DEPARTMENT OF NETWORKING AND COMMUNICATIONS

FACULTY OF ENGINEERING & TECHNOLOGY

MINI PROJECT

SUBJECT CODE: 18CSC202J

SUBJECT TITLE: OBJECT ORIENTED DESIGN AND PROGRAMMING

PROJECT TITLE

RESTAURANT MANAGEMENT SYSTEM

BY

TEAM MEMBERS

ARNAV AGGARWAL

NAVDEEP SINGH JAKHAR

SRM University, SRM Nagar, Kattankulathur-603203

Chengalpattu District, Tamil Nadu

Rubrics

Experiment Component	Max. Marks	Grading Rubrics		
Documentation/ Procedure	10	UML Diagrams are well documented. The documentation supporting all functional requirement and non-functional requirement (10 Marks)	Missing two or more required functional requirement. The documentation work not up to the mark. (5 Mark)	
Concept	5	Completeness of concept, consistent variable naming and relationship in static view. (5 Marks)	Completeness of concept, inconsistent variable naming and relationship in static view. (3 Marks)	Incomplete static view. (1 Mark)
Usage of Symbols	3	Precise usage of symbols in dynamic view. (3 Marks)	Improper usage of Symbol's. (2 Marks)	Symbol's misplaced in diagram. (1 Mark)
Diagrams	4	Completion of all 8 UML Diagrams using Visual Paradigm Tool. (4 Marks)	Construction of UML Diagrams using other tools. (2 Marks)	Construction of few diagrams. (1 Mark)
Viva and Innovative Idea	3	Oral Viva and Innovative approach. (3 Marks)	Oral Viva and partial idea. (2 Marks)	Oral Viva not fulfilled. (1 Mark)
TOTAL	25			

BONAFIDE

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY project report** titled "**RESTAURANT MANAGEMENT SYSTEM"** is the bonafide work of **ARNAV AGGARWAL (RA2111032010002), NAVDEEP SINGH JAKHAR (RA2111032010030)** who undertook the task of completing the project within the allotted time.

Signature of the Guide

Dr. Gouthaman. P

Assistant Professor

Department of NWC

SRM Institute of Science and Technology

Signature of the II Year Academic Advisor

Professor and Head

Department of NWC

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
4. Construct programs using exceptional handling and collections 5. Create UML component diagram and deployment diagram
6. Create programs using object-oriented approach and design methodologies

LIST OF EXPERIMENTS 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

TABLE OF CONTENTS

Chapter No.	Title	Page No.
	Rubrics	ii
	About the Course	iv
	List of Experiments	v
	Abstract	1
	Module Description	2
1	Use Case & Class Case Diagram	3
2	Sequence & Communication Diagram	4
3	State Chart & Activity Diagram	5
4	Package, Component & Deployment Diagram	6
	Conclusion	7
	References	8

