## Hotel Management System

PROJECT REPORT

**18CSC202J/ 18AIC203J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY**

**(2018 Regulation)**

**II Year/ III Semester**

**Academic Year: 2022 -2023**

By

**Joeasm Dinesh C (RA2111003011485)**

**Shrinidhi S (RA2111003011488)**

Under the guidance of

**Dr. A DEVIPRIYA**

**Assistant Professor**

**Department of Computational Technologies**



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SCHOOL OF COMPUTING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**Kattankulathur, Kancheepuram**

**NOVEMBER 2022**

**BONAFIDE**

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY project report** titled “**SMART FARM** **MONITORING SYSTEM”** is the bonafide work of **JOESAM DINESH C (RA2111003011485) SHRINIDHI S (RA2111003011488)**

who undertook the task of completing the project within the allotted time.

**Signature of the Guide** **Signature of the II Year Academic Advisor**

Dr. A. Devipriya -------------------------

**Assistant Professor** **Professor and Head**

Department of CTECHL, Department of CTECHL

SRM Institute of Science and Technology SRM Institute of Science and Technology

**About the course:-**

18CSC202J/ 8AIC203J - Object Oriented Design and Programming are 4 credit courses with **L T P C as 3-0-2-4** (Tutorial modified as Practical from 2018 Curriculum onwards)

**Objectives:**

The student should be made to:

* Learn the basics of OOP concepts in C++
* Learn the basics of OOP analysis and design skills.
* Be exposed to the UML design diagrams.
* Be familiar with the various testing techniques

**Course Learning Rationale (CLR): The purpose of learning this course is to:**

1. Utilize class and build domain model for real-time programs
2. Utilize method overloading and operator overloading for real-time application development programs
3. Utilize inline, friend and virtual functions and create application development programs
4. Utilize exceptional handling and collections for real-time object-oriented programming applications
5. Construct UML component diagram and deployment diagram for design of applications
6. Create programs using object-oriented approach and design methodologies for real-time application development

**Course Learning Outcomes (CLO): At the end of this course, learners will be able to:**

1. Identify the class and build domain model
2. Construct programs using method overloading and operator overloading
3. Create programs using inline, friend and virtual functions, construct programs using

standard templates

1. Construct programs using exceptional handling and collections
2. Create UML component diagram and deployment diagram
3. Create programs using object oriented approach and design methodologies

**Table 1: Rubrics for Laboratory Exercises**

(Internal Mark Splitup:- As per Curriculum)

|  |  |  |
| --- | --- | --- |
| **CLAP-1** | 5=(2(E-lab Completion) + 2(Simple Exercises)( from CodeZinger, and any other coding platform) + 1(HackerRank/Code chef/LeetCode Weekend Challenge) | Elab test |
| **CLAP-2** | 7.5=(2.0(E-lab Completion)+  2.0 (Simple Exercises)( from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge) | Elab test |
| **CLAP-3** | 7.5=(2.0(E-lab Completion(80 Pgms)+  2.0 (Simple Exercises)( from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge) | **2 Mark -** E-lab Completion **80 Program** Completion from 10 Session (Each session min 8 program)  **2 Mark -** Code to UML conversion GCR Exercises  **3.5 Mark - Hacker Rank** Coding challenge completion |
| **CLAP-4** | 5= 3 ( Model Practical) + 2( Oral Viva) | * **3 Mark** – Model Test * **2 Mark** – Oral Viva |
| **Total** | 25 |  |

