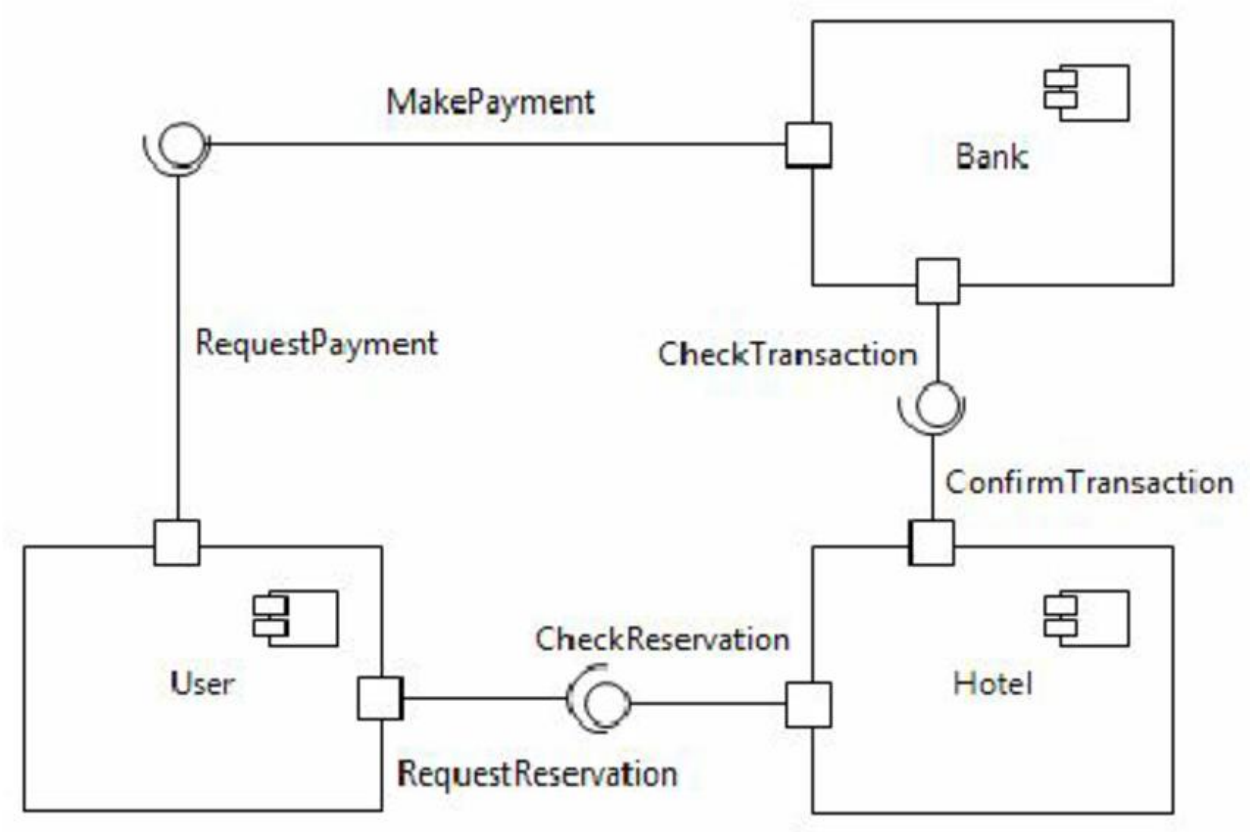
1. A component
2. An artifact
3. An interface
4. A node



Component Diagram: component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.
It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

The main purposes of the component diagram are:

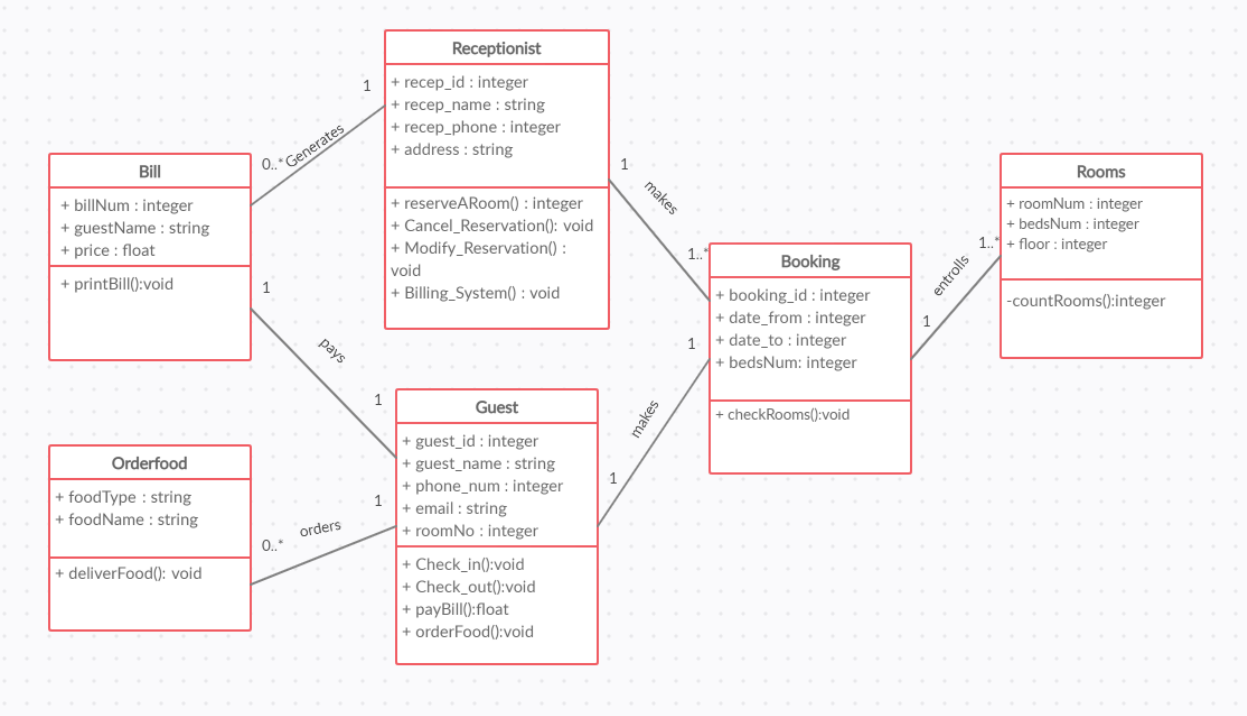
1. It envisions each component of a system.
2. It constructs the executable by incorporating forward and reverse engineering.
3. It depicts the relationships and organization of components.



Package Diagram: Package diagrams are structural diagrams used to show the organization and arrangement of various model elements in the form of packages. A package is a grouping of related UML elements, such as diagrams, documents, classes, or even other packages. Each element is nested within the package, which is depicted as a file folder within the diagram, then arranged hierarchically within the diagram. Package diagrams are most commonly used to provide a visual organization of the layered architecture within any UML classifier, such as a software system.

Uses of package diagram:

1. Simplify complex class diagrams and organize classes into packages
2. Define packages as file folders and use them on all of the UML diagrams
3. Define the hierarchical relationships (groupings) amongst packages as well as other packages or objects
4. Create a structure and visualize complex processes into simplified packages in technology, education and other fields, in order to visually depict non-linear processes.



Result: The UML Diagrams required for the proper execution of Hotel Management System have been created