ABSTRACT

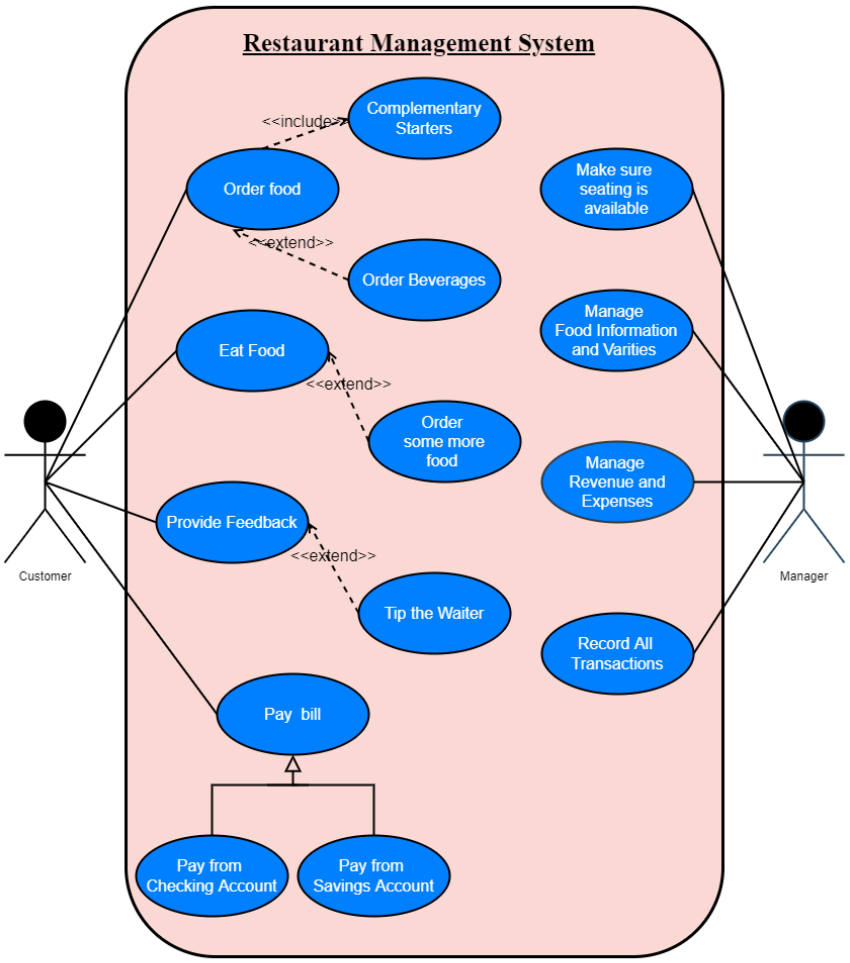
A restaurant introduces an information system to manage the various essential processes involved in running a restaurant. In the restaurant, a lot of administrative and managerial processes are involved to satisfy even a single customer and provide a customer with a comfortable experience. These diagrams focus on the responsibilities of the manager and the actions of a customer and the manager. When a customer arrives, the first process that takes place is getting their details to ensure that the suggestions and the service provided to them is accurate and suitable for them. The customer's next task is to place an order and wait for the food, whereas, a manager's work is going on simultaneously behind the scenes. The duties of a manager include managing the customers' information, managing the food menu, keeping track of the orders, managing expenses, and recording all transactions. Only when all of these processes are completed efficiently and in a timely manner does the customer leave with a seamless and enjoyable experience.

MODULE DESCRIPTION

This Restaurant Management System project is a management system that produces user-friendly, quick, and cost-effective software. It handles and secures customer and kitchen information, financial transactions, and so on. It is the Restaurant manager's job to register and maintain customer and kitchen information and to access and update the information related to the menu and the raw materials needed. Customer information and their order are entered into the system, then the output is used to display these details on the screen. The order details can be accessed by customers, managers, and chefs but the customer information is only visible to the manager. The data is well-protected, and the data processing is quick. The software is user-friendly, quick, and cost-effective.

CHAPTER 1
USE CASE DIAGRAM

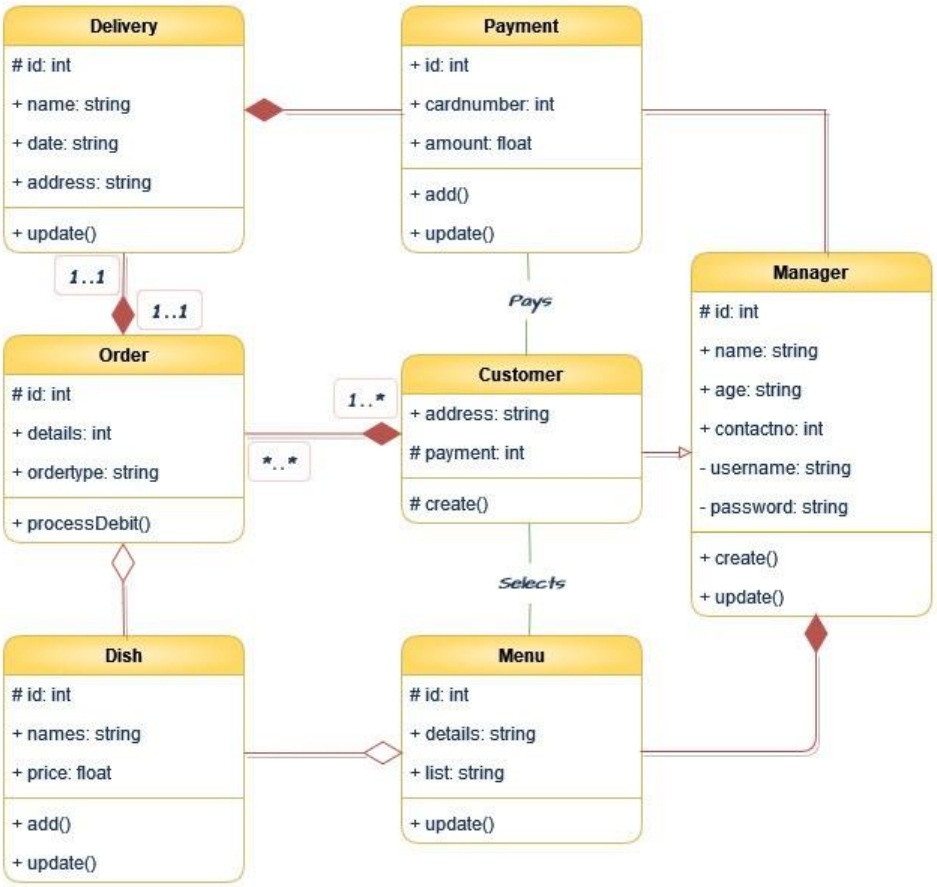
The Use Case Diagram for restaurant management system shows the general processes of the system. These processes involve managing restaurant activities as well as their customers' orders and payments.This Use Case Diagram is a graphic depiction of the interactions among the elements of the Restaurant Management System. It represents the methodology used in system analysis to identify, clarify, and organize system requirements.The use case diagram using include and extend is used to elaborate the proceeding diagrams. The terms include and extend in the use case diagram are known as indicators.



