INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

OBJECT ORIENTED SOFTWARE DESIGN LABORATORY								
V Semester: CSE								
Course Code	**Category**	**Hours / Week**			**Credits**	**Maximum Marks**		
ACSC21	**Core**	**L**	**T**	**P**	**C**	**CIA**	**SEE**	**Total**
		0	0	3	1.5	30	70	100
Contact Classes: NIL	**Tutorial Classes: NIL**	**Practical Classes: 45**				**Total Classes: 45**		
Prerequisite: There are no prerequisites to take this course.								

I. COURSE OVERVIEW:

This course introduces analyses, design of a system by applying the object-orientated concepts, and develops a set of graphical system models during the development life cycle of the software. This course includes techniques to produce detailed object models and designs from system requirements, use the modeling concepts provided by UML, identify use cases and expand into full behavioral designs, expand the analysis into a design ready for implementation and construct designs that are reliable, various testing scenarios for the given problem statements.

II. COURSES OBJECTIVES:

The students will try to learn:

I. How to select suitable software development process model for the given scenario.
II. How to classify the requirements and prepare software requirement documents for analyzing the projects.
III. The different design techniques and their implementation to develop the software.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

CO 1 Summarize the features of software in view of Software development process
CO 2 Make Use of UML notations to represent requirement of the system
CO 3 Develop a design model of the software system with the help of UML structural diagrams.
CO 4 Design a behavioral model of the software system with the help of UML structural diagrams
CO 5 Develop a design model for different real time application.

IV. SYLLABUS:

Exercises for Object Oriented Software Design Laboratory

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started Exercises

1.1 *Installation*

1. Install StarUML on your machine.
2. Creating New Diagram using StarUML, such as:
   - StarUML supports 11 types of UML diagrams. The user can freely design and manage different diagrams as needed
   - Select from the model explorer or diagram area an element to contain the new diagram
   - Right-click and select the [Add Diagram] menu. A new diagram will be designed when selection is made for the diagram type
3. Design ALL UML diagrams.

1.2 Software Requirement Specification

The production of the requirements stage of the software development process is Software Requirements Specifications (SRS) also called a requirements document. This report lays a foundation for software engineering activities and is constructing when entire requirements are elicited and analyzed. SRS is a formal report, which acts as a representation of software that enables the customers to review whether it (SRS) is according to their requirements. Also, it comprises user requirements for a system as well as detailed specifications of the system requirements.

Hints

SRS Document Outline.

```
 /* SRS Document Outline */

1. Introduction
Purpose
Document conventions
Intended audience
Additional information
Contact information/SRS team members
References
2. Overall Description
Product perspective
Product functions
User classes and characteristics
Operating environment
User environment
Design/implementation constraints
Assumptions and dependencies
```

```
3.External Interface Requirements
User interfaces
Hardware interfaces
Software interfaces
Communication protocols and interfaces
4. System Features
System feature
Description and priority
Action/result
Functional requirements
System feature B
5. Other Nonfunctional Requirements
Performance requirements
Safety requirements
Security requirements
Software quality attributes
Project documentation
User documentation
6. Other Requirements
Appendix A: Terminology/Glossary/Definitions list
Appendix B: To be determined
```

Conclusion: The SRS document was made successfully by following the steps described above

1.3 Software Requirement Specification Document for e-Health billing software

Design software requirement specification document for e-Health billing software which is to computerize the front office Automation of hospital to develop software which is user friendly, simple, fast and cost effective. This document should contain introduction, general description, external interface requirements, system features, other nonfunctional requirements and other requirements

Hints

Follow SRS Document Outline.

```
 /* SRS for e-Health billing software */

1.Introduction:
Purpose – At first, main aim of why this document is necessary and what's
purpose of document is explained and described.
Scope – In this, overall working and main objective of document and what
value it will   provide to customer is described and explained. It also
includes a description of development cost and time required.
Overview – In this, description of product is explained. It's simply
summary or overall review of product.
2.General description:
In this, general functions of product which includes objective of user, a
user characteristic, features, benefits, about why its importance is
mentioned. It also describes features of user community.
3.Functional Requirements:
In this, possible outcome of software system which includes
effects due to operation of program is fully explained. All functional
```

```
requirements which may include calculations, data processing, etc. are
placed in a ranked order.
4.Interface Requirements:
In this, software interfaces which mean how software program communicates
with each other or users either in form of any language, code, or message
are fully described and explained. Examples can be shared memory, data
streams, etc.
5.Performance Requirements:
In this, how a software system performs desired functions under specific
condition is explained. It also explains required time, required memory,
maximum error rate, etc.
6.Design Constraints:
In this, constraints which simply means limitation or restriction are
specified and explained for design team. Examples may include use of a
particular algorithm, hardware and software limitations, etc.
7.Non-Functional Attributes:
In this, non-functional attributes are explained that are required by
software system for better performance. An example may include Security,
Portability, Reliability, Reusability, Application compatibility, Data
integrity, Scalability capacity, etc.
8.Preliminary Schedule and Budget:
In this, initial version and budget of project plan are explained which
include overall time duration required and overall cost required for
development of project.
9.Appendices:
In this, additional information like references from where information is
gathered, definitions of some specific terms, acronyms, abbreviations,
etc. are given and explained.
```

Conclusion: The SRS was made successfully by following the steps described above

1.4 Software Requirement Specification Document for electronic shopping System

Design Software Requirement Specification Document for electronic shopping System to develop a web-based application to improve the service to the customers and merchant which in turn increases the sales and profit. This document contains introduction, general description, external interface requirements, system features, other nonfunctional requirements and other requirements

Hints

Follow SRS Document Outline.

```
/* SRS for electronic shopping System */

1.Introduction:
Purpose – At first, main aim of why this document is necessary and what's
purpose of document is explained and described.
Scope – In this, overall working and main objective of document and what
value it will provide to customer is described and explained. It also
includes a description of development cost and time required.
```

```
Overview – In this, description of product is explained. It's simply
summary or overall review of product.

2.General description: In this, general functions of product which
includes objective of user, a user characteristic, features, benefits,
about why its importance is mentioned. It also describes features of user
community.

3.Functional Requirements: In this, possible outcome of software system
which includes effects due to operation of program is fully explained. All
functional requirements which may include calculations, data processing,
etc. are placed in a ranked order.

4.Interface Requirements: In this, software interfaces which mean how
software program communicates with each other or users either in form of
any language, code, or message are fully described and explained. Examples
can be shared memory, data streams, etc.

5.Performance Requirements: In this, how a software system performs
desired functions under specific condition is explained. It also explains
required time, required memory, maximum error rate, etc.

6.Design Constraints: In this, constraints which simply means limitation
or restriction are specified and explained for design team. Examples may
include use of a particular algorithm, hardware and software limitations,
etc.

7.Non-Functional Attributes: In this, non-functional attributes are
explained that are required by software system for better performance. An
example may include Security, Portability, Reliability, Reusability,
Application compatibility, Data integrity, Scalability capacity, etc.

8.Preliminary Schedule and Budget: In this, initial version and budget of
project plan are explained which include overall time duration required
and overall cost required for development of project.

9.Appendices: In this, additional information like references from where
information is gathered, definitions of some specific terms, acronyms,
abbreviations, etc. are given and explained.
```

Conclusion: The SRS was made successfully by following the steps described above

1.5 Software Requirement Specification Document for train ticketing Service

Design Software Requirement Specification Document for train ticketing Service. This is very important to design a good-working system software for ticket booking and related transactions. To design it, full-track documentation of models is required as per as software development is concerned.

Hints

Follow SRS Document Outline.

```
/* SRS for Online Railway train ticketing Service */
```

1.Introduction:
Purpose – At first, main aim of why this document is necessary and what's purpose of document is explained and described.
Scope – In this, overall working and main objective of document and what value it will provide to customer is described and explained. It also includes a description of development cost and time required.
Overview – In this, description of product is explained. It's simply summary or overall review of product.

2.General description: In this, general functions of product which includes objective of user, a user characteristic, features, benefits, about why its importance is mentioned. It also describes features of user community.

3.Functional Requirements: In this, possible outcome of software system which includes effects due to operation of program is fully explained. All functional requirements which may include calculations, data processing, etc. are placed in a ranked order.

4.Interface Requirements: In this, software interfaces which mean how software program communicates with each other or users either in form of any language, code, or message are fully described and explained. Examples can be shared memory, data streams, etc.

5.Performance Requirements: In this, how a software system performs desired functions under specific condition is explained. It also explains required time, required memory, maximum error rate, etc.

6.Design Constraints: In this, constraints which simply means limitation or restriction are specified and explained for design team. Examples may include use of a particular algorithm, hardware and software limitations, etc.

7.Non-Functional Attributes: In this, non-functional attributes are explained that are required by software system for better performance. An example may include Security, Portability, Reliability, Reusability, Application compatibility, Data integrity, Scalability capacity, etc.

8.Preliminary Schedule and Budget: In this, initial version and budget of project plan are explained which include overall time duration required and overall cost required for development of project.

9.Appendices: In this, additional information like references from where information is gathered, definitions of some specific terms, acronyms, abbreviations, etc. are given and explained

Conclusion: The SRS was made successfully by following the steps described above

2. Structural Modeling-Object Diagram

Object diagrams represent the static view of a system but this static view is a snapshot of the system at a particular moment. Object diagrams are used to render a set of objects and their relationships as an instance. Object diagrams represent an instance of a class diagram

Common Modeling Techniques for Object Diagram:

1. First, analyze the system and decide which instances have important data and association.
2. Second, consider only those instances, which will cover the functionality.
3. Third, make some optimization as the number of instances are unlimited.

2.1 Object diagram for a Company's structure

Design and develop an object diagram for a company's structure. a company's structure, from which there are mainly two departments - The sales department and the R&D department. The department contains persons.

Hints:

```
/* Object diagram for Company’s structure */

To design an Object Diagram:
Select first an element where a new Object Diagram to be contained as a child.
Select Model | Add Diagram | Object Diagram in Menu Bar or select Add Diagram | Object Diagram in Context Menu.

To design an Object:
Select Object in Toolbox.
Drag on the diagram as the size of Object.
Name Expression: Edit name expression.
Syntax of Name Expression
expression:: = [ '<<' stereotype `>>` ] [ visibility] name
stereotype: = (identifier)
visibility: = '+' | '#' | '-' | '~'
name: = (identifier)

To design a Link (or Directed Link):
Select Link (or Directed Link) in Toolbox.
Drag from an instance and drop on another instance.

Objects for company structure : Company, department1, department 2,person
```

Conclusion: The Object diagram was designed successfully by following the steps described above

2.2 Object diagram for a course department

Design and develop an object diagram for a university course department. the courses are math, statics, it divided by graduate and undergraduate.

Hints:

```
/* Object diagram for Company's structure */

To design an Object Diagram:

Select first an element where a new Object Diagram to be contained as a
child.
Select Model | Add Diagram | Object Diagram in Menu Bar or select Add
Diagram | Object Diagram in Context Menu.

To design a Object:
Select Object in Toolbox.
Drag on the diagram as the size of Object.

Name Expression: Edit name expression.
Syntax of Name Expression
expression:: = [ '<<' stereotype `>>` ] [ visibility] name
stereotype: = (identifier)
visibility:: = '+' | '#' | '-' | '~'
name:: = (identifier)
Link

To design a Link (or Directed Link):
Select Link (or Directed Link) in Toolbox.
Drag from an instance and drop on another instance.
Objects : Deparment, Course, Math, stats, as attributes are graduates,
undergraduates.
```

Conclusion: The Object diagram was designed successfully by following the steps described above

2.3 Object diagram for just-in-time (JIT) inventory system

Design and develop an object diagram for just-in-time (JIT) inventory system. It specifically shows the instance process of a purchase at a particular time in a system. There are objects such as customer, order, normal order, and special order. The customer is related to the order objects which are associated with normal and special-order objects.

Hints:

```
/* Object diagram for just-in-time (JIT) inventory system */
To design an Object Diagram:
Select first an element where a new Object Diagram to be contained as a
child.
Select Model | Add Diagram | Object Diagram in Menu Bar or select Add
Diagram | Object Diagram in Context Menu.

To design an Object:
Select Object in Toolbox.
Drag on the diagram as the size of Object.
```

Conclusion: The Object diagram was designed successfully by following the steps described above

2.4 Object diagram for e-Learning on Smart Library

Design and develop an object diagram for e-Learning on Smart Library. This should illustrate an instance in a library system. There are five objects in the diagram such as administrator, magazine, article, comment, and person. What's more, it contains different relationships where one object would comprise another object

Hints:

```
/* Object diagram for e-Learning on Smart Library */

To design an Object Diagram:
Select first an element where a new Object Diagram to be contained as a child.
Select Model | Add Diagram | Object Diagram in Menu Bar or select Add Diagram | Object Diagram in Context Menu.

To design an Object:
Select Object in Toolbox.
Drag on the diagram as the size of Object.
Objects: administrator, magazine, article, comment, and person

/* Object diagram for Order Automation Systems */
Objects: administrator, magazine, article, comment, and person
```

Conclusion: The Object diagram was designed successfully by following the steps described above

2.5 Object diagram for e-tailing system

Design and develop an object diagram for e-tailing system. The object diagram of an e-tailing system is used to show how the parts of a system work together to make the online shopping operate. The object Diagram for e-tailing system represents the objects and the links between objects. It's an instance of class diagram that shows how objects are linked one to other.