**COURSE ASSESSMENT PLAN FOR OODP LAB**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No** | **List of Experiments** | **Course Learning Outcomes (CLO)** | **Blooms Level** | **PI** | **No of Programs in each session** |
| 1. | Implementation of I/O Operations in C++ | CLO-1 | Understand | 2.8.1 | 10 |
| 2. | Implementation of Classes and Objects in C++ | CLO-1 | Apply | 2.6.1 | 10 |
| 3, | To develop a problem statement. 1. From the problem statement, Identify Use Cases and develop the Use Case model. 2. From the problem statement, Identify the conceptual classes and develop a domain model with a UML Class diagram. | CLO-1 | Analysis | 4.6.1 | Mini Project Given |
| 4. | Implementation of Constructor Overloading and Method Overloading in C++ | CLO-2 | Apply | 2.6.1 | 10 |
| 5. | Implementation of Operator Overloading in C++ | CLO-2 | Apply | 2.6.1 | 10 |
| 6. | Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams and Collaboration diagrams | CLO-2 | Analysis | 4.6.1 | Mini Project Given |
| 7. | Implementation of Inheritance concepts in C++ | CLO-3 | Apply | 2.6.1 | 10 |
| 8. | Implementation of Virtual function & interface concepts in C++ | CLO-3 | Apply | 2.6.1 | 10 |
| 9. | Using the identified scenarios in your project, draw relevant state charts and activity diagrams. | CLO-3 | Analysis | 4.6.1 | Mini Project Given |
| 10. | Implementation of Templates in C++ | CLO-3 | Apply | 2.6.1 | 10 |
| 11. | Implementation of Exception of Handling in C++ | CLO-4 | Apply | 2.6.1 | 10 |
| 12. | Identify the User Interface, Domain objects, and Technical Services. Draw the partial layered, logical architecture diagram with UML package diagram notation such as Component Diagram, Deployment Diagram. | CLO-5 | Analysis | 4.6.1 | Mini Project Given |
| 13. | Implementation of STL Containers in C++ | CLO-6 | Apply | 2.6.1 | 10 |
| 14. | Implementation of STL associate containers and algorithms in C++ | CLO-6 | Apply | 2.6.1 | 10 |
| 15. | Implementation of Streams and File Handling in C++ | CLO-6 | Apply | 2.6.1 | 10 |

**LIST OF EXPERIMNENTS FOR UML DESIGN AND MODELLING:**

**To develop a mini-project by following the exercises listed below.**

1. To develop a problem statement.

2. Identify Use Cases and develop the Use Case model.

3. Identify the conceptual classes and develop a domain model with UML Class diagram.

4. Using the identified scenarios, find the interaction between objects and represent them

using UML Sequence diagrams.

5. Draw relevant state charts and activity diagrams.

6. Identify the User Interface, Domain objects, and Technical services. Draw the partial

layered, logical architecture diagram with UML package diagram notation.