The main actors in this Use Case Diagram are the Customer and the Manager, who perform different types of tasks such as ordering and eating food, paying the bill, managing information, and recording transactions.

CLASS DIAGRAM

A class diagram shows a static view of a system. It includes the classes, or types of objects, that will exist in the system. It also shows the attributes, or characteristics, of these classes, and the methods, or the way that each class will interact with other classes. A Class Diagram describes the structure of Restaurant Management System classes, their attributes, methods, and the relationships among various objects. A class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The main classes of the Restaurant Management System are Manager, Customer, Order, Delivery, and Order, etc.

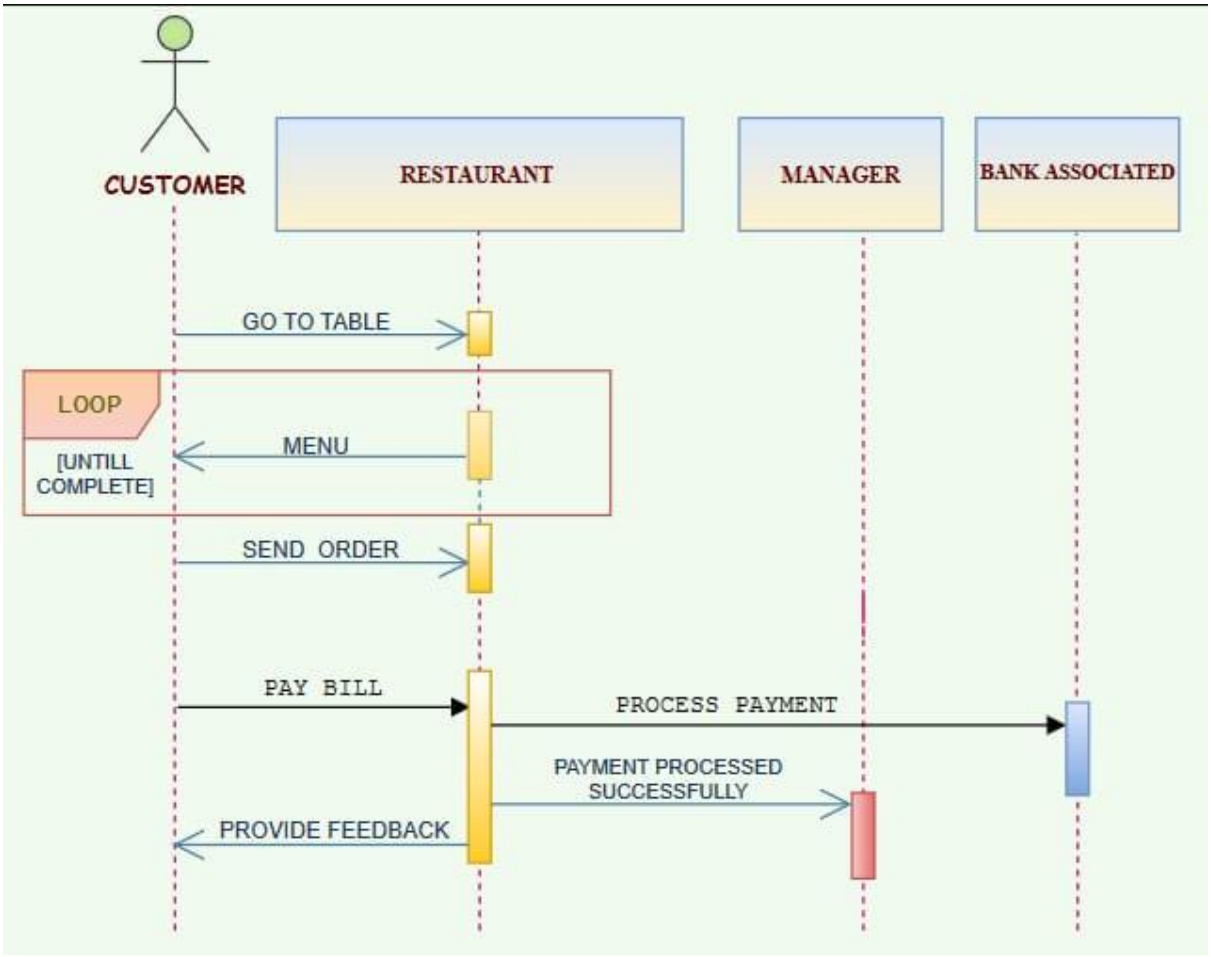


A Class Diagram describes the structure of Restaurant Management System classes, their attributes, methods, and the relationships among various objects. A class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The main classes of the Restaurant Management System are Manager, Customer, Order, Delivery, and Order, etc.