Hints:

```
/* Object diagram for e-tailing system */

To design an Object Diagram:
Select first an element where a new Object Diagram to be contained as a child.
Select Model | Add Diagram | Object Diagram in Menu Bar or select Add Diagram | Object Diagram in Context Menu.


To design an Object:
Select Object in Toolbox.
Drag on the diagram as the size of Object.
```

```
Objects: order, product, customer, account.
```

Conclusion: The Object diagram was designed successfully by following the steps described above

2.6 Object diagram for Elevator Control System

Design and develop an object diagram for Elevator Control System. The elevator system designed an "ideal" elevator in which some of the technical corners are cut. Our elevator has the basic function that all elevator systems have, such as moving up and down, open and close doors, and of course, pick up passengers. The elevator is supposed to be used in a building having floors numbered from 1 to MaxFloor, where the first floor is the lobby. There are car call buttons in the car corresponding to each floor. For every floor except for the top floor and the lobby, there are two hall call buttons for the passengers to call for going up and down. There is only one down hall call button at the top floor and one up hall call button in the lobby. When the car stops at a floor, the doors are opened and the car lantern indicating the current direction the car is going is illuminated so that the passengers can get to know the current moving direction of the car. The car moves fast between floors, but it should be able to slow down early enough to stop at a desired floor. In order to certificate system safety, emergency brake will be triggered and the car will be forced to stop under any unsafe conditions.

Hints:

```
/* Object diagram for Elevator Control Systems */

To design an Object Diagram:
Select first an element where a new Object Diagram to be contained as a child.
Select Model | Add Diagram | Object Diagram in Menu Bar or select Add Diagram | Object Diagram in Context Menu.

To design an Object:
Select Object in Toolbox.
Drag on the diagram as the size of Object.

Objects: Door, elevator control, car, button, indicator, safety
```

Conclusion: The Object diagram was designed successfully by following the steps described above

3. Structural Modeling-Class Diagram

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

Common Modeling Techniques for Class Diagram:

1. Modeling Simple Collaborations
2. Modeling a logical database Schema
3. Forward and Reverse Engineering

3.1 Class Diagram for Intelligent Information Service System of Smart Library

Design and develop a class diagram for a Intelligent Information Service System of Smart Library, which is a software built to handle the primary functions of a library. An innovative system provides intelligent services in the library to both users and the terminal. Compared to the core digital reading room, they can make wise judgments on the retrieval and use of information assets. The implementation of succeeding value management based on the latest technological tools is required for learning to provide knowledge services and fulfil its role as convergence is capable of reacting to varied data needs. The major obstacles to digital libraries are lack of planning and software, import restrictions on equipment, inadequately skilled staff, lack of standards, and a refusal to cooperate. Libraries rely on library Automation systems to manage asset collections as well as relationships with their members. Library Automation systems help libraries keep track of the books and their checkouts, as well as members' subscriptions and profiles.

Library Automation systems also involve maintaining the database for entering new books and recording books that have been borrowed with their respective due dates.

1. Identify the objects and classes
2. Clearly identify what each class is responsible for
3. Identify attributes and methods of each class
4. Identify the suitable relationships among the classes

Hints:

```
/* Class diagram for Library Automation System */

 1. Classes of Library Automation System:
•Library Automation System class
It manages all operations of Library Automation System. It is central part
of organization for which software is being designed.
•User Class
It manages all operations of user.
•Librarian Class
It manages all operations of Librarian.
•Book Class
It manages all operations of books. It is basic building block of system.
•Account Class
It manages all operations of account.
•Library database Class
It manages all operations of library database.
•Staff Class
It manages all operations of staff.
•Student Class
It manages all operations of student.
2.  Attributes of Library Automation System:
•Library Automation System Attributes
```

User Type, Username, Password
•**User Attributes**
Name, Id
•**Librarian Attributes**
Name, Id, Password, Search_String
•**Book Attributes**
Title, Author, ISBN, Publication
•**Account Attributes**
no_borrowed_books, no_reserved_books, no_returned_books, no_lost_books
fine amount
•**Library database Attributes**
List_of_books
•**Staff Class Attributes**
Dept
•**Student Class Attributes**
Class
Methods of Library Automation System:
•**Library Automation System Methods**
Login (), Register (), Logout ()
•**User Methods –**
Verify (), Check Account (), get_book_info ()
•**Librarian Methods**
Verify librarian (), Search ()
•**Book Methods**
Show_duedt (), Reservation_status (), Feedback (), Book_request (), Renew_info ()
•**Account Methods**
Calculate fine ()
•**Library database Methods**
Add (), Delete (), Update (), Display (), Search ()

Conclusion: The class diagram was designed successfully by following the steps described above

3.2 Class Diagram for Smart Automatic Teller Machine

Design and develop a class diagram for Smart ATM system, this diagram should provide an effective visual representation of how an automated teller machine operates. An ATM class diagram contains common classes, relationships, associations and components of an ATM, including the customer, card, terminal and bank. This diagram should show how the different components interact within the system to enable customers to withdraw and deposit money. The ATM class diagram should also highlight the role of ATMs in financial transactions, such as transferring funds from one account to another, bill payments and banking operations. This diagram is essential for understanding the functionality of ATMs and how they are used in financial services.

1. Identify the objects and classes
2. Clearly identify what each class is responsible for
3. Identify attributes and methods of each class
4. Identify the suitable relationships among the classes

Hints:

```
/* Class diagram for Smart Automatic Teller Machine */

To design a Class Diagram:
First select an element where a new Class Diagram to be contained as a
child.
Select Model | Add Diagram | Class Diagram in the Menu Bar or select Add
Diagram | Class Diagram in Context Menu.
Name Expression: Edit name expression.
Syntax of Name Expression
expression: = [ '<<' stereotype `>>`] [ visibility] name
stereotype :: = (identifier)
visibility:: = '+' | '#' | '-' | '~'
name:: = (identifier)
Visibility: Change visibility property.
Add Note: Add a linked note.
Add Constraint: Add a constraint.
Add Attribute (Ctrl+Enter): Add an attribute.
Add Operation (Ctrl+Shift+Enter): Add an operation.
Add Reception: Add a reception.

To design a Class:
Select Class in Toolbox.
Drag on the diagram as the size of Class.

To design a Class (model element only) by Menu:
Select an Element where a new Class to be contained.
Select Model | Add | Class in Menu Bar or Add | Class in Context Menu.
Name Expression: Edit name expression.
Syntax of Name Expression
expression:: = [ '<<' stereotype `>>` ] [ visibility] name
stereotype:: = (identifier)
visibility:: = '+' | '#' | '-' | '~'
name:: = (identifier)
Visibility: Change visibility property.
Add Note: Add a linked note.
Add Constraint: Add a constraint.
Add Attribute (Ctrl+Enter): Add an attribute.
Add Operation (Ctrl+Shift+Enter): Add an operation.
Add Template Parameter: Add a template parameter.
Add Reception: Add a reception.
Add Sub-Class: Add a sub-class.
Add Super-Class: Add a super class.
Add Provided Interface: Add a provided interface.
Add Required Interface: Add a required interface.
Add Associated Class: Add an associated class.
Add Aggregated Class: Add an aggregated class.
Add Composited Class: Add a composited class.
Add Port: Add a port.

Attribute
To add an Attribute:
Select a Classifier.
```

```
Select Model | Add | Attribute in Menu Bar or Add | Attribute in Context
Menu
Attribute Expression: Edit Attribute expression.
Syntax of Attribute Expression
attribute: = [ '<<' stereotype `>>`] [ visibility] name [':' type] [ '['
multiplicity ']' ] [ '=' default-value ]
stereotype ::= (identifier)
visibility ::= '+' | '#' | '-' | '~'
name ::= (identifier)
type ::= (identifier)
multiplicity: = multiplicity-bound [ '..' multiplicity-bound]
multiplicity-bound :: = (number) | '*'
default-value: = (string)

To add an operation:
Select a Classifier.
Select Model | Add | Operation in Menu Bar or Add | Operation in Context
Menu.
Operation Expression: Edit Operation expression.
Syntax of Operation Expression
Operation:: = [ '<<' stereotype `>>` ] [ visibility ] name [ '('
parameter-list ')' ] [ ':' return-type ]
stereotype:: = (identifier)
visibility ::= '+' | '#' | '-' | '~'
name ::= (identifier)
parameter-list:: = parameter [ ',' parameter] *
parameter ::= (identifier)
return-type:: = (identifier)
```

Conclusion: The class diagram was designed successfully by following the steps described above

3.3 Class Diagram for e-caravansary System

Design and develop a class diagram for an e- caravansary System. An e- caravansary System is a software built to handle all online caravansary activities easily and safely. This System will give the hotel caravansary automation power and flexibility to manage the entire system from a single online portal. The system allows the manager to keep track of all the available rooms in the system as well as to book rooms and generate bills.

1. Identify the objects and classes
2. Clearly identify what each class is responsible for
3. Identify attributes and methods of each class
4. Identify the suitable relationships among the classes

Hints:

```
/* Class diagram for e-caravansary System */

The classes used in this system are,
•Hotel Automation: This class depicts the entire hotel and says whether
the hotel is opened or closed.
```

•Employees: It contains the details of the Employee. There are two kinds of employees, Server and the chef. This employee class is the parent class of two subclass – Server and Chef
•Server: It contains the details of the server, the table to which they are assigned, the order which is currently serving, etc.
•Chef: It contains the details of the chef working on a particular order.
•Customer: It contains the details of the customer.
•Table: It contains the table details like table number and the server who are assigned to that table.
•Menu: Menu contains all the food items available in the restaurant, their availability, prize, etc.
•Order: Order depicts the order associated with a particular table and the customer.
•Bill: Bill is calculated using the order and menu.
•Payment: This class is for doing payment. The payment can be done in two ways either cash or card. So, payment is the parent class and cash and card are subclasses.
•Cash: Payment can be done by cash
•Card: Payment can be done by card or online
Attributes:
Hotel Automation – HotelName, NumberOfEmployees
•Employees – EmployeeId, EmployeeName, EmployeeSalary
•Server – ServerId, OrderId
•Chef – Chef_Id, OrderId
•Customer – CustomerId, CustomerName, Bill_Id, OrderId, PaymentId
•Table – TableNumber, OccupiedStatus, ServerId, CustomerId
•Menu – ItemId, ItemName, Amount
•Order – OrderId, ItemId, ItemName, Quantity, CustomerId, ServerId
•Bill – Bill_Id, OrderId, TotalBill
•Payment – PaymentId, Bill_Id
Methods:
1. Hotel Automation:
•open () -This is used to indicate if the hotel is functioning or not.
2. Employees:
•employee details () – This method contains the details of the employee.
3. Customer:
•customer_details () – This depicts the details of the customer.
•ordered_items () – This method contains the items which are ordered by the customer.
•payment_status () -This says whether the customer paid or not.
4. Table:
•table_details () – This method contains the details of the table along with the customer and no of seats.
•availability status () – This method says whether the table is occupied or not.
5. Menu:
•items () – This method displays the menu items, their availability and their price.
6. Order:
•order_items () – This method orders the items selected by the user from the menu.
7. Bill:
•calculate bill () – This method calculates the bill for a particular table.

8. Payment:
•ispaid () – It shows whether payment is successful or not.
Relationships:
Inheritance:
Here, Employee is parent class Server and Chef are child classes because server is a employee and chef is a employee.

Association:
Here,
•Employee and customer
•Server and table
•Customer and payment
•Chef and order
follows association relationships.

Composition:
Here,
•Menu and Order
•Order and Bill
•Bill and Payment
follows composition relation
Order cannot exist without Menu; Bill cannot exist without order and payment cannot exist without bill. So here order is contained inside the menu, bill is contained inside the order and payment is contained inside the bill.

Aggregation:
Here,
•Customer and Server
•Chef and Server
follows Aggregation relation
Server is associated with the customer but can exist without the customer as well, Likewise Chef is associated with the server but can exist without the server as well.

Conclusion: The class diagram was designed successfully by following the steps described above

3.4 Class Diagram for Automated Financial Services

Design and develop a class diagram for an Automated Financial Services. Automated Financial Services is a web-based tool that allows the Bureau of the Fiscal Service to pay financial institutions for services rendered. BMS also has analytical tools that may be used to examine and approve pay, budgets, and outflows. Along with this, the main goal of bank Automation is to make sure that all of the different parts of a bank work together in a way that makes the most money possible. Besides, the system allows customers to design accounts, deposit and withdraw money from their accounts, and examine reports for all of their accounts.

1. Identify the objects and classes
2. Clearly identify what each class is responsible for
3. Identify attributes and methods of each class
4. Identify the suitable relationships among the classes

Hints:

```
/* Class diagram for Automated Financial Services */
List of Classes
1.Customer
2.Bank
3.ATM
4.Account
5.ATM transaction
class (customer)

Attributes/Variables of the class (customer)
1.customer_ password: varchar
Public Attributes/Variables
There are following public attributes in the mentioned class diagram.
1.+customer_ id: int
2.+customer_ name: string
3.+customer_ email: string
4.+customer_ phone No: string
1.+customer_ username: string
2.+customer_ address: string
3.+customer_ card no: int
Functions of the class (customer)

functions
1.+add_ customer ()
2.+delete_ customer ()
3.+edit_ customer ()
4.+search_ customer ()
5.+verify_ password ()
 class (Bank)
Attributes/Variables of the class (Bank)
Private Attributes/Variables:
We can assign private attributes/ values to this class, but suppose that
currently we are not willing to make the attributes private.

Public Attributes/Variables:
public attributes.
1. +code: int
2.+address: int
3.+name: string
Functions of the class (bank)
There are following functions in the mentioned class diagram.
+get_ Account ()
 class (ATM)
Attributes/Variables of the class (ATM)
Private Attributes/Variables:
•+ATM_ location: int
•+ATM_ managed by int
Functions of the class (ATM)
There are the following functions in the mentioned class diagram.
•+deposit ()
•+withdraw ()
•+check_ balance ()
 class (Account)
```

```
Attributes/Variables of the class (Account)
Private Attributes/Variables:
assign private attributes/ values to this class, but suppose that currently, we are not willing to make the attributes private.
Public Attributes/Variables:
```

Conclusion: The class diagram was designed successfully by following the steps described above

4. Structural Modeling-Component Diagram

Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

4.1 Component diagram for Passport Issuance Automation

Design the component diagram for passport issuance automation system. It is used in the effective dispatch of passport to all of the applicants. This system adopts a comprehensive approach to minimize the manual work and schedule resources, time in a cogent manner. The core of the system is to get the online registration form (with details such as name, address etc.,) filled by the applicant whose testament is verified for its genuineness by the Passport Automation System with respect to the already existing information in the database.