**Suggested Software Tools for UML:**

StarUML, Rational Suite, Argo UML (or) equivalent, Eclipse IDE and Junit

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

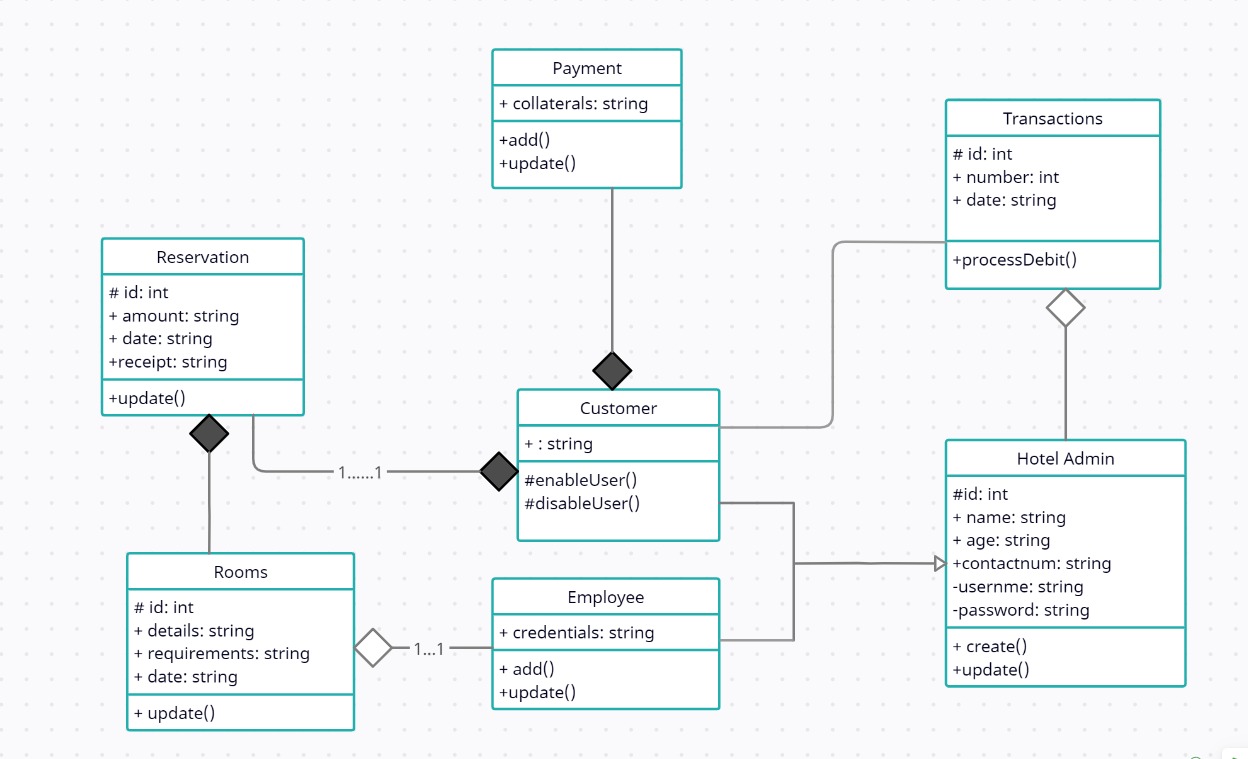
structured rather than how it behaves.

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.



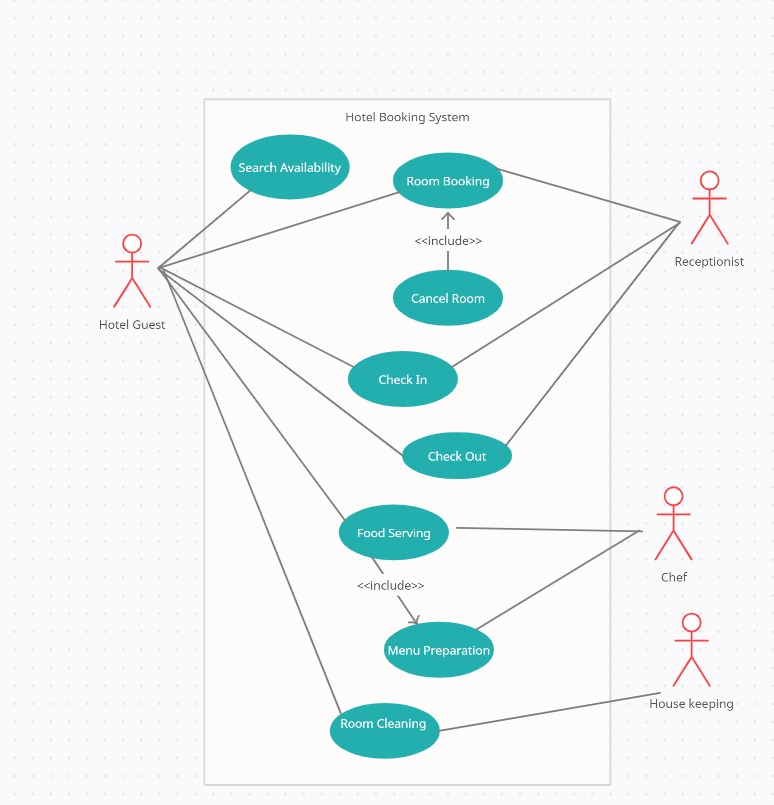
**Use Case Diagram:** Use case diagrams describe the functionality of a system and users