CHAPTER 2

SEQUENCE DIAGRAM

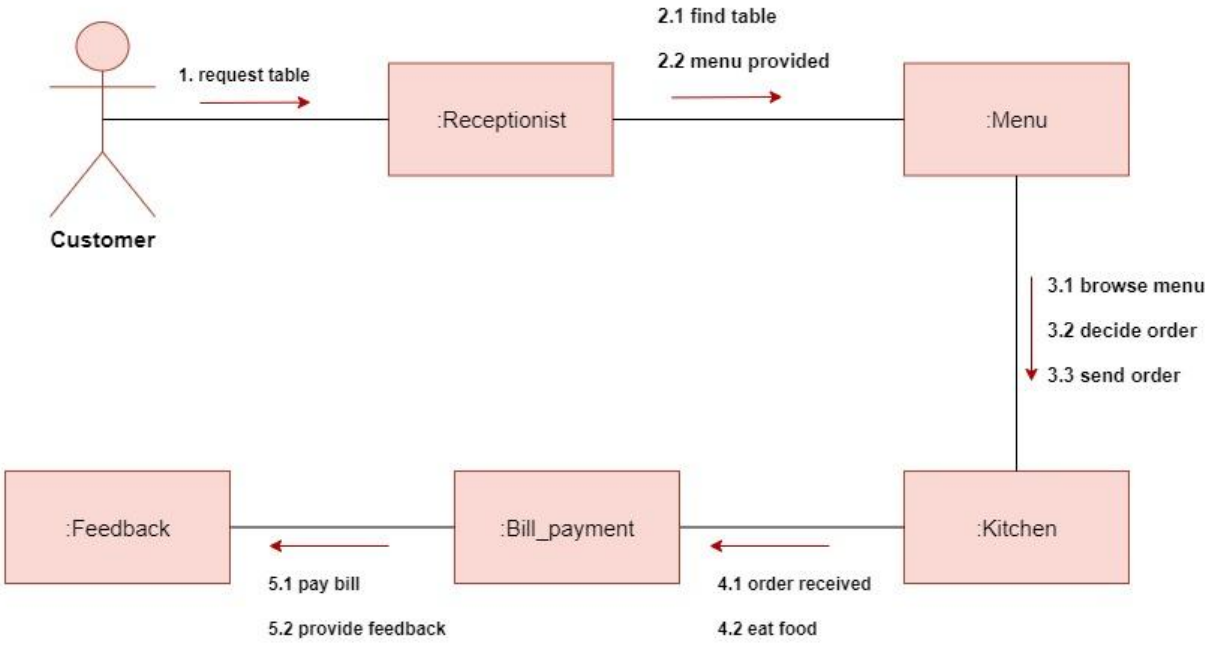
A sequence diagram is the most commonly used interaction diagram.An interaction diagram is used to show the interactive behavior of a system. Since visualizing the interactions in a system can be a cumbersome task, we use different types of interaction diagrams to capture various features and aspects of interaction in a system.A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function.



This UML Diagram for the Restaurant Management System shows the interaction between the customer, restaurant, manager, and bank

COMMUNICATION DIAGRAM

Communication diagrams, formerly known as collaboration diagrams, are almost identical to sequence diagrams in UML, but they focus more on the relationships of objects—how they associate and connect through messages in a sequence rather than interactions.A communication diagram shows the interactions between the objects or roles associated with lifelines and the messages that pass between lifelines. Communication diagrams are a type of interaction diagram that you can use to explore the dynamic behavior of a system or software application.