Hints

```
/**
 *Design the Static implementation view with Component diagram.
 */
To design a Component Diagram:
Select first an element where a new Component Diagram to be contained as a child.
Select Model | Add Diagram | Component Diagram in Menu Bar or select Add Diagram | Component Diagram in Context Menu.

To design a Component:
Select Component in Toolbox.
Drag on the diagram as the size of Component.

To design a Component (model element only) by Menu:
Select an Element where a new Component to be contained.
Select Model | Add | Component in Menu Bar or Add | Component in Context Menu.
A component represents a modular part of a system, that encapsulates it contents and whose manifestation is replaced with in its environment. A component defines its behaviors in terms of provide and required interfaces.

• Here the three components are applicant, system admin and authority.
```

```
• The interface between people and system admin, from people to authority.

The applicant, System admin, enquiry verification are components being
interacted. Here the three components are applicant, system admin and
authority.
The interface between people and system admin, from people to authority.
```

Conclusion: The component diagram was designed successfully by following the steps described above

4.2 Component diagram for Digital commerce platform

The component diagram of an Digital commerce platform is used to show how the parts of a system work together to make the online shopping operate. A component diagram shows how the software's parts are organized and how they depend on each other. This diagram gives a high-level look at the parts of a system.

Hints

```
      //* Design the Static implementation view with Component diagram *//

To design a Component Diagram:
Select first an element where a new Component Diagram to be contained as a
child.
Select Model | Add Diagram | Component Diagram in Menu Bar or select Add
Diagram | Component Diagram in Context Menu.

To design a Component:
Select Component in Toolbox.
Drag on the diagram as the size of Component.

To design a Component (model element only) by Menu:
Select an Element where a new Component to be contained.
Select Model | Add | Component in Menu Bar or Add | Component in Context
Menu

component diagram shows how an online shopping system will be made up of a
set of deployable components, such as dynamic-link library (DLL) files,
executable files, or web services. Using well-defined interfaces, these
parts communicate with each other and keep their internal details hidden
from each other and the outside world.

components which are the product database, transaction database, product
list, shopping cart, order details, and the user. The component product
list is the required interface which is dependent on the provided
component Product database. The component transaction database is also
dependent to order details.
```

Conclusion: The component diagram was designed successfully by following the steps described above

4.3 Component diagram for Smart bank services

Design and develop a component diagram for an Smart bank services which will be made up of a set of deployable components, such as dynamic-link library (DLL) files, executable files, or

web services. Using well-defined interfaces, these parts communicate with each other and keep their internal details hidden from each other and the outside world.

Hints

```
/**
 * Design the Static implementation view with Component diagram for Smart bank services.
 */
To design a Component Diagram:
Select first an element where a new Component Diagram to be contained as a child.
Select Model | Add Diagram | Component Diagram in Menu Bar or select Add Diagram | Component Diagram in Context Menu.

To design a Component:
Select Component in Toolbox.
Drag on the diagram as the size of Component.
To design a Component (model element only) by Menu:
Select an Element where a new Component to be contained.
Select Model | Add | Component in Menu Bar or Add | Component in Context Menu
The components are labeled to clarify their part in the system's operation. They were represented by symbols that explain their function and role in the overall ATM system operation.

The component diagram of ATM system has 8 components which are the account database, transaction database, balance inquiry, withdraw, deposit, loan, card, and the user. The components under the ATM system are the required interface at the same time are provided interface which serves as the provider for the transaction database and required for the accounts database

The dependencies on each component are explained through the lines and arrows drawn in the diagram
```

Conclusion: The component diagram was designed successfully by following the steps described above

4.4 Component diagram for E-Health Record software (EHR)System

Design and develop the Component Diagram for E-Health Record software (EHR)System which is used to show the overall flow of the system parts work together to make the hospital system perform efficiently.

Hints

```
 /**
 * Design the Static implementation view with Component diagram for E-Health Record software(EHR)System.
 */
To design a Component Diagram:
Select first an element where a new Component Diagram to be contained as a child.
```

```
Select Model | Add Diagram | Component Diagram in Menu Bar or select Add Diagram | Component Diagram in Context Menu.

To design a Component:
Select Component in Toolbox.
Drag on the diagram as the size of Component.

To design a Component (model element only) by Menu:
Select an Element where a new Component to be contained.
Select Model | Add | Component in Menu Bar or Add | Component in Context Menu
```

Conclusion: The component diagram was designed successfully by following the steps described above

4.5 Component diagram for Digital Food Delivery Platform

Design and develop a component diagram of a Digital Food Delivery Platform. An Online Food Ordering System is a piece of software that allows restaurants, coffee shops, and bars to take orders over the internet. In most cases, customers can choose and pay for food before the kitchen is told that an order has been made.

Hints

```
/**
 * Design the Static implementation view with Component diagram for Digital Food Delivery Platform.
 */
Here are the steps in developing the food ordering system component diagram

1. Finalize the Function and Processes of the Software
2. Put the Components included
3. Add the Dependencies (Ports and interfaces)

The component's port is a feature that indicates where the component and its environment meet.

Interfaces show how components are wired together and how they work together. When a component needs a certain interface, the assembly connector lets you connect it to another component that already has that interface. It looks like a semi-circle and a line.
```

Conclusion: The component diagram was designed successfully by following the steps described above

4.6 Component diagram for Smart library information services

Design and develop a component diagram of a Smart library information services which is software that can be used to manage all the tasks of a library, like how many members can come in and out. This tool keeps track of new books and the books that members have checked out. They can also be monitored when the borrowing session is due.

Hints

```
/**
 * Design the Static implementation view with Component diagram for Smart library information services.
 */
This component diagram shows the structure of the library system, which consists of the software components and their interfaces, and how they work together.

The component diagram of library Automation system has 5 components which are book database, transaction database, output, online search, and the user. The component “output” is the required interface which is dependent on the provided book database component. The included components were just based on the main function of the system.

The dependencies on each component are explained through the lines and arrows drawn in the diagram.
```

Conclusion: The component diagram was designed successfully by following the steps described above

5. Structural Modeling-Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships. Deployment diagrams are used for describing the hardware components, where software components are deployed. Component diagrams and deployment diagrams are closely related.

5.1 Deployment diagram for Mobile Banking Android Services

Design and develop a deployment diagram of a Mobile banking android services which showcase the execution architecture of a system. This includes both the hardware and the software execution environments and their connecting factors.

Hints

```
/**
 * Design the Static Deployment view with Deployment diagram for Mobile banking android services */

To design a Deployment Diagram:
Select first an element where a new Deployment Diagram to be contained as a child.
Select Model | Add Diagram | Deployment Diagram in Menu Bar or select Add Diagram | Deployment Diagram in Context Menu.

To design a Node:
Select Node in Toolbox.
Drag on the diagram as the size of Node.

To design a Node (model element only) by Menu:
Select an Element where a new Node to be contained.
```

```
Select Model | Add | Node in Menu Bar or Add | Node in Context Menu.
Name Expression: Edit name expression.
Syntax of Name Expression
expression:: = [ '<<' stereotype `>>` ] [ visibility] name
stereotype:: = (identifier)
visibility ::= '+' | '#' | '-' | '~'
name:: = (identifier)
To design a Deployment:
Select Deployment in Toolbox.
Drag from an element (to be deployed) and drop on a Node.
To design a Communication Path:
Select Communication Path in Toolbox.
Drag from a Node and drop on another Node.
a node represents the client's device, which is an Android system. A
component represents the banking application in the device. The user goes
through the web to interact with the banking server and perform the
required task.
Nodes are Client device android, web services, Application Server, Data
Storage
Components in the application server are Account statement preparation,
fund transfer, cheque transfer, other services
```

Conclusion: The deployment diagram was designed successfully by following the steps described above

5.2 Deployment diagram for Digital payment solutions

Design and develop a deployment diagram of an Digital payment solutions which showcase the execution architecture of a system. This includes both the hardware and the software execution environments and their connecting factors

Hints

```
 /**
 * Design the Static Deployment view with Deployment diagram for Digital
payment solutions */

This diagram will show how an e-commerce platform receives payment from a
customer for a product. There's an interaction between the user's PC, the
e-commerce server, and the bank server to get the payment done. You can
see three nodes: the user's PC, the bank server, and the e-commerce
server.
```

Conclusion: The deployment diagram was designed successfully by following the steps described above

5.3 Deployment diagram for an intelligent banking system service

Design and develop a deployment diagram of an intelligent banking system services. The deployment diagram clarifies the communications between links(nodes) present in the banking system. This concept enables the banking system to work according to the design given to it. Deployment diagrams depict the setup of run-time processing nodes and the components that reside on them.

Hints

```
/**
 * Design the Static Deployment view with Deployment diagram for an intelligent banking system service */

The banking system uses a UML deployment diagram to show how should the developed software be deployed.

The deployment diagram shows the scenario when the system is deployed. It has 4 nodes represented with boxes and relationship connections.

The nodes are the banking system, the client’s PC, the bank application, and the bank database. The system node contains developed software that will hold the banking materials needed online.

For the connection, the system is connected to the application and database using a private network which enables it to pass a connection to the devices and enable clients to access the system. The database and the application then can communicate using a TCP/IP connection.
```

Conclusion: The deployment diagram was designed successfully by following the steps described above

5.4 Deployment diagram for Gaming and Entertainment System

Design and develop a deployment diagram of UML is also used in the gaming industry for modeling the game's architecture, character interactions, and story.

A deployment diagram in the context of a Gaming and Entertainment System illustrates the physical arrangement of hardware and software components in a distributed system. This type of diagram shows how software components are deployed and interact with the hardware components in different nodes. Here's a simplified example of a deployment diagram for a Gaming and Entertainment System.

Hints

```
  /**
 * Design the Static Deployment view with Deployment diagram for Gaming and Entertainment System */

The nodes include
User's Device: Represents the device used by the user to access and interact with the gaming and entertainment system. This could be a PC, console, mobile device, or any other platform.

Game Client: The software component running on the user's device responsible for rendering the game graphics, handling user input, and communicating with the game server.

Game Server: This component manages the game's logic and coordinates communication between players. It may also handle tasks like matchmaking,
```

```
scoring, and other multiplayer aspects. The game server interacts with
both the user's device and the database server.

Game Logic: This component represents the core logic of the game and is
executed on the game server. It includes game rules, scoring mechanisms,
and other gameplay-related functionality.

Database Server: Stores and manages game-related data such as player
profiles, game state, and other relevant information. The game server
communicates with the database server to retrieve and store data.
```

Conclusion: The deployment diagram was designed successfully by following the steps described above

5.5 Deployment diagram for Cyber Car Rental Network

Design and develop a deployment diagram of Cyber Car Rental Network. Online vehicle rental software allows to keep precise records of your whole fleet in one location, making day-to-day operations straightforward.

Customers can hire a vehicle through a car rental web-based system. This technology allows the company to make its services available to the general public via the internet while also keeping track of its performance.

Hints

```
  /**
 * Design the Static Deployment view with Deployment diagram for Cyber Car
Rental Network */
The deployment diagram shows the scenario when the system is deployed. It
has 5 nodes represented with boxes and relationship connections.

The nodes are the car rental system, the webserver (system server), the
admin’s device, the client’s device, and the car owner’s device. The
system car rental system node contains a developed database and other
component that will hold the details of the system online.

For the connection, the system is placed within the server, whilst the
client and car owner’s devices and the server were connected using HTTPS.
The admin’s device uses a private network which enables it to pass a
connection to the devices and enable the admin to access the system and
database. The admin and the other users can communicate through the
system.
```

Conclusion: The deployment diagram was designed successfully by following the steps described above

5.6 Deployment diagram for HR Operations Automation System

Design and develop a deployment diagram of an HR Operations Automation System which is a designed software to keep up with employees’ information in an establishment. This software keeps track of their employees’ information and the specifics of their payroll system allowing them to issue payroll information.

The software has the complete set of employee Automation tools that a company needs to keep track of employee information, engagement, and performance, as well as make more money for the whole company.

The employee system also directs and supervises the activities of employees in the appropriate direction. Keeps and manages information that is important to your employees in a safe way, like personal and work-related information

Hints

```
 /**
 * Design the Static Deployment view with Deployment diagram for HR
Operations Automation System */

The designed deployment diagram for employee system shows the components
(nodes) included to carry out the process. Nodes are represented by boxes
that are labeled as software or hardware that specify the included
components to carry out the employee Automation process. The boxes will
then be connected and labeled to declare the type of connection they have
with the other components.
```

Conclusion: The deployment diagram was designed successfully by following the steps described above

6. Behavioral Modeling-Use case Diagram

Use case diagrams model the behavior of a system and help to capture the requirements of the system. Use case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

6.1 Use case diagram for Airport check-in and security screening business model

Design and development a use case diagram for Airport check-in and security screening business model.