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



**Sequence Diagram:** Sequence diagrams typically show the flow of functionality through

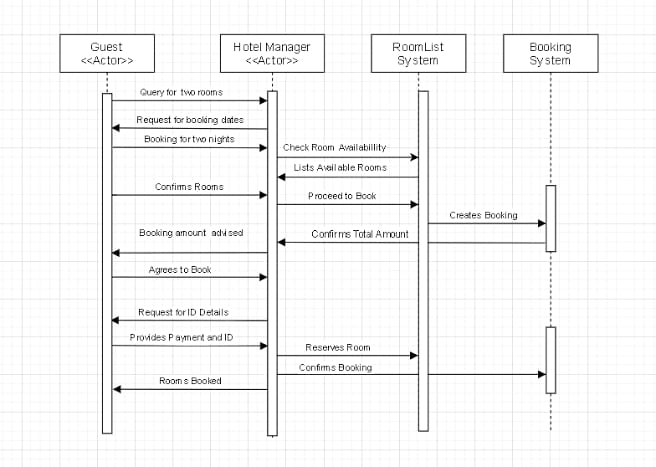
a use case, and consist of the following

components:

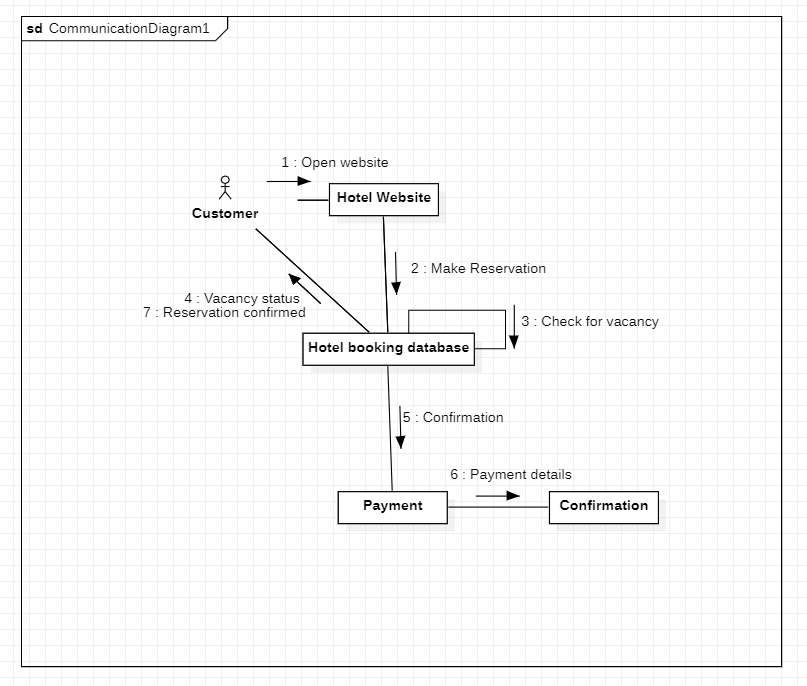
1. Actors, involved in the functionality