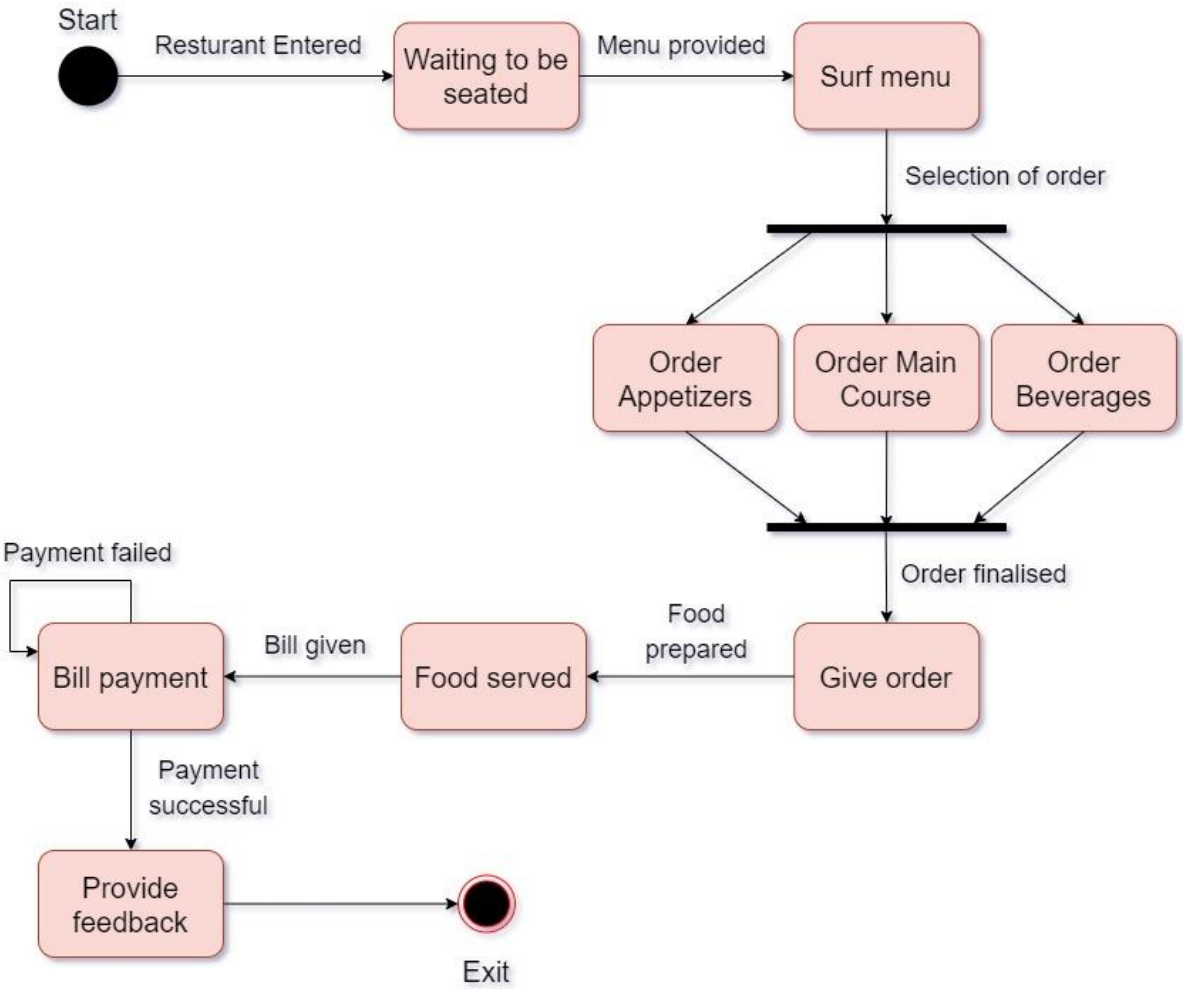


They provide an alternate view of the same information as sequence diagrams. In sequence diagrams, the focus is the ordering of the messages over time; in communication diagrams, the focus is the structure of the messages that pass between the objects in the interaction. These diagrams illustrate the flow of messages between objects and the implied relationships between classes.