Hints

```
  /**
 * Design the Static Deployment view with Use case diagram for Airport
check-in and security screening.

 */To design a Use Case Diagram:
Select first an element where a new Use Case Diagram to be contained as a
child.
Select Model | Add Diagram | Use Case Diagram in Menu Bar or select Add
Diagram | Use Case Diagram in Context Menu.

To design a Use Case Subject:
Select Use Case Subject in Toolbox.
```

```
Drag on the diagram as the size of Use Case Subject.

To design an Actor:
Select Actor in Toolbox.
Drag on the diagram as the size of Actor.

To design an Actor (model element only) by Menu:
Select an Element where a new Actor to be contained.
Select Model | Add | Actor in Menu Bar or Add | Actor in Context Menu.
Name Expression: Edit name expression.
Syntax of Name Expression
expression:: = [ '<<' stereotype `>>` ] [ visibility ] name
stereotype:: = (identifier)
visibility ::= '+' | '#' | '-' | '~'
name ::= (identifier)

To design an Include:
Select Include in Toolbox.
Drag from a Use Case and drop on another Use Case (to be included).

To design an Extend:
Select Extend in Toolbox.
Drag from a Use Case (to be extended) and drop on another Use Case.
Business actors are Passenger, Tour Guide, Minor (Child), Passenger with
Special Needs (e.g.
with disabilities), all playing external roles in relation to airport
business.

Business use cases are Individual Check-In, Group Check-In (for groups of
tourists), Security Screening, etc. - representing business functions or
processes taking place in airport and serving the needs of passengers.

Business use cases Baggage Check-in and Baggage Handling extend Check-In
use cases, because passenger might have no luggage, so baggage check-in
and handling are optional.
```

Conclusion: The use case diagram was designed successfully by following the steps described above

6.2 Use case diagram for Transportation and Logistics

Design and development a use case diagram for Transportation and Logistics. A use case diagram is a visual representation of the functional requirements of a system from the user's perspective. It helps to capture the interactions between users and the system, showing how users interact with the system to achieve specific goals. In the context of Transportation and Logistics, here's a simplified example of a use case diagram:

Hints

```
/**
```

```
 * Design the use case view with Use case diagram for Transportation and Logistics */
 Actors:
Customer
Dispatcher
Driver
Administrator
Use Cases:

Place Order:

Actors: Customer
Description: The customer can place a new transportation order, specifying the pickup and delivery locations, package details, and any special requirements.
Assign Driver:

Actors: Dispatcher
Description: The dispatcher assigns a driver to a specific transportation order based on factors such as proximity, availability, and type of cargo.
Track Shipment:

Actors: Customer, Driver
Description: Both customers and drivers can track the status and location of a shipment in real-time.
Update Order Status:

Actors: Driver
Description: The driver updates the status of the transportation order, indicating when the package has been picked up, in transit, and delivered.
Manage Fleet:

Actors: Administrator
Description: The administrator can add, remove, or update information about vehicles and drivers in the fleet.
Generate Reports:

Actors: Administrator
Description: The administrator can generate reports related to order fulfillment, delivery times, and other relevant metrics.
Cancel Order:

Actors: Customer
Description: The customer can cancel a transportation order before it has been assigned to a driver.
Optimize Routes:

Actors: Dispatcher
Description: The dispatcher can optimize delivery routes to improve efficiency and reduce delivery times.
Manage Inventory:

Actors: Administrator
```

```
Description: The administrator can manage inventory levels and update
information about available stock.
```

Conclusion: The use case diagram was designed successfully by following the steps described above

6.3 Use case diagram for Credit Card Processing System

Design and development a use case diagram for Ticket Vending Machine. Credit Card Processing System (aka Credit Card Payment Gateway) is a subject, i.e. system under design or consideration. Primary actor for the system is a Merchant's Credit Card Processing System. The merchant submits some credit card transaction request to the credit card payment gateway on behalf of a customer. Bank which issued customer's credit card is actor which could approve or reject the transaction. If transaction is approved, funds will be transferred to merchant's bank account.

Hints

```
/**
 * Design the usecase view with Use case diagram for Credit card
Processing System*/

Authorize and Capture use case is the most common type of credit card
transaction. The requested amount of money should be first authorized by
Customer's Credit Card Bank, and if approved, is further submitted for
settlement.
During the settlement funds approved for the credit card transaction are
deposited into the Merchant's Bank account.

In some cases, only authorization is requested and the transaction will
not be sent for settlement. In this case, usually if no further action is
taken within some number of days, the authorization expires. Merchants can
submit this request if they want to verify the availability of funds on
the customer's credit card, if item is not currently in stock, or if
merchant wants to review orders before shipping.

Capture (request to capture funds that were previously authorized) use
case describes several scenarios when merchant needs to complete some
previously authorized transaction – either submitted through the payment
gateway or requested without using the system, example using voice
authorization.
```

Conclusion: The use case diagram was designed successfully by following the steps described above

6.4 Use case diagram for Radiology Diagnostic Reporting

Design and development a use case diagram for Radiology Diagnostic Reporting. The Simple Image and Numeric Report (SINR) [IHE Radiology Integration Profile, IHE RAD TF Vol. 1, Rev. 11.0] facilitates the growing use of digital dictation, voice recognition, and specialized reporting packages, by separating the functions of diagnostic reporting into discrete actors for creation, Automation, storage and report viewing. Separating these functions while defining transactions to exchange the reports between them enables a vendor to include one or more of these functions in an actual system. The IHE Technical Framework (TF)

identifies IHE Actors - functional components of a healthcare enterprise from the point of view of their interactions in distributed healthcare environment.

Hints

```
/**
 * Design the use case view with Use case diagram for Radiology Diagnostic
Reporting System*/

Radiology diagnostic reporting UML use case diagram example
for Simple Image and Numeric Report (SINR) IHE Radiology Integration
Profile

The Simple Image Report allows documents with multiple sections (with
headings) containing report text and references to relevant images. Some
text items of these documents may also be related to specific images. This
allows a reading physician to identify one or more images from which their
conclusions were inferred.

Reports are processed and modified by the Report Manager IHE actor. This
involves adding and changing report data as well as verifying draft
reports.

In the Report Issuing transaction, the Report Manager transmits either an
unchanged draft DICOM SR or a new modified DICOM SR to the Report
Repository.

The Report Repository provides permanent storage of DICOM Structured
Reports. It also allows reports to be queried and retrieved throughout the
enterprise by Report Readers.

The External Report Repository Access actor is a gateway to obtain other
enterprise department reports, such as Laboratory and Pathology, from
within the Imaging department.

In the Structured Report Export [RAD-28] transaction, the Report Manager
transmits verified Structured Reports as unsolicited HL7 observations to
the Enterprise Report Repository
```

Conclusion: The use case diagram was designed successfully by following the steps described above

6.5 Use case diagram for Software Protection and Licensing

Design and development a use case diagram for Software Protection and Licensing. Sentinel License Development Kit (Sentinel LDK) is a Software Digital Rights Automation (DRM) solution by SafeNet Inc. that delivers strong copy protection, protection for Intellectual Property (IP), and secure and flexible licensing. Sentinel LDK separates licensing and production processes (implemented with Sentinel EMS) from the software protection process (implemented with Sentinel Licensing API or Sentinel LDK Envelope). Sentinel EMS is a web-based graphical application provided as part of Sentinel LDK that is used to perform a range of functions required to manage the licensing, production, distribution, customer support, and maintenance of protected applications. This application is a role-based application

designed to manage the business activities required to implement and maintain Sentinel LDK in the organization which needs to protect its software. Sentinel EMS Server maintains a database containing a wide range of information, including data related to product features, licenses, sales, orders, and customers.

Hints

```
/**
 * Design the use case view with Use case diagram Software protection and
Licensing*/

The Sentinel EMS handles three major workflows:
license planning,
order processing and production, and activation of trial software.

Product Manager defines Features and Products.
Each Product has one or more Features. After Features and Products have
been defined in Sentinel EMS, entitlements can be processed and produced
using the Production group of functions. Users assigned the Development
role can fulfil one of the following development-related.

activities:
Generate bundles of Provisional (Trial) Products
Generate a customized Sentinel LDK Run-time Environment (RTE) installer
file
Customize the Sentinel Remote Update System utility (RUS utility)
Entitlement Manager defines and manages customers, and also enters and
manages entitlements.

An entitlement is the execution of a customer order for Sentinel LDK
items, and can be either an order for Products to be supplied with one or
more Sentinel protection keys, or a Protection Key Update that specifies
changes to be made to the license terms and/or data stored in Sentinel
protection keys that have already been deployed.
Customer Services role can manage customers the same way as Entitlement
Manager does, and can also manage Product activation.

For entitlements that generate Product Keys, the customer receives an
email from Sentinel EMS that contains the keys. The customer is able to
log in to the EMS Customer Portal using the Product Key in order to
activate the Product.
```

Conclusion: The use case diagram was designed successfully by following the steps described above

7. Behavioral Modeling-Sequence Diagram

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. Sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time., the lifeline is

represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.

7.1. Sequence diagram for Traffic Monitoring System

Design and develop a sequence diagram for Airline Reservation System. Creating a complete sequence diagram for a Traffic Monitoring System involves understanding the specific interactions between different components or actors in the system. However, I can provide you with a basic example to give you an idea of how a sequence diagram might look for a simple Traffic Management System. Keep in mind that the actual diagram may vary based on the specific requirements and architecture of your system..

Hints

```
/**
 * Design the interaction view with sequence diagram for Traffic
Monitoring System */

To design a Sequence Diagram:
Select first an element where a new Sequence Diagram to be contained as a
child.
Select Model | Add Diagram | Sequence Diagram in Menu Bar or select Add
Diagram | Sequence Diagram in Context Menu.

To design a Lifeline:
Select Lifeline in Toolbox.
Drag on the diagram as the size of Lifeline.

To design a Lifeline from a Classifier (Class, Interface, etc.) by Drag-
and-Drop:
Drag a Classifier from Explorer.

Drop on the diagram.
To design a Message (or Self Message):
Select Message (or Self Message) in Toolbox.
Drag from a Lifeline and drop on another Lifeline. (Just click on a
Lifeline if you want to design a self-message.)

To design an Endpoint:
Select Endpoint in Toolbox.
Click at the position on the diagram
sequence diagram is able to show programmers and readers about the
sequence of messages between the actor and the objects.

Modeling Flows of Control by Time Ordering

The conditions and interactions are emphasized, These interactions ae
essential for the Airline Reservation System development.

as an airline or computer reservation system, stores flight-related data
such as schedules, rates, and rules for each booking class, passenger name
records (PNRs), and e-tickets, among other things.
```

Conclusion: The sequence diagram was designed successfully by following the steps described above

7.2 Sequence diagram for Machine Learning-Based Fraud Detection for a Financial Institution

Design and develop a sequence diagram for ATM System. Creating a detailed sequence diagram for a Machine Learning-Based Fraud Detection system in a financial institution involves multiple steps. The following is a simplified representation of the sequence of interactions between various components in such a system:

- The user logs in through the frontend, which sends a login request to the backend server.
- The backend server validates the user's credentials by querying the database.
- Upon successful login, the user performs a transaction through the frontend.
- The frontend sends a transaction request to the backend server.
- The backend server utilizes a machine learning model to analyze the transaction data and determine the probability of fraud.
- The machine learning model returns the fraud probability to the backend server.
- Based on the fraud probability, the backend server decides whether to approve or deny the transaction.
- The backend server updates the transaction records in the database.
- The frontend receives the transaction status from the backend server and displays it to the user.
- When the user logs out, a similar process occurs to update the user's session status and confirm the logout.

Note: This is a simplified representation, and the actual implementation may involve more details and interactions based on the specific architecture and requirements of the fraud detection system in the financial institution.

Hints

```
/**
 * Design the interaction view with sequence diagram for Machine Learning-
Based Fraud Detection for a Financial Institution*/

Sequence diagram is able to show programmers and readers about the
sequence of messages between the actor and the objects.

Modeling Flows of Control by Time Ordering
Customers can use an ATM to access their bank deposits or credit accounts.
They can also perform financial operations, including cash withdrawals,
balance checks, and credit transfers to and from mobile phones.
```

Conclusion: The sequence diagram was designed successfully by following the steps described above.

7.3 Sequence diagram for IoT platform in a smart home company

Creating a complete and accurate sequence diagram for an IoT platform in a smart home company would depend on the specific functionalities and interactions involved. However, I can provide you with a basic outline that you can customize based on your requirements. In this example, I'll illustrate a simplified sequence diagram for a smart home IoT platform that involves user authentication, device control, and data processing.

Hints

```
/**
 * Design the interaction view with sequence diagram for IoT platform in a
smart home company */
Participants:
- User
- Smart Home IoT Platform
- Mobile App
- Smart Home Devices (e.g., thermostat, lights)

Note: The sequence diagram is a high-level representation, and the actual
interactions may involve more details and steps.

User -> Mobile App: Open Smart Home App
activate Mobile App
Mobile App -> Smart Home IoT Platform: Authenticate User
activate Smart Home IoT Platform
Smart Home IoT Platform -> User: Authentication Success
deactivate Mobile App
User -> Mobile App: View Home Dashboard
activate Mobile App
Mobile App -> Smart Home IoT Platform: Request Home Data
activate Smart Home IoT Platform
Smart Home IoT Platform -> Smart Home Devices: Retrieve Device Status
activate Smart Home Devices
Smart Home Devices --> Smart Home IoT Platform: Device Status
deactivate Smart Home Devices
Smart Home IoT Platform --> Mobile App: Home Data
deactivate Smart Home IoT Platform
Mobile App -> Smart Home IoT Platform: Control Smart Home Device
activate Smart Home IoT Platform
Smart Home IoT Platform -> Smart Home Devices: Send Control Command
activate Smart Home Devices
Smart Home Devices --> Smart Home IoT Platform: Acknowledge Command
deactivate Smart Home Devices
Smart Home IoT Platform --> Mobile App: Command Acknowledgment
deactivate Smart Home IoT Platform
Mobile App -> Smart Home IoT Platform: Logout
activate Smart Home IoT Platform
Smart Home IoT Platform -> User: Logout Success
deactivate Mobile App
deactivate Smart Home IoT Platform
```

Conclusion: The sequence diagram was designed successfully by following the steps described above

7.4 Sequence diagram for Blockchain-Based Supply Chain Tracking system

Creating a complete sequence diagram for a Blockchain-Based Supply Chain Tracking system requires a detailed understanding of the specific components, actors, and interactions involved in the system. However, I can provide you with a simplified example to give you an

idea of how a sequence diagram for such a system might look. Please note that this is a generic representation, and you may need to adapt it to your specific use case.