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

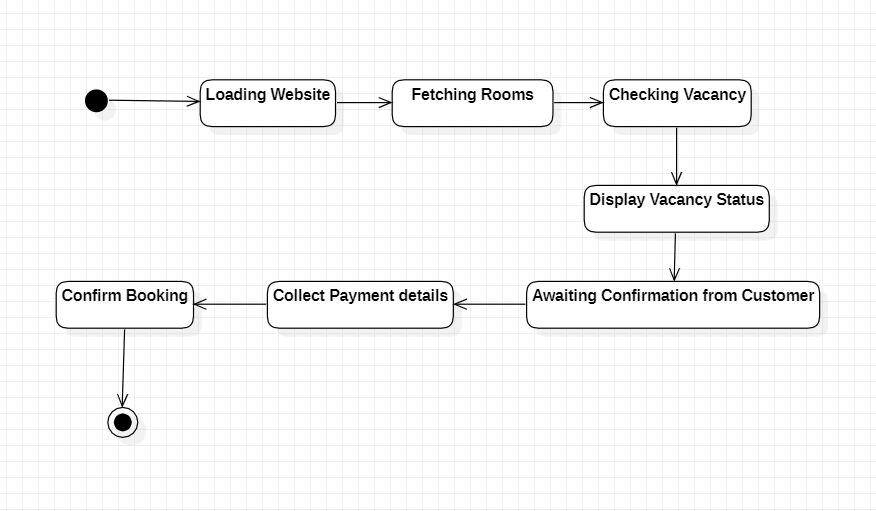
their relationships, state transition diagrams model the dynamic behaviour of a system in

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

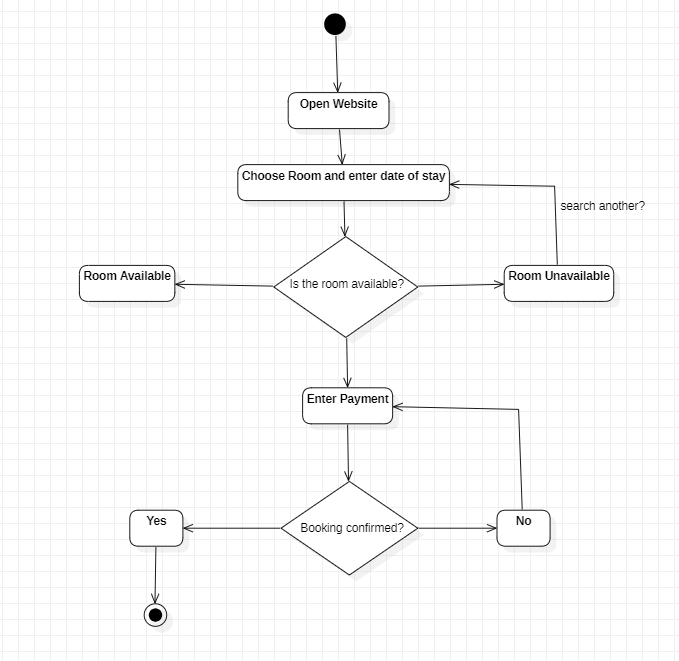
execution of an algorithm

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.