CHAPTER 3

STATE CHART DIAGRAM

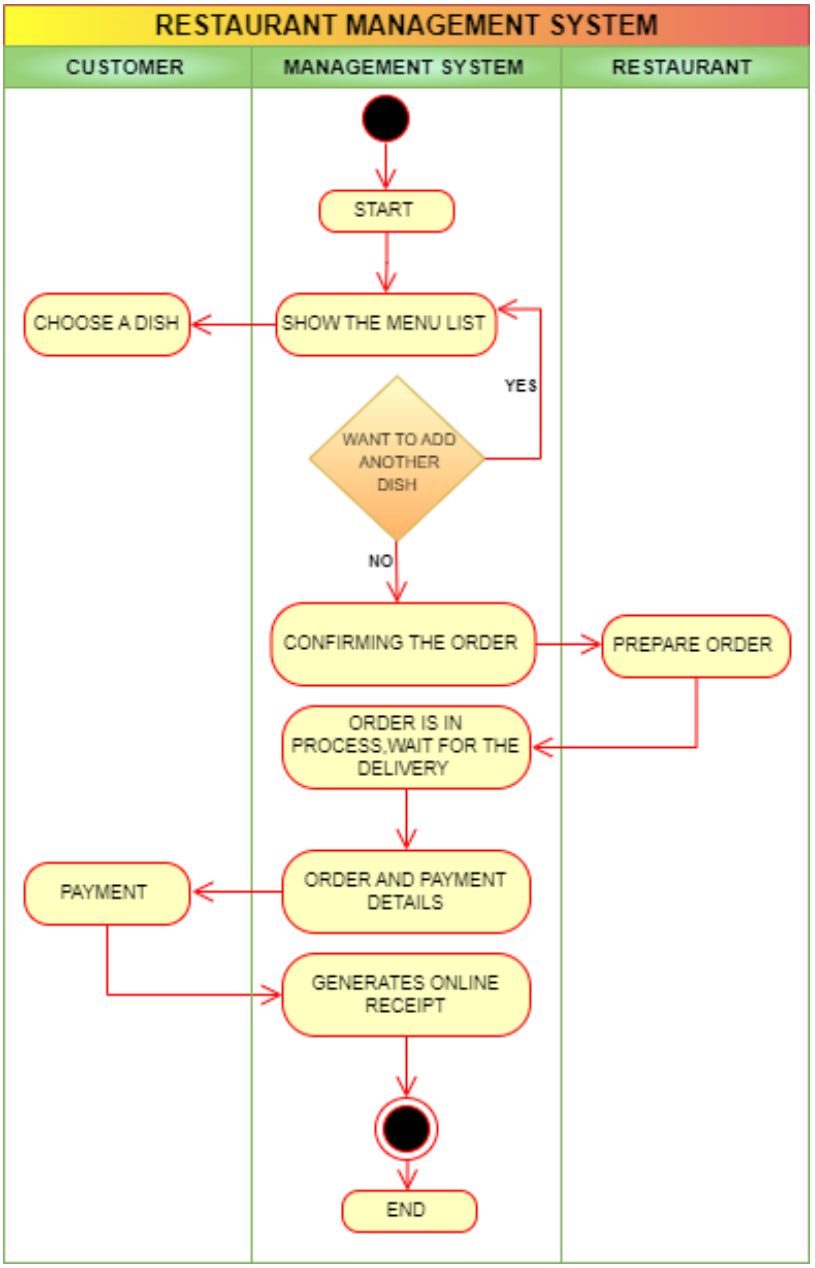
A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. We can say that each and every class has a state but we don't model every class using State diagrams.Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of the Statechart diagram is to model the lifetime of an object from creation to termination.



This statechart diagram of the Restaurant Management System shows the beginning of the process and then shows the various states involved till the completion of the process. We've used various components of statechart diagrams like fork, join, self transition, etc to describe the process to the best of our ability.

ACTIVITY DIAGRAM

An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system. An activity diagram is very similar to a flowchart.