Hints

```
/**
 * Design the interaction view with sequence diagram for Blockchain-Based Supply Chain Tracking system */

Scenario: Updating Supply Chain Information on the Blockchain
Actor Roles:
Manufacturer
Distributor
Retailer
Blockchain Network
```

Conclusion: The sequence diagram was designed successfully by following the steps described above

7.5 Sequence diagram for Social Media Analytics Platform

Creating a sequence diagram for a Social Media Analytics Platform involves illustrating the interactions and communication flow between different components or actors in the system. Below is a simplified example of a sequence diagram for a Social Media Analytics Platform n this diagram:

Hints

```
/**
 * Design the interaction view with sequence diagram for Social Media Analytics Platform */
The User Interface initiates the analytics process by sending a request for analytics.
The request is received by the Analytics Controller, which processes the request.
The Analytics Controller interacts with the Data Retrieval Module to retrieve social media data.
The Data Retrieval Module communicates with the Social Media API to query the social media server for relevant data.
The social media API interacts with the Social Media Server to fetch the required data.
The fetched data is then passed to the Data Processing and Analysis module, where the actual analytics take place.
Once the analysis is complete, the platform generates an analytics report.
The analytics report is sent back to the User Interface.
Finally, the User Interface displays the analytics report to the user.
Note that this is a high-level and simplified representation. Depending on the complexity of your Social Media Analytics Platform, you may need to include more details and interactions in your sequence diagram.
```

Conclusion: The sequence diagram was designed successfully by following the steps described above

7.6 Sequence diagram for Face Recognition Attendance System

The sequence diagram of a face recognition attendance system is used to show how the parts of a system work together to make the online shopping operate. The Sequence Diagram for face recognition attendance system represents the scenario and the messages that must be passed between objects. It's an interaction diagram that shows how activities are carried out, including when and how messages are send.

Hints

```
/**
 * Design the interaction view with sequence diagram for Face Recognition
Attendance System */

The face recognition attendance system sequence diagram has several boxes
(objects) which are the face recognition (device), the system records, the
attendance information, and the visitor's records. Its user could be the
employer and employees (establishments) and students (schools), and the
messages have a flow showing the alternative in every decision.

A sequence diagram depicts the timeline and order in which messages are
sent between devices to carry out process functions.

Sequence diagrams are based on objects rather than classes
Finalize the purpose of the project
Place your users or objects
Add the lifelines in each user and object
Structure the sequence of messages (interaction)
Add the X symbol as the lifeline end
```

Conclusion: The sequence diagram was designed successfully by following the steps described above

8. Behavioral modeling- Collaboration diagram

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system.

8.1 Collaboration diagram for Stock Maintenance

Stock maintenance has the details about the product, purchase, sales and stock what are the stocks we had. The product details contain product code, Product name, Opening Stock and Prices. These details are maintained in database. In the purchasing function we must have the details about the store, quantity and also price.

Hints

```
/**
 * Design the interaction view with collaboration diagram for Stock
Maintenance */

Stock maintenance has the details about the product, purchase, sales and
stock what are the stocks we had.

To design a Communication Diagram:

Select first an element where a new Communication Diagram to be contained
as a child.
Select Model | Add Diagram | Communication Diagram in Menu Bar or select
Add Diagram | Communication Diagram in Context Menu.

The product details contain product code, Product name, Opening Stock and
Prices. These details are maintained in database.

To design a Connector (or Self Connector):
Select Connector (or Self Connector) in Toolbox.
Drag from a Lifeline and drop on another Lifeline. (Just click on a
Lifeline if you want to design a Self-Connector.)

To design a Forward Message:
Select Forward Message in Toolbox.
Click on a Connector.
To design a Reverse Message:
Select Reverse Message in Toolbox.
Click on a Connector.
In the purchasing function we must have the details about the store,
quantity and also price.

Following are the components of a component diagram that are enlisted
below:

1.Objects:
The representation of an object is done by an object symbol with its name
and class underlined, separated by a colon. In the collaboration diagram,
objects are utilized in the following ways:
The object is represented by specifying their name and class.
```

```
It is not mandatory for every class to appear.
A class may constitute more than one object.
```

Conclusion: The collaboration diagram was designed successfully by following the steps described above

8.2 Collaboration diagram for Agriculture Technology Software

Design and develop a collaboration diagram for Agriculture Technology Software, n the context of Agriculture Technology Software, the diagram would showcase the collaboration between different components or modules involved in the system. Keep in mind that creating a specific diagram requires knowledge of the software's architecture and design, so the following is a simplified example to give you an idea.

Let's consider a basic scenario where the Agriculture Technology Software involves three main components: User Interface (UI), Crop Monitoring Module (CMM), and Database. Each of these components may have several classes or objects that collaborate to provide functionality.

Hints

```
/**
 * Design the interaction view with collaboration diagram for Agriculture
Technology Software */

Following are the components of a component diagram that are enlisted
below:

1.Objects: The representation of an object is done by an object symbol
with its name and class underlined, separated by a colon.
In the collaboration diagram, objects are utilized in the following ways:
The object is represented by specifying their name and class.
It is not mandatory for every class to appear.

A class may constitute more than one object.
In the collaboration diagram, firstly, the object is designd, and then its
class is specified.
To differentiate one object from another object, it is necessary to name
them.

2.Actors: In the collaboration diagram, the actor plays the main role as
it invokes the interaction. Each actor has its respective role and name.
In this, one actor initiates the use case.

3.Links: The link is an instance of association, which associates the
objects and actors. It portrays a relationship between the objects through
which the messages are sent. It is represented by a solid line. The link
helps an object to connect with or navigate to another object, such that
the message flows are attached to links.
```

4.Messages: It is a communication between objects which carries information and includes a sequence number, so that the activity may take place. It is represented by a labeled arrow, which is placed near a link. The messages are sent from the sender to the receiver, and the direction must be navigable in that particular direction. The receiver must understand the message.

Conclusion: The collaboration diagram was designed successfully by following the steps described above.

8.3 Collaboration diagram for Cybersecurity Software for Banking

Design and develop a collaboration diagram for Cybersecurity Software for Banking. n the context of cybersecurity software for banking, the diagram would represent the collaboration between various modules or components involved in ensuring the security of banking systems. Please note that creating a detailed collaboration diagram would depend on the specific functionalities and components of the cybersecurity software you have in mind. However, I can provide you with a high-level example to give you an idea.

In a cybersecurity software system for banking, you might have components such as:

User Interface (UI): Represents the interface through which bank employees or administrators interact with the cybersecurity software.
Authentication Module: Handles user authentication and authorization processes.
Intrusion Detection System (IDS): Monitors network traffic and detects potential security threats.
Firewall: Controls and monitors incoming and outgoing network traffic based on predetermined security rules.
Encryption Module: Manages the encryption and decryption of sensitive data to protect it from unauthorized access.
Logging and Auditing Module: Records security events and provides audit trails for monitoring and analysis.
Virus and Malware Protection: Protects against malicious software by scanning files and monitoring system behavior.
Database Security Module: Ensures the security of the banking database, protecting sensitive customer information.

Hints

```
/**
 * Design the interaction view with collaboration diagram for Passport automation system */

Following are the components of a component diagram that are enlisted below:

1.Objects: The representation of an object is done by an object symbol with its name and class underlined, separated by a colon.
In the collaboration diagram, objects are utilized in the following ways:
The object is represented by specifying their name and class.
It is not mandatory for every class to appear.
A class may constitute more than one object.
```

```
In the collaboration diagram, firstly, the object is designd, and then its
class is specified.

To differentiate one object from another object, it is necessary to name
them.

2.Actors: In the collaboration diagram, the actor plays the main role as
it invokes the interaction. Each actor has its respective role and name.
In this, one actor initiates the use case.

3.Links: The link is an instance of association, which associates the
objects and actors. It portrays a relationship between the objects through
which the messages are sent. It is represented by a solid line. The link
helps an object to connect with or navigate to another object, such that
the message flows are attached to links.

4.Messages: It is a communication between objects which carries
information and includes a sequence number, so that the activity may take
place. It is represented by a labeled arrow, which is placed near a link.
The messages are sent from the sender to the receiver, and the direction
must be navigable in that particular direction. The receiver must
understand the message.
```

Conclusion: The collaboration diagram was designed successfully by following the steps described above.

8.4 Collaboration diagram for E-commerce platform enhancement

Design and develop a collaboration diagram for E-commerce platform enhancement. In the context of an E-commerce platform enhancement, the collaboration diagram can illustrate the communication and collaboration between various modules or components involved in the enhancement.

Here's a simplified example of a collaboration diagram for an E-commerce platform enhancement. In this example, let's consider a scenario where a new feature is being added to allow users to write and submit product reviews:

Hints

```
/**
 * Design the interaction view with collaboration diagram for E-commerce
platform enhancement */


2. Customers will use web based interface to browse books based on
categories, search books using keywords. Initially only the title and
author of the book(s) are displayed, on click other attributes are
displayed. Customers can buy books using their e-purse. The store also
displays the number of copies of the book left in stock. Out of stock
books cannot be purchased immediately, but can be ordered.

3. Customers design accounts in the book store. Each account contains
customer profile information: name, age, geographical location, categories
of interest, email. Each account has an e-purse. Customers can specify the
amount of money to be deposited with the e-purse. Profile and e-purse
```

information can be updated by the customer. Customers will login to the book store using an account name and password.

4. All online sales data are recorded in the database with timestamp.

5. Owner of the bookstore can give requisition for buying of books to publishers based on the amount of stock remaining. For each book the owner maintains a stock which is at least the number of copies of the book sold over last 3 months. Books ordered by some customers are immediately requisitioned. Requisitions are placed in a requisition table. The publishers inspect the table on the 1st of every month and immediately supply the books. Once a book is supplied it is cleared from the requisition table.

Conclusion: The collaboration diagram was designed successfully by following the steps described above.

8.5 Collaboration diagram for Weather Forecasting System

Design and develop a collaboration diagram for Weather Forecasting System. A collaboration diagram, also known as a communication diagram, illustrates how objects interact to achieve a particular goal. In the case of a Weather Forecasting System, you might have various components and entities collaborating to gather, process, and display weather information

Hints

```
/**
 * Design the interaction view with collaboration diagram for Weather Forecasting System */

Communication diagrams are also interaction diagrams. They convey the same information as sequence diagrams, but they focus on object roles instead of the times that messages are sent. In a sequence diagram, object roles are the vertices and messages are the connecting links.

The notation for a communication/Collaboration diagram, objects (actors in use cases) are represented by rectangles. In the example (generic communication diagram):

•The objects are Object1, Object2, Object..., ObjectN-1 ..., and Object N.
•Messages passed between objects are represented by labeled arrows that start with the sending object (actor) and end with the receiving object.

•The sample messages passed between objects are labeled 1: message1, 2: message2, 3: message3, etc., where the numerical prefix to the message name indicates its order in the sequence.

•Object1 first sends Object2 the message message1, Object2 in turn sends ObjectN-1 the message message2, and so on.
•Messages that objects send to themselves are indicated as loops (e.g., message message5).
```

```
Communication diagrams are also interaction diagrams. They convey the same
information as sequence diagrams, but they focus on object roles instead
of the times that messages are sent. In a sequence diagram, object roles
```

Conclusion: The collaboration diagram was designed successfully by following the steps described above

8.6 Collaboration Diagram for Music Streaming Service

Design and develop a collaboration diagram for Music Streaming Service. A collaboration diagram, also known as a communication diagram, illustrates the interactions and relationships among different elements in a system. In the case of a Music Streaming Service, these elements might include users, the streaming server, the music database, and other components.

The "User" initiates a playback request, which is sent to the "Streaming Server."
The "Streaming Server" then communicates with the "Music Database" to retrieve the requested music.
The "Music Database" sends the music data back to the "Streaming Server."
The "Streaming Server" streams the music data to the "User" for playback

Hints

```
/**
 * Design the interaction view with collaboration diagram for Order
Processing system */

Communication diagrams are also interaction diagrams. They convey the same
information as sequence diagrams, but they focus on object roles instead
of the times that messages are sent. In a sequence diagram, object roles
are the vertices and messages are the connecting links.

The notation for a communication/Collaboration diagram, objects (actors in
use cases) are represented by rectangles. In the example (generic
communication diagram):

•The objects are Object1, Object2, Object..., ObjectN-1 ..., and Object N.
•Messages passed between objects are represented by labeled arrows that
start with the sending object (actor) and end with the receiving object.

•The sample messages passed between objects are labeled 1: message1, 2:
message2, 3: message3, etc., where the numerical prefix to the message
name indicates its order in the sequence.

•Object1 first sends Object2 the message message1, Object2 in turn sends
ObjectN-1 the message message2, and so on.
•Messages that objects send to themselves are indicated as loops (e.g.,
message message5).
```

Conclusion: The collaboration diagram was designed successfully by following the steps described above

9. Behavioral modeling- Activity diagram

The activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities.

The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc.

It is also termed as an object-oriented flowchart. It encompasses activities composed of a set of actions or operations that are applied to model the behavioral diagram.

9.1 Activity Diagram for Garage Parking Solution

Design and develop activity diagram for Garage Parking Solution. Creating an activity diagram for a Garage Parking Solution involves representing the flow of activities and actions that occur within the system. Below is a simplified example of an activity diagram for a Garage Parking Solution. This diagram assumes a basic process flow, and you may need to tailor it to fit the specific requirements of your system.
This activity diagram outlines a simple process for a Garage Parking Solution, including detecting a vehicle, checking available parking spaces, displaying the available spaces, user selection, reservation, parking the vehicle, and updating the availability status. The process ends once the vehicle is parked. Note that this is a basic representation, and real-world systems may involve more detailed steps and decision points.