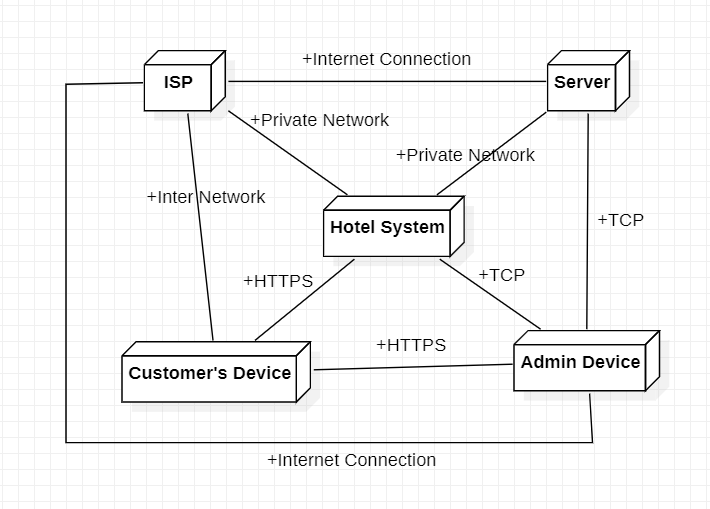


**Deployment Diagram:** The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships.

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

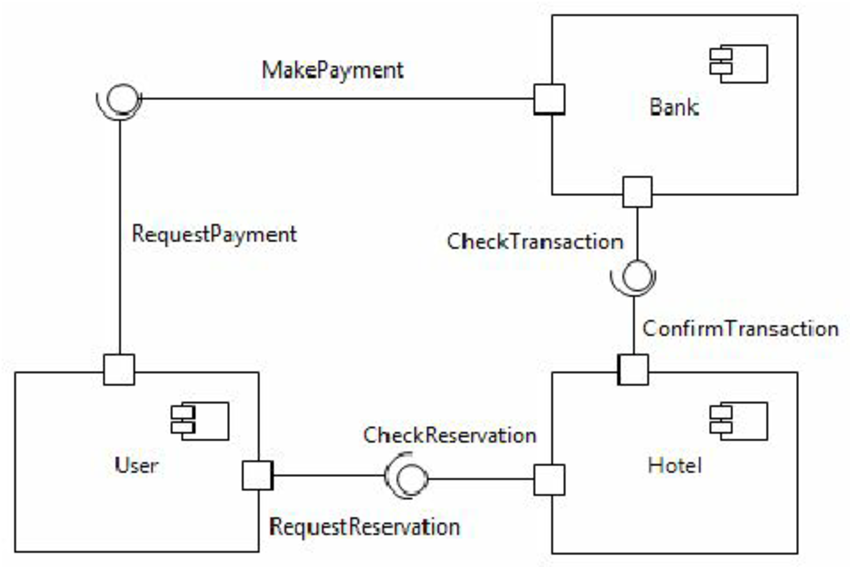


**Component Diagram**: component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

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:

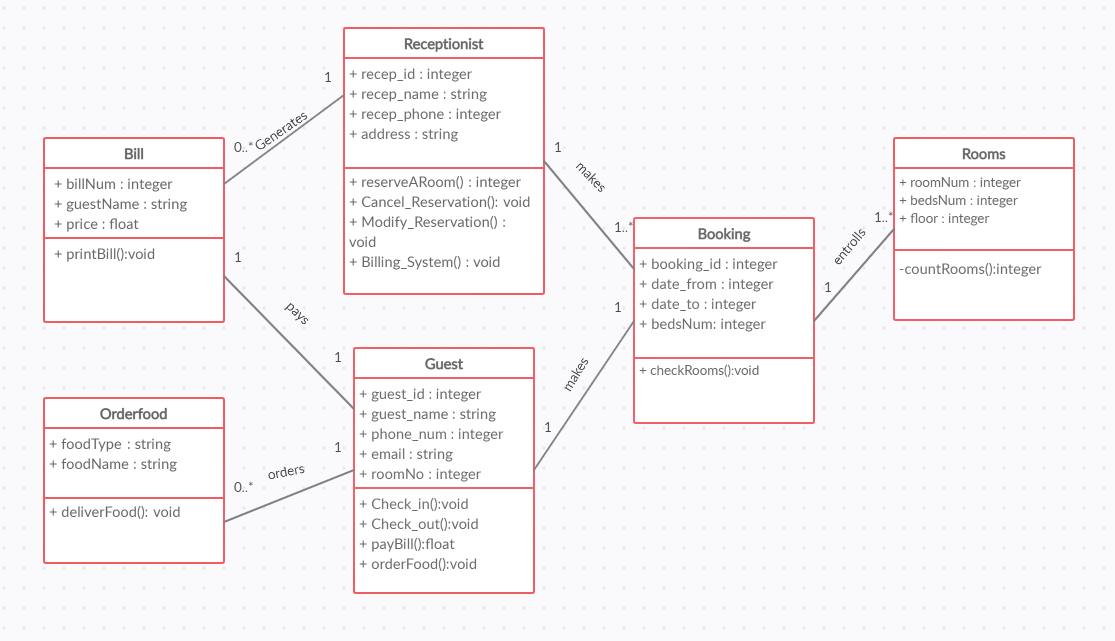
1. It envisions each component of a system.
2. It constructs the executable by incorporating forward and reverse engineering.
3. It depicts the relationships and organization of components.



**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