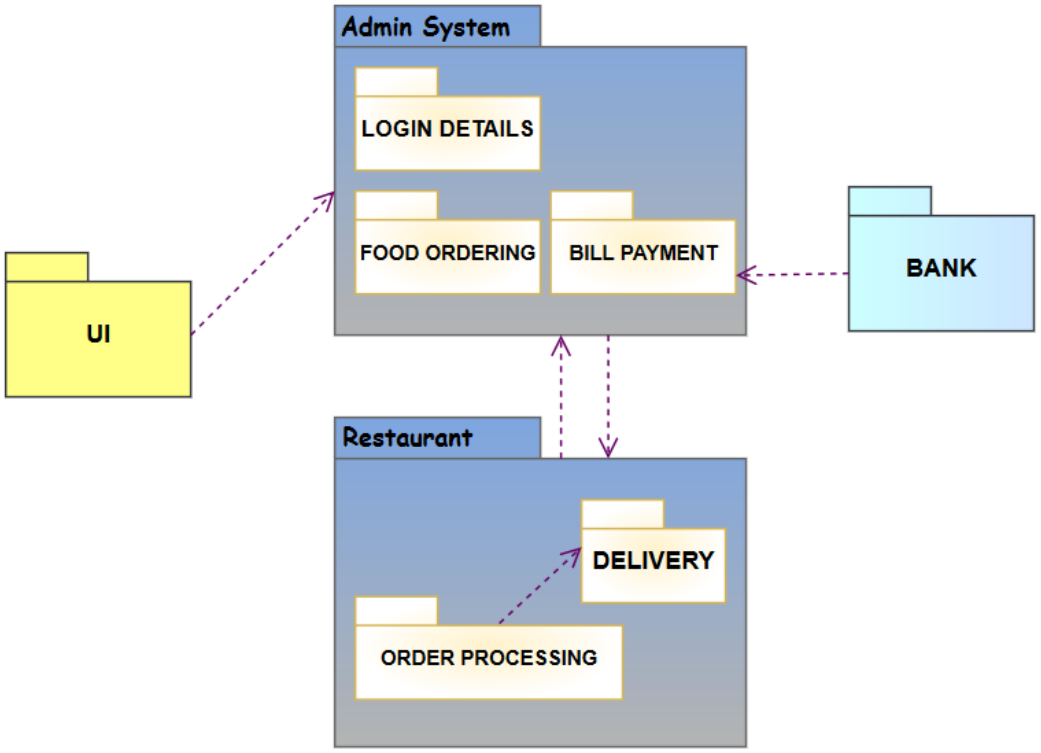


This activity diagram contains 3 swimlanes, namely Customer, Management System, and Restaurant. Activity diagrams notations like control flows, swimlanes, and decision nodes have been used to create the diagram.

CHAPTER 4

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

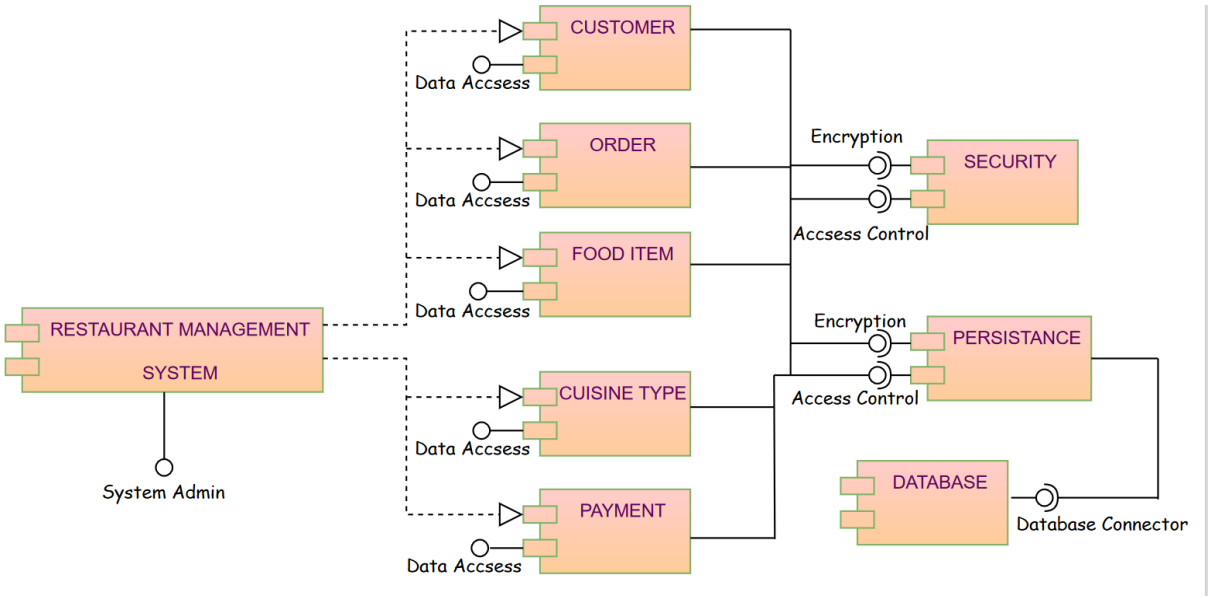


A well-designed package diagram provides numerous benefits to those looking to create a visualization of their UML system or project.