Hints

```
/**
 * Design the activity view with activity diagram for Garage Parking
Solution */

To design an Activity Diagram:

Select first an element where a new Activity Diagram to be contained as a
child.
Select Model | Add Diagram | Activity Diagram in Menu Bar or select Add
Diagram | Activity Diagram in Context Menu.
To design an Action:
Select Action in Toolbox.
Drag on the diagram as the size of Action.

To add a Trigger:
Select an Action.
Select Model | Add | Trigger in Menu Bar or Add | Trigger in Context Menu.
To design an Initial Node:
Select Initial in Toolbox.
Click at the position on the diagram.

To design an Activity Final Node:
Select Activity Final in Toolbox.
Click at the position on the diagram.
```

```
Following are the rules that are to be followed for drawing an activity diagram:
1.A meaningful name should be given to each and every activity.
2.Identify all of the constraints.
3.Acknowledge the activity associations
```

Conclusion: The activity diagram was designed successfully by following the steps described above

9.2 Activity diagram for Greenhouse automation

Design and develop activity diagram for Greenhouse automation. Creating a complete activity diagram for greenhouse automation involves understanding the specific functionalities and processes involved in the system. Below is a simplified example of an activity diagram for a greenhouse automation system. This diagram focuses on the main activities and interactions within the system.

Explanation of key activities:

Start/Initialize System: Represents the starting point of the greenhouse automation system where the initialization processes occur.

Sensor Data Collection: Involves collecting data from various sensors such as temperature, humidity, light intensity, and soil moisture.

Analyze Sensor Data: Analyzes the collected sensor data to determine the current environmental conditions within the greenhouse. Decision-making processes may occur here.

Control Actuators: Based on the analysis, the system adjusts actuators to control the greenhouse environment. This includes activities like adjusting temperature, regulating humidity, controlling light, and activating the watering system.

Monitor System Status: Monitors the overall status of the system, displaying information to users and generating alerts if necessary.

Stop/Shutdown System: Represents the termination or shutdown of the greenhouse automation system.

End: The endpoint of the activity diagram.

Please note that this is a simplified example, and in a real-world scenario, you may need to include more details and specific activities based on the features and complexity of your greenhouse automation system.

Hints

```
/**
 * Design the activity view with activity diagram for Ticket Vending Machine system */
To design an Activity Diagram:
Select first an element where a new Activity Diagram to be contained as a child.
Select Model | Add Diagram | Activity Diagram in Menu Bar or select Add Diagram | Activity Diagram in Context Menu.

To design an Action:
Select Action in Toolbox.
Drag on the diagram as the size of Action. Add Input Pin: Add an input pin.
Add Output Pin: Add an output pin.
```

```
Add Note: Add a linked note.
Add Constraint: Add a constraint.
Add Trigger Event: Add a trigger event.
Add Outgoing Control Flow: Add an outgoing control flow with an action.
Add Incoming Control Flow: Add an incoming control flow with an action.
Add Outgoing Object Flow: Add an outgoing object flow with an object node.
Add Incoming Object Flow: Add an incoming object flow with an object node.
Add Decision: Add a decision with two additional actions.
Add Merge: Add a merge with two additional actions.
Add Fork: Add a fork with two additional actions.
Add Join: Add a join with two additional actions.
Add Initial Node: Add an initial node with a connected control flow.
Add Final Node: Add a final node with a connected control flow.

To add a Trigger:
Select an Action.
Select Model | Add | Trigger in Menu Bar or Add | Trigger in Context Menu.
To design an Initial Node:
Select Initial in Toolbox.
Click at the position on the diagram.
To design an Activity Final Node:
Select Activity Final in Toolbox.
Click at the position on the diagram.
To design an Activity Final Node:
Select Activity Final in Toolbox.
Click at the position on the diagram.
To design a Fork Node:
Select Fork in Toolbox.
Drag on the diagram as the size of Fork.
To design a Join Node:
Select Join in Toolbox.
Drag on the diagram as the size of Join.
To design a Merge Node:
Select Merge in Toolbox.
Click at the position on the diagram.
To design a Decision Node:
Select Decision in Toolbox.
Click at the position on the diagram.
```

Conclusion: The activity diagram was designed successfully by following the steps described above

9.3 Activity diagram for a chatbot system design

Creating a detailed activity diagram for a chatbot system design involves multiple steps and interactions. Below is a simplified example of an activity diagram for a basic chatbot system. Please note that the actual design may vary based on specific requirements and features of your chatbot system.
Here's a brief explanation of each step:

Hints

```
/**
```

```
 * Design the activity view with activity diagram for electronic prescription service
*/
User Interaction: This is where the user initiates interaction with the chatbot, typically by sending a message or query.
Chatbot System Initialization: The chatbot system initializes and prepares for processing the user's input.
Receive User Input and Process: The chatbot system receives the user's input and processes it to understand the user's intent and extract relevant entities.
Determine Intent and Entities: The system determines the user's intent (what the user wants) and extracts entities (specific information) from the user's input.
Query Knowledge Base or External APIs: Based on the user's intent and entities, the chatbot system may need to query a knowledge base or external APIs to gather relevant information.
Generate Response and Output to User: The chatbot generates a response based on the gathered information and communicates it back to the user.
User Interaction with Response: The user interacts with the response provided by the chatbot, and the cycle can repeat based on further user inputs.
This is a high-level overview, and the actual activity diagram could include more detailed steps, decision points, and error handling depending on the complexity of your chatbot system
```

Conclusion: The activity diagram was designed successfully by following the steps described above

9.4 Activity diagram for Document Automation Process

Design and develop activity diagram for Document Automation Process. Document Automation Process has some kind of formal and properly communicated document Automation process is usually required in any major corporation especially under a regulatory compliance. A document goes through different state or stages - it is designed, reviewed, updated, approved, and at some point, archived.

Hints

```
 /**
 * Design the activity view with activity diagram for Document Automation Process
*/

Different roles participating in this process are Author, Reviewer, Approver, and Owner. These roles are represented on the diagram by partitions rendered as horizontal "swimlanes”.

This activity diagram should show responsibilities of different roles and flow or sequence of document changes. Alternative type of diagram - state machine diagram - could also be used in this case to show how document changes its state over time.
```

Conclusion: The activity diagram was designed successfully by following the steps described above.

9.5 Activity diagram for to resolve an issue in a software design

Design and develop activity diagram for resolve an issue in a software design. which shows how to resolve an issue in a software design. After ticket is designed by some authority and the issue is reproduced, issue is identified, resolution is determined, issue is fixed and verified, and ticket is closed, if issue was resolved.

This example does not use partitions, so it is not very clear who is responsible for fulfilling each specific action.

Hints

```
 /**
 * Design the activity view with activity diagram for electronic
prescription service
*/

Different roles participating in this process are Author, Reviewer,
Approver, and Owner. These roles are represented on the diagram by
partitions rendered as horizontal "swimlanes”.

This activity diagram should show responsibilities of different roles and
flow or sequence of document changes.
Alternative type of diagram - state machine diagram - could also be used
in this case to show how document changes its state over time.

The Document object is not the only object node shown on this activity
diagram. There is also another object - Change Request, an object which is
used to pass changes to the document requested by Reviewer. State diagram
for the Document will only show the document states and transitions, so
activity diagram is useful when different roles and several object nodes
are involved.
```

Conclusion: The activity diagram was designed successfully by following the steps described above

10. Behavioral modeling- State chart diagram

A state–chart diagram shows a state machine that depicts the control flow of an object from one state to another. A state machine portrays the sequences of states which an object undergoes due to events and their responses to events.

State–Chart Diagrams comprise of –States: Simple or Composite, Transitions between states Events causing transitions, Actions due to the events, State-chart diagrams are used for modeling objects which are reactive in nature.

10.1 State chart diagram for an AI-based image recognition system

Design and develop state chart diagram for AI-based image recognition system. A state chart diagram is a type of diagram used in computer science and engineering to describe the behavior of a system. For an AI-based image recognition system, the state chart diagram can represent the different states the system can be in and the transitions between those states. Below is a simplified example of a state chart diagram for an AI-based image recognition system:

Hints

```
 /**
 * Design the state machine view with State chart diagram for AI-based
image recognition system
*/
Explanation:
Idle: The initial state when the system is waiting for an image to be
received.
Image Received: The system transitions to this state when it receives an
image for processing.
Image Processing: Represents the state where the system is actively
processing the received image using AI algorithms for object recognition.
Object Recognition: The state where the actual object recognition is
taking place.
Recognition Done: The final state indicating that the image recognition
process is completed, and the system has identified objects in the image.
The transitions between states are labeled with the events or conditions
that trigger the transitions. For example, the transition from "Idle" to
"Image Received" is triggered by the event of receiving an image.
Similarly, the transition from "Object Recognition" to "Recognition Done"
might be triggered by the completion of the object recognition process.
```

Conclusion: The state chart diagram was designed successfully by following the steps described above

10.2 State chart diagram for Augmented Reality (AR) Game:

Design and develop state chart diagram for Augmented Reality (AR) Game. a state chart diagram for an Augmented Reality (AR) game involves illustrating the various states and transitions that the game can go through.

In this diagram:

Start: Represents the initial state of the game.

Loading: Represents the state where the game is loading resources, initializing AR components, etc.

Active: Represents the main state when the game is actively being played.

Paused: Represents the state when the game is paused, perhaps due to user input or other events.

Game Over: Represents the state when the game has ended, and the player has either won or lost.

Transitions between states are depicted by arrows. For example:

Transition from Start to Loading indicates the initialization phase.

Transition from Loading to Active indicates the completion of loading and the start of the game.

Transition from Active to Paused may occur when the player pauses the game.

Transition from Active to Game Over occurs when the game ends.

Hints

```
 /**
 * Design the State machine view with state chart diagram for Bank ATM
behavioral
*/
State Machine Diagrams. A state machine diagram models the behavior of a
single object, specifying the sequence of events that an object goes
through during its lifetime in response to events. As an example, the
following state machine diagram shows the states that a door goes through
during its lifetime.
States
A state is denoted by a round-cornered rectangle with the name of the
state written inside it.
Initial and Final States
The initial state is denoted by a filled black circle and may be labeled
with a name. The final state is denoted by a circle with a dot inside and
may also be labeled with a name.
Transitions
Transitions from one state to the next are denoted by lines with
arrowheads. A transition may have a trigger, a guard and an effect.
State Actions
In the transition an effect was associated with the transition. If the
target state had many transitions arriving at it, and each transition had
the same effect associated with it, it would be better to associate the
effect with the target state rather than the transitions. This can be done
by defining an entry action for the state. The diagram below shows a state
with an entry action and an exit action.
Self-Transitions
A state can have a transition that returns to itself, as in the following
diagram. This is most useful when an effect is associated with the
transition.
```

Conclusion: The state chart diagram was designed successfully by following the steps described above

10.3 State chart diagram for Predictive Analytics System

Draw and develop a state chart diagram for Predictive Analytics System. a state chart diagram for a Predictive Analytics System involves representing the various states and transitions that the system can undergo. State chart diagrams are part of Unified Modeling Language (UML) and are used to model the dynamic aspects of a system.

In this diagram:

Initializing: The system is in the process of initializing its components.

Idle: The system is ready and waiting for user input or a trigger to start processing.

Data Collection: The system collects data from various sources.

Data Preprocessing: The collected data undergoes preprocessing to clean and transform it for analysis.

Model Training: The system trains a predictive model using the preprocessed data.

Model Evaluation: The trained model is evaluated for its performance.

Prediction: The system uses the trained model to make predictions on new data.

Results Display: The results of predictions are displayed to the user.

Error Handling: If any errors occur during the process, the system handles them appropriately.
System Error: In case of critical errors, the system enters an error state.

Hints

```
 /**
 * Design the State machine view with state chart diagram for Predictive
Analytics System */
To design a State chart Diagram:
Select first an element where a new State chart Diagram to be contained as
a child.
Select Model | Add Diagram | State chart Diagram in Menu Bar or select Add
Diagram | State chart Diagram in Context Menu.
State
To design a Simple State:
Select Simple State in Toolbox.
Drag on the diagram as the size of Simple State.
To design a Composite State:
Select Composite State in Toolbox.
Drag on the diagram as the size of Composite State.
To design a Submachine State:
Select Submachine State in Toolbox.
Drag on the diagram as the size of Submachine State.
Select a State Machine in Element Picker Dialog.
To design an Orthogonal State:
Select Orthogonal State in Toolbox.
Drag on the diagram as the size of Orthogonal State.
Internal Transition
To add an Internal Transition:
Select a State.
Select Add Internal Transition button in Quick Edit.
To design a Transition (or Self Transition):
Select Transition (or Self Transition) in Toolbox.
Drag from a State and drop on another State. (Just click on a State if you
want to design a Self-Transition.
```

Conclusion: The state chart diagram was designed successfully by following the steps described above.

10.4 State Chart diagram for Educational Learning Management System

Design and develop a state chart diagram for an Educational Learning Management System. In the context of an Educational Learning Management System (LMS), a state chart diagram can represent the various states and transitions that the system can undergo.

Hints

```
  /**
 * Design the state machine view with state chart diagram for Educational
Learning Management System
*/

Explanation of States and Transitions:
Idle State:
Initial state when the system is not actively engaged.
Transition to the "Login" state occurs when a user attempts to log in.
```

```
Login State:
Represents the process of user authentication.
Transitions to "Dashboard" upon successful login or back to "Idle" if
login fails.
Dashboard State:
User is presented with the main dashboard.
Transitions to various states based on user interactions.
Course Selection State:
Represents the process of selecting a course from the available options.
Transitions to "Course Content" when a course is selected.
Course Content State:
User is engaged in the selected course.
Transitions to "Quiz" or "Assignment" states based on user choice.
Quiz State:
User is taking a quiz.
Transitions to "Quiz Results" upon completion.
Assignment State:
User is working on an assignment.
Transitions to "Assignment Submission" upon completion.
Quiz Results State:
Displays the results of the quiz to the user.
Transitions back to "Course Content" or "Dashboard."
Assignment Submission State:
User submits an assignment.
Transitions to "Assignment Results" upon submission.
Assignment Results State:
Displays the results and feedback of the submitted assignment.
Transitions back to "Course Content" or "Dashboard."
Logout State:
User initiates logout.
Transitions back to "Idle" state.
These states and transitions are just a basic representation and may vary
based on the specific features and functionalities of the LMS. State chart
diagrams can be expanded to include more detailed states and transitions
to cover the entire range of functionalities in the system.
```

Conclusion: The state chart diagram was designed successfully by following the steps described above

10.5 State Chart diagram for Stock Trading Platform

Design and develop a Stock Trading Platform. In the context of a Stock Trading Platform, a state chart diagram can illustrate the various states that the system and its components can go through and the transitions between these states.

Explanation:

Initializing:

The initial state of the Stock Trading Platform. It transitions to the "Ready" state when the system is ready for operation.

Ready:

The platform is ready to accept trading requests. It can transition to the "Trading" state when a user initiates a trade.

Trading:

Represents the active trading state where buy/sell orders are processed. It can transition to the "Suspended" state in case of system maintenance or unexpected issues.
Suspended:
The platform temporarily halts trading activities. It can transition back to the "Trading" state when the suspension is lifted or transition to "Closed" if the platform is shut down.
Closed:
The final state indicating that the Stock Trading Platform has been closed. This might happen due to the end of a trading day or other operational reasons.
This is a simplified representation, and the actual state chart for a Stock Trading Platform could be more detailed based on the specific features and requirements of the system.

Hints

```
  /**
 * Design the state machine view with state chart Hospital Automation
System
*/
To design a State chart Diagram:
Select first an element where a new State chart Diagram to be contained as
a child.
Select Model | Add Diagram | State chart Diagram in Menu Bar or select Add
Diagram | State chart Diagram in Context Menu.

States:    initiated,    doctor    availability,    consultation    attended,
consultation cancelled, diagnosis provided, invoice issued, invoice paid
```

Conclusion: The state chart diagram was designed successfully by following the steps described above.

11. Case study- An Automated Corporation

Automate a small manufacturing company. The resulting application will enable the user to take out a loan, purchase a machine, and over a series of monthly production runs, follow the performance of their company.

11.1 Design a Class diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

Hints

```
 /**
 * Design the Static view with class diagram for An automated company
*/
```

Identifying classes:
1.Company
2.Bank
3.Employees
4.Technical
5.non-technical
6.Customers
7.Shares
8.Machines
9.Goods
10.Customer
11.Market
12.Sales
13.Status
14.Profit
15.Loss
16.Rawmaterials
17.Department

Identifying relationships between classes:
Inheritance:
1.status contains profit
2.status contains loss
3.employee contains non-technical
Aggregation:
1.employee is a part of company
2.machines are part of company
3.department is a part of company
Association:
1.shares are provided by the company
2.bank provide loan for company
3.machines produce the goods
4.raw materials are given to the company
5.sales tells the status
6.customer buys goods through sales in market

Dependency:
1.market is dependent on goods
Identifying attributes:
1.company:name, code, address
2.bank:name, address, branch name
3.employee: department, hrswork
4.technical:name, id, hourswork
5.nontechnical:name, id, hourswork
6.shares: code
7.machines: type, cost, size, capacity
8.goods:name, code
9.customer:name
10.market:name, address
11.sales: quantity, quantity sold, quantity left
12.status: mention status
13.profit
14.loss
15.rawmaterials: type, code, quality type

```
16.department:name of dept, type

Identifying operations for classes:
Company:
1.check attendance
2.give salary
3.pay for salary
4.maintained machines
5.sell goods to markets
6.recruit employee
7.dismiss employee
8.note raw materials

Customer:
1.buy goods
2.pay money
Bank:
1.give loan
2.collect interest
3.collect loan

Employee:
1.work
2.take salary
3.maintain union

Non-technical:
1.work
12.status: mention status
13.profit
14.loss
15.rawmaterials: type, code, quality type
16.department:name of dept, type

Identifying operations for classes:
Company:
1.check attendance
2.give salary
3.pay for salary
4.maintained machines
5.sell goods to markets
6.recruit employee
7.dismiss employee
8.note raw materials

Customer:
1.buy goods
2.pay money

Bank:
1.give loan
2.collect interest
3.collect loan
```

```
Employee:
1.work
2.take salary
3.maintain union

Non-technical:
1.work
```

11.2 Design a Use case diagram

Use case diagram is to capture the dynamic aspect of a system. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

Hints

```
  /**
 * Design the use case view with use case diagram for an automated company
*/

Identifying actors
1. Company is hardware thing
2. Employee is a person in company
3. Technical employee is a person in company
4. non-technical employee is person in company
5. Bank is hardware thing that gives loan to company
6. Customer is a person who buys goods

Identifying use cases
1. Recruit employee
2. Give salary
3. Pay for raw materials
4. Maintain machines
5. Dismiss employee
6. Product goods
7. Work
8. Maintain union
9. Take salary
10. Give loan
11. Collect interest
12. Collect loan
13. Bye goods
14. Pay money
Identifying relationships
Generalization
1. Technical staff is a type of employee
2. non-technical staff is a type of employee

Association
1.Company recruits employee
2.Company gives salaries.
3.Company pays for raw materials
4.Company maintains machines
```

```
5.Company dismisses employee.
6.Company product goods
7.Employee works in company.
8.Employee takes salary.
9.Bank gives loan
10.Bank collect interest
11.Customer buy goods
12.Customer pays
```

11.3 Design an interaction view with Sequence diagram

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.

Hints

```
  /**
 * Design the interaction view with sequence diagram for an automated
company */
Identifying objects
1.c: company
2.s: shares
3.r: raw materials
4.m: machinery
5.sa: sales
6.c: customer
7.m: market
8.g: goods
9.e: employee
10.b: bank

Identifying messages
1.company plans for its development
2.company request loan from bank
3.bank check account of the company
4.company purchase machinery
5.company purchase raw materials
6.company receives order
7.company decides the quality of products
8.company manufactures the products
9.company analyze quality in market
10.customer buys from market
11.production analyses the sales depending on market
12.monthly production decides profit or loss depending on sales
```

11.4 Design an activity diagram

The activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities.

The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc.
It is also termed as an object-oriented flowchart. It encompasses activities composed of a set of actions or operations that are applied to model the behavioral diagram.

Hints

```
  /**
 * Design the activity view with activity diagram for an automated company
*/

Identification of activities:

1.company takes loan from banks
2.buy raw materials
Develop the activity diagram
```

11.5 Design a component diagram

A 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.

Hints

```
  /**
 * Design the implementation view with component diagram for an automated
company */

Identification of components:

1.Company
2.Bank
3.Employees
4.Technical
5.non-technical
6.Customers
7.Shares
8.Machines
9.Goods
10.Customer
11.Market
12.Sales
13.Status
14.Profit
```

```
15.Loss
16.Rawmaterials
17.Department

Develop the component diagram by using these component
```

11.6 Design a 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 designd 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.

Hints

```
  /**
 * Design the Deployment view with deployment diagram for an automated
company */

Identification of nodes:

1.Company
2.Bank
3.Employees
4.Customers
5.Shares
6.Machines
7.Sales
8.Department

Develop the deployment diagram by using these nodes
Identify the links among the nodes
```

11.7 Design a Collaboration diagram

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.

Hints

```
  /**
 * Design the interaction view with collaboration diagram for an automated
company */

Identifying objects
1.c: company
```

```
2.s: shares
3.r: raw materials
4.m: machinery
5.sa: sales
6.c: customer
7.m: market
8.g: goods
9.e: employee
10.b: bank

Identifying messages
1.company plans for its development
2.company request loan from bank
3.bank check account of the company
4.company purchase machinery
5.company purchase raw materials
6.company receives order
7.company decides the quality of products
8.company manufactures the products
9.company analyze quality in market
10.customer buys from market
11.production analyses the sales depending on market
12.monthly production decides profit or loss depending on sales
```

11.8 Design a State chart diagram

The state machine diagram is also called the State chart or State Transition diagram, which shows the order of states underwent by an object within the system. It captures the software system's behavior. It models the behavior of a class, a subsystem, a package, and a complete system.

It tends out to be an efficient way of modeling the interactions and collaborations in the external entities and the system. It models event-based systems to handle the state of an object. It also defines several distinct states of a component within the system. Each object/component has a specific state.

Hints

```
  /**
 * Design the state machine view with state chart diagram for an automated
company */

Identification of activities:

1.company takes loan from banks
2.buy raw materials

Develop the state machine diagram
```

12. Case study- Innovative point-of-sale solution

The case study is the Innovative point-of-sale solution (POS) system. A POS system is a computerized application used (in part) to record sales and handle payments; it is typically used in a retail store. It includes hardware components such as a computer and bar code scanner, and software to run the system. It interfaces to various service applications, such as a third-party tax calculator and inventory control. These systems must be relatively fault-tolerant; that is, even if. POS system increasingly must support multiple and varied client-side terminals and interfaces. These include a thin-client Web browser terminal, a regular personal computer with something like a Java Swing graphical user interface, touch screen input, wireless PDAs, and so forth. Furthermore, we are creating a commercial POS system that we will sell to different clients with disparate needs in terms of business rule processing. Each client will desire a unique set of logic to execute at certain predictable points in scenarios of using the system, such as when a new sale is initiated or when a new line item is added. Therefore, we will need a mechanism to provide this flexibility and customization. Using an iterative development strategy, we are going to proceed through requirements, object-oriented analysis, design, and implementation.

12.1 Design Use case model

The Use Case Model describes the proposed functionality of the new system. A Use Case represents a discrete unit of interaction between a user (human or machine) and the system. A Use Case is a single unit of meaningful work; for example, login to system, register with system and design order are all Use Cases. Each Use Case has a description which describes the functionality that will be built in the proposed system. A Use Case may 'include' another Use Case's functionality or 'extend' another Use Case with its own behavior Use Cases are typically related to 'actors'. An actor is a human or machine entity that interacts with the system to perform meaningful work.

Hints

```
  /**
 * Design the Use case view with Use case diagram for Innovative point-of-
sale solution */

A Use Case description will generally include:
General comments and notes describing the use case.
Requirements
Constraints
Scenarios
Actors

An Actor is a user of the system. This includes both human users and other
computer systems. An Actor uses a Use Case to perform some piece of work
which is of value to the business. The set of Use Cases an actor has
access to defines their overall role in the system and the scope of their
action.

Steps for process of sale
1.Customer arrives at POS checkout with goods to purchase.
2.Cashier starts a new sale.
3.Cashier enters item identifier.
```

4.System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.
5.Cashier repeats step 3-4 until done with all items.
6.System presents total with taxes calculated.
7.Cashier tells Customer the total, and asks for payment.
8.Customer pays and System handles payment.
9.System logs completed sale and sends sale and payment information to the external accounting system (for accounting and commissions) and Inventory system (to update inventory).
10.System presents receipt.
11.Customer leaves with receipt and goods.

Constraints, Requirements and Scenarios

The formal specification of a Use Case includes:

1. Requirements. These are the formal functional requirements that a Use Case must provide to the end user. They correspond to the functional specifications found in structured methodologies. A requirement is a contract that the Use Case will perform some action or provide some value to the system.

2. Constraints. These are the formal rules and limitations that a Use Case operates under, and includes pre- post- and invariant conditions. A pre-condition specifies what must have already occurred or be in place before the Use Case may start. A post-condition documents what will be true once the Use Case is complete. An invariant specifies what will be true throughout the time the Use Case operates.

3.Scenarios. Scenarios are formal descriptions of the flow of events that occurs during a Use Case instance. These are usually described in text and correspond to a textual representation of the Sequence Diagram.

Includes and Extends relationships between Use Cases

One Use Case may include the functionality of another as part of its normal processing. Generally, it is assumed that the included Use Case will be called every time the basic path is run. An example may be to list a set of customer orders to choose from before modifying a selected order - in this case the <list orders> Use Case may be included every time the <modify order> Use Case is run.

A Use Case may be included by one or more Use Cases, so it helps to reduce duplication of functionality by factoring out common behavior into Use Cases that are re-used many times.

One Use Case may extend the behavior of another - typically when exceptional circumstances are encountered. For example, if before modifying a particular type of customer order, a user must get approval from some higher authority, then the <get approval> Use Case may optionally extend the regular <modify order> Use Case.

.

12.2 Design Sequence diagram

UML provides a graphical means of depicting object interactions over time in Sequence Diagrams. These typically show a user or actor, and the objects and components they interact with in the execution of a use case. One sequence diagram typically represents a single Use Case 'scenario' or flow of events. Sequence diagrams are an excellent way to document usage scenarios and to both captures required objects early in analysis and to verify object usage later in design. Sequence diagrams show the flow of messages from one object to another, and as such correspond to the methods and events supported by a class/object.

Hints

```
  /**
 * Design the interaction view with sequence diagram for Innovative point-
of-sale solution */

The sequence diagram design with the user or actor on the left initiating
a flow of events and messages that correspond to the Use Case scenario.
The messages that pass between objects will become class operations in the
final model.
```

12.3 Design Domain Modeling

A domain model is generally implemented as an object model within a layer that uses a lower-level layer for persistence and "publishes" an API to a higher-level layer to gain access to the data and behavior of the model. In the Unified Modeling Language (UML), a class diagram is used to represent the domain model.

Hints

```
  /**
 * Design the Domain model for NextGen PoS */
Goal
The problem domain is captured in a domain model
Activities
Identify the conceptual classes with their attributes and their
associations
Input
Use Cases
Result
Conceptual class diagram
Domain objects
Associations among the objects

Attributes of the Apply the following steps:
1.Identify candidate conceptual classes
2.Add associations between the classes
3.Add attributes to the classes
Summarizes some typical situations, which leads to strategies to Identify
Conceptual Classes

Two techniques are presented in the following sections:
 1.Use a conceptual class category list.
 2.Identify noun phrases.
```

Another excellent technique for domain modeling is the use of analysis patterns, which are existing partial domain models designd by experts

Finding Conceptual Classes with Noun Phrase Identification

Another useful technique (because of its simplicity) suggested in [Abbot83] is linguistic analysis: identify the nouns and noun phrases in textual descriptions of a domain, and consider them as candidate conceptual classes or attributes.
Care must be applied with this method; a mechanical noun-to-class mapping isn't possible, and words in natural languages are ambiguous. Nevertheless, it is another source of inspiration. The fully dressed use cases are an excellent description to draw from for this analysis. For example, the current scenario of the Process Sale use case can be used.