- They provide a clear view of the hierarchical structure of the various UML elements within a given system.
- These diagrams can simplify complex class diagrams into well-ordered visuals.

COMPONENT DIAGRAM

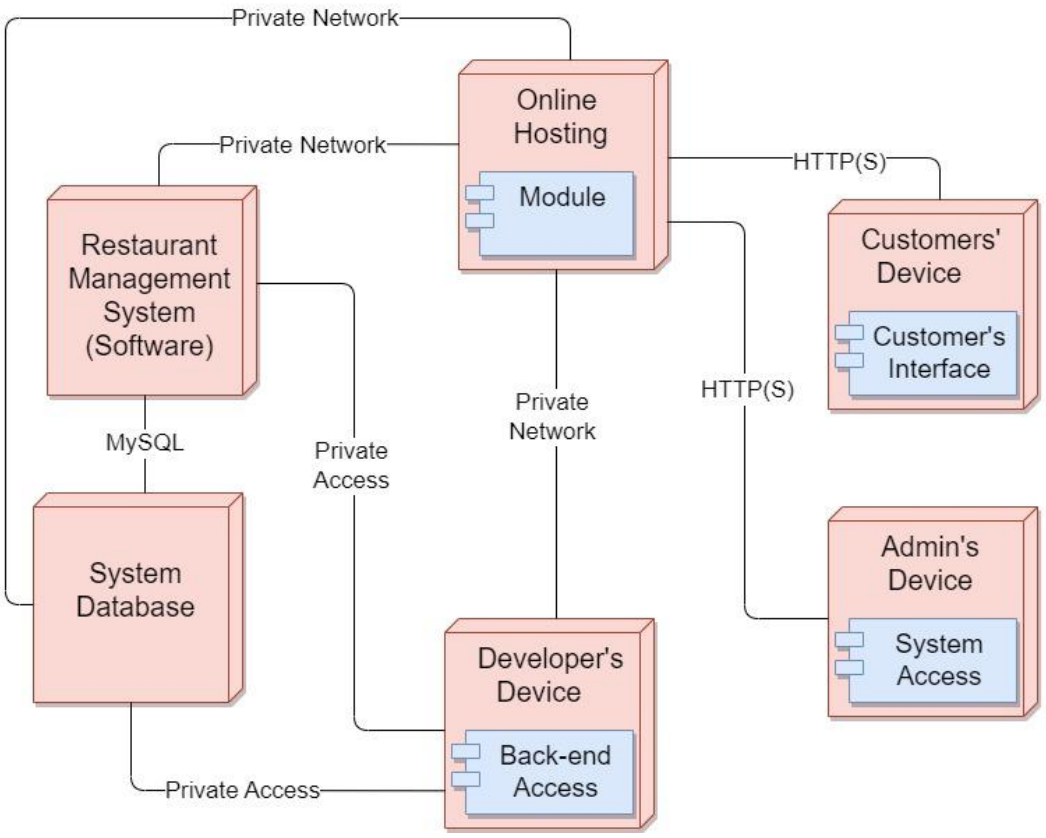
A component diagram is used to break down a large object-oriented system into smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.



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.

DEPLOYMENT DIAGRAM

The main purpose of the deployment diagram is to represent how software is installed on the hardware component. It depicts in what manner software interacts with hardware to perform its execution. The component diagram represents the components of a system, whereas the deployment diagram describes how they are actually deployed on the hardware.



Following are the purposes of the deployment diagram enlisted below:

1. To envision the hardware topology of the system.
2. To represent the hardware components on which the software components are installed.
3. To describe the processing of nodes at the runtime.

CONCLUSION

The UML is non-proprietary and open to all. It addresses the needs of the user and scientific communities, as established by experience with the underlying methods on which it is based. It effectively ends many of the differences, often inconsequential, between the modeling languages of previous methods. It also unifies the perspectives among many different kinds of systems (business versus software), development phases (requirements analysis, design, and implementation), and internal concepts.

UML diagrams work together to guide programmers and beginners on the behaviour of the software. They help programmers in understanding the structure and the working of the system. UML diagrams teach and guide us through our project development journey. All the UML diagrams in this project were created for a Restaurant Management System.

REFERENCES

1. https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/
2. https://www.ibm.com/docs/en
3. https://www.geeksforgeeks.org/unified-modeling-language-uml-state-diagrams/
4. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-practical-guide/#:~:text=At%20the%20conclusion%2C%20UML%20can,to%20identify%20the%20object%20efficiently.
5. https://www.tutorialspoint.com/uml/uml_component_diagram.html