Main Success Scenario (or Basic Flow):

Customer arrives at a POS checkout with goods and/or services to purchase.
Cashier starts a new sale.
Cashier enters item identifier.
System records sale line item and presents item description, price, and running total.

Price calculated from a set of price rules. Cashier repeats steps 2-3 until indicates done.
System presents total with taxes calculated.
Cashier tells Customer the total, and asks for payment.
Customer pays and System handles payment.
System logs the completed sale and sends sale and payment information to the external Accounting (for accounting and commissions) and Inventory systems (to update inventory).
System presents receipt.
Customer leaves with receipt and goods (if any).

Extensions (or Alternative Flows):

7a. Paying by cash:

Cashier enters the cash amount tendered.
System presents the balance due, and releases the cash drawer.
Cashier deposits cash tendered and returns balance in cash to Customer.
System records the cash payment.

The domain model is a visualization of noteworthy domain concepts and vocabulary. Where are those terms found? In the use cases. Thus, they are a rich source to mine via noun phrase identification.

12.5 Design and Develop a component diagram

A 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.

Hints

```
  /**
 * Design the Domain model for Innovative point-of-sale solution */
Common Associations list
Summarizes some typical situations, which leads to
associations
Is physical part of
Is logical part of
Is physically contained in
Is logically contained in
Focus on those associations for which knowledge of the
relationship needs to be preserved for some duration
Need
to know associations
Too many associations rather confuse than illuminate
Avoid redundant or derivable associations
```

12.6 Design and develop a 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 designd 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.

Hints

```
  /**
 * Design the Domain model for Innovative point-of-sale solution */

Deployment diagrams are used to visualize the topology of the physical
components of a system where the software components are deployed. So,
deployment diagrams are used to describe the static deployment view of a
system.
Deployment diagrams consist of nodes and their relationships.
Purpose:
```

The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related.

The purpose of deployment diagrams can be described as:

Visualize hardware topology of a system.
Describe the hardware components used to deploy software components.
Describe runtime processing nodes.

13. Case study – E-Vaccination system

Young children are at increased risk for infectious diseases because their immune systems have not yet built up the necessary defenses to fight serious infections and diseases. Making sure that children have access to proper healthcare and immunization against diseases that can be prevented by vaccines, is a huge challenge that is being faced by developing countries like ours.

This highlights the importance and need of having a better, smarter system in place, to improve the situations. This application provides a system to provide information, store records and help parents schedule vaccination appointments for their children.

Our python-based Child Vaccination Management System helps parents book vaccination appointments for their children with just a few clicks. Admin will manage the child and vaccination report and approval of the appointment. Hospitals will update the status of the vaccination applied for the child

13.1 Software requirement specification document

A Software Requirement Specification (SRS) is basically an organization's understanding (in writing) of a customer or potential client's system requirements and dependencies at a particular point in time prior to any actual design or development work. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time.
It's important to note that an SRS document contains functional and nonfunctional requirements only; it doesn't offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understands the customer's system requirements to be.
SRSs are typically developed during the first stages of "Requirements Development," which is the initial product development phase in which information is gathered about what requirements are needed--and not. This information-gathering stage can include onsite visits, questionnaires, surveys, interviews, and perhaps a return-on-investment (ROI) analysis or needs analysis of the customer or client's current business environment. The actual specification, then, is written after the requirements have been gathered and analyzed.

Hints

```
/* SRS Document Outline */
1. Introduction
Purpose
Document conventions
Intended audience
Additional information
Contact information/SRS team members
References
2. Overall Description
Product perspective
Product functions
User classes and characteristics
Operating environment
User environment
Design/implementation constraints
Assumptions and dependencies
3.External Interface Requirements
User interfaces
Hardware interfaces
Software interfaces
Communication protocols and interfaces
4. System Features
System feature
Description and priority
Action/result
Functional requirements
System feature B
5. Other Nonfunctional Requirements
Performance requirements
Safety requirements
Security requirements
Software quality attributes
Project documentation
User documentation
6. Other Requirements
Appendix A: Terminology/Glossary/Definitions list
Appendix B: To be determined
```

13.2 Design an Use case View

Use case view described by a use case diagram. A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

Hints

```
/* Design Usecase view with usecase diagram */

The actors in use case diagram are Applicant, regional administrator, database,
passport Administrator, Police.
• The use cases are Login, givedetails, logout, collectdetails, verification, issue.
• The actors use the use case are denoted by the arrow
• The login use case checks the username and password for applicant, regional
administrator, passport administrator and police.
• The submit details use case is used by the applicant for submitting his details
• The check status use case is used by the applicant for checking the status of the
application process.
• The get details, verify and store verification use case is used by passport
administrator, regional administrator, and police.
• The details use case is used for getting the details form the database for verification
The verify use case is used for verifying the details by comparing the data in the
database.
• The store verification use case is to update the data in the database
• And finally, the issue passport use case is used by the passport administrator for
issuing passport whose application verified successfully by all the actor
```

13.3 Design a class diagram

A class is drawn as rectangle box with three compartments or components separated by horizontal lines. The top compartment holds the class name and middle compartment holds the attribute and bottom compartment holds list of operations.

Hints

```
/* Design static design view with class diagram */

Class diagram contains classes and their relationships.

The classes are

APPLICANT-The applicant has attribute such as name and password and operations
are login, give details and logout. The applicant login and fill the details that are
required for applying the passport. After applying the person can view the status of
the passport verification process.

THE DATABASE-The database has attributed such as name and operation are store.
```

```
The purpose is to store the data.

REGIONAL ADMINISTRATOR- The regional administrator has attribute such as
name and operation are getting details, verify details and send. The
regional
administrator get the details form database and verify with their
database.

PASSPORT ADMINISTRATOR-The passport administrator has attributed such as
name and operation are getting details, verify details and issue. The
passport
administrator get the details form database and verify with their
database, update the verification and issue the passport.

THE POLICE-The police have attribute such as name and operation are
getting details,
verify details and send. The police get the details form database and
verify with their database, update the verification in the database.

The relationships are Association, dependency, generalization
```

13.4 Design an Interaction view

This interaction view is represented in UML by two diagrams known as Sequence diagram and Collaboration diagram. The basic purpose of both the diagrams are similar. Sequence diagram emphasizes on time sequence of messages and collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.

Hints

```
/* Design interaction view with sequence diagram */

Sequence diagram contains objects and their interactions.
A sequence diagram shows an interaction arranged in time sequence,

It shows object participating in interaction by their lifeline by the
message they exchange arranged in time sequence. Vertical dimension
represents time and horizontal dimension represent object.

•The applicant login the database and give his details and database store
the details.

•The passport administrator gets the details from the database and do
verification and the forward to regional administrator.

•The regional administrator gets details form passport administrator and
perform verification and send report to passport administrator.

•The police get the details form passport administrator and perform
verification and send report to passport administrator

A collaboration diagram is similar to sequence diagram but the message in
```

```
number format. In a collaboration diagram sequence diagram is indicated by
the
numbering the message.

A collaboration diagram, also called a communication diagram or
interaction diagram, A sophisticated modeling tool can easily convert a
collaboration diagram into a sequence diagram and the vice versa.

A collaboration diagram resembles a flowchart that portrays the roles,
functionality and behavior of individual objects as well as the overall
operation of the system in real time
```

13.5 Design an Implementation View

The component diagram is represented by figure dependency and it is a graph of design of figure dependency. The component diagram's main purpose is to show the structural relationships between the components of a systems. It is represented by boxed figure. Dependencies are represented by communication association.

Hints

```
/* Design implementation view with component diagram */

component diagram contains components and relationships.

The components in the passport automation system are passport automation
system,
applicant, passport administrator, regional administrator, and police.
```

14. Software Testing

A unit test is a type of software test that focuses on components of a software product. The purpose is to ensure that each unit of software code works as expected. A unit can be a function, method, module, object, or other entity in an application's source code. The objective of a unit test is to test an entity in the code, ensure that it is coded correctly with no errors, and that it returns the expected outputs for all relevant inputs.

Performance testing tests the non-functional requirements of the system. The different types of performance testing are loading testing, stress testing, endurance testing and spike testing.

14.1 Unit test for a function to add two numbers

Unit testing is testing the smallest testable unit of an application. It is done during the coding phase by the developers. To perform unit testing, a developer writes a piece of code (unit tests) to verify the code to be tested (unit) is correct.

Hints

```
  /** Unit Testing **/
Sample code

int Add (int a, int b)
 {
return a+b;
}

Test case:
void TestAdd1()
{
Assert.IsEqual(Add (5, 10), 15)
}
The above unit test “asserts” that 5 + 10 is equal to 15. If the Add function returns anything else Assert.IsEqual result in error and the test case will fail.

probably add a few more unit test cases like these:
void TestAdd2() {Assert.IsEqual(Add (500, 1000), 1500)}
void TestAdd3() {Assert.IsEqual(Add (0, 1000), 1000)}
void TestAdd4() {Assert.IsEqual(Add (-100, 100), 0)}
void TestAdd5() {Assert.IsEqual(Add (-100, -1100), -1200) }

After write test cases, will run them to verify that everything is working correctly.
```

14.1 Performance testing-scenario1

Following are the types of performance testing.

- Load testing
- Stress testing
- Scalability testing
- Stability testing

In the below image, 1000 users are the desired load, which is given by the customer, and 3/second is the goal which we want to achieve while performing a load testing.

Hints

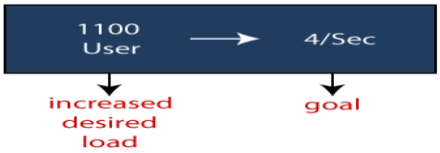
```
  /** Performance Testing **/

Identify the suitable performance testing
```

```
The load testing is used to check the performance of an application by
applying some load which is either less than or equal to the desired load
is known as load testing.
```

14.2 Performance testing-scenario2

If we took the below example and increased the desired load 1000 to 1100 users, and the goal is 4/second. While performing the stress testing in this scenario, it will pass because the load is greater (100 up) than the actual desired load.



Hints

```
  /** Performance Testing **/

Identify the suitable performance testing

The stress testing is testing, which checks the behavior of an application
by applying load greater than the desired load.
```

14.3 Performance testing-scenario3

When we have the 1000 users as desired load, and the 2.7/sec is goal time, these scenarios will pass while performing the load test because in load testing, we will concentrate on the no. of users, and as per the requirement it is equal to 1000 user.

Hints

```
  /** Performance Testing **/

Identify the suitable performance testing
```

14.4 Performance testing-scenario4

Increase the desired load by 100 users, and goal time will go up to 3.5\sec. This scenario will pass if we perform stress testing because here, the actual load is greater than (1100) the desired load (1000).

Hints

```
  /** Performance Testing **/

Identify the suitable performance testing
```

15. Final Notes

Unified Modeling Language which is abbreviated as UML refers to a general purpose and standardized modeling language which is primarily used in the object-oriented software

engineering field. This specific modeling language features a collection of graphical notation techniques that are extremely useful in developing visual models of software-intensive systems that are also known for being object-oriented. One of the most widely recognized versions of UML is UML 2 which is also popular for its 3 major diagram classifications. The first one is the behavior diagram which is capable of depicting a business process or system's behavioral features. The second one is called the interaction diagram which is known to be a subset of the behavior diagram and works by emphasizing interactions in objects including interaction overview, timing diagrams, sequence and communication. The last diagram classification is called the structure diagram which is capable of depicting the major elements of specifications known for being irrespective of time including composite structure, deployment package diagrams, object, component and class. UML 2 is also composed of features that make it even more functional to users.

If you plan to become a certified UML 2 expert, an online certification provider which has gained worldwide recognition for its excellent reputation, now some organizations offer a free UML 2 practice course. It also features a free study guide and a well-developed free practice test that are necessary in your attempt to confidently prepare yourself towards taking the certification exam.

- OMG-Certified UML Professional 2 (OCUP 2) exams test an individual's ability to properly interpret and construct UML model diagrams in the way UML is used today. (https://www.omg.org/ocup-2/)
- Free UML 2 Certification Test – (https://www.brainmeasures.com/courses/online/899/free-uml-2-certification-test.aspx)
- O'Reilly UML2 certification https://www.oreilly.com/library/view/uml-2-certification

Student must have any one of the following certifications:

- OMG Certifications– UML 2 certification
- OMG Certified UML® Professional (OCUP 2™)
- Brain measures UML 2 Practice Certification
- O'Reilly learning platform- UML 2 Certification
- NPTEL – Object Oriented Analysis and design
- NPTEL - Object Oriented System Development using UML, Java and Patterns

V. TEXT BOOKS:

1. Grady Booch, James Rumbaugh, Ivar Jacobson, *The Unified Modeling Language User Guide*, Pearson Education, 2nd edition, 2004.
2. Craig Larman, *Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Iterative Development*, Pearson Education, 3rd edition, 2005.

VI. REFERENCE BOOKS:

1. Karoly Nyisztor and Monika Nyisztor *UML and Object-Oriented Design Foundations: Understanding Object-Oriented Programming and the Unified Modeling Language*, Pearson, 7th edition, 2018.
2. Matt Weisfeld *The Object-Oriented Thought Process*, 4th edition, 2013.
3. Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, O'Reilly Media ,3rd edition, 2004.
4. Mike O'Docherty. *Object-Oriented Analysis and Design - Understanding System Development with UML 2.0*, 2nd edition, 2005

VII. ELECTRONICS RESOURCES:

1. https://staruml.sourceforge.net/docs/user-guide(en)/ch04.html
2. https://www.tutorialspoint.com/uml/uml_standard_diagrams.html
3. https://www.uml-diagrams.org/
4. https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/

VIII. MATERIALS ONLINE:

1. Syllabus
2. Lab manual