SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chennai-603203.

FACULTY OF ENGINEERING AND TECHNOLOGY

School of Computing

Department of Data Science and Business Systems

Academic Year (2022–2023)

18CSC268J - Software Design with UML

SEMESTER–IV

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chennai-603203.

FACULTY OF ENGINEERING AND TECHNOLOGY

BONAFIDE CERTIFICATE

Certified that this is the Bonafide record of work done by **Katha Sai Indra Reddy** **RA2111042010021** of IV semester B.Tech COMPUTER SCIENCE WITH BUSSINESS SYSTEMS during the academic year 2022-2023 in the 18CSC268J - Software Design with UML.

S.JEEVA
Staff -Incharge

Dr. M. Lakshmi
Head of The Department

Submitted for the practical examination held on _____at SRM Institute of Science and Technology, Kattankulathur, Chennai-603203

Examiner-1

Examiner-2

INDEX

SESSION	DATE	EXERCISE	SIGN
1	19-01-23	5 OPEN-SOURCE TOOLS FOR UML	
2	25-01-23	UML PACKAGE DIAGRAM	
3	03-02-23	UML PACKAGE DIAGRAM WITH DRILL DOWN FEATURES	
4	10-02-23	CASE STUDY ON STATE-CHART AND ACTIVITY DIAGRAM	
5	22-02-23	UML STATE-CHART DIAGRAM	
6	22-02-23	UML ACTIVITY DIAGRAM	
7	09-03-23	UML USE CASE AND CLASS DIAGRAM	
8	16-03-23	UML COMPONENT DIAGRAM	
9	24-03-23	UML INTERACTION DIAGRAM	
10	10-0423	UML COLLABORATION DIAGRAM	
11	18-04-23	UML DEPLOYMENT DIAGRAM	
12	25-04-23	UML THREADS	

EXPERIMENT NO-1

5 open-source tools for drawing UML diagrams

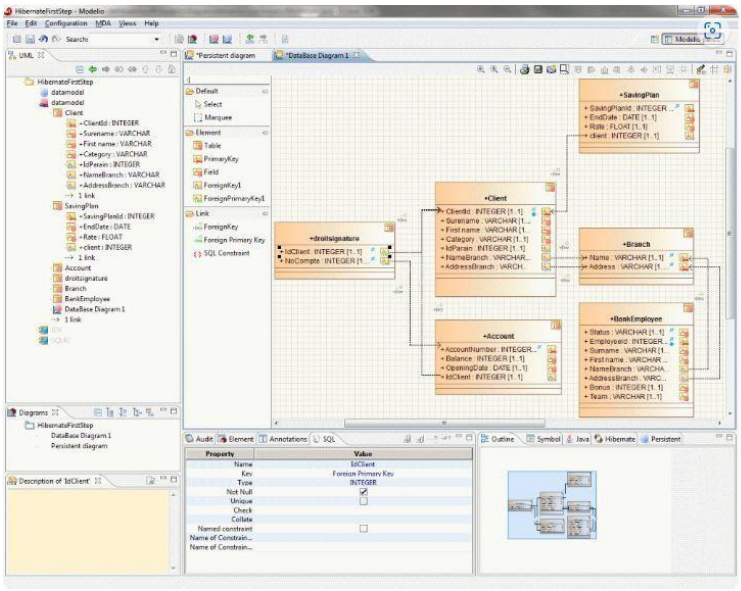
AIM: Understanding and studying about various open source tools to draw UML diagrams.

MODELIO

- Modelio is the first modelling environment.
- The tool combines BPMN support and UML support.
- It is one of the best free UML tools that provide support for a wide range of models and diagrams.
- It is accessible to the user as an open-source UML diagram tool for developing UML diagrams.

Features:

- Modelio offers an XMI import/export feature that enables you to exchange UML2 models between various tools.
- It is one of the best open-source UML tool that offers you to export diagrams in SVG, Gif or JPEG format.
- You can extend Modelio for any language, methodology, or modeling technique.
- It is a free and open-source HTML5 online flowchart software
- It provides multiple diagram options cloud storage
- Supports PNG, JPEG, WMF, XML, GIF and BMP file formats
- It Provides import/export options for XML, XMI, EMF, PNG, JPEG and BMP
- Support programming language like C++, C#, Java and SQL
- Offers Multi-file support, Drag and Drop, and Print option



Limitations

- BPMN documentation is not exported nor imported
- Builds fail because of missing directories in build properties
- Unable to look for installed plugins
- Observation of translucent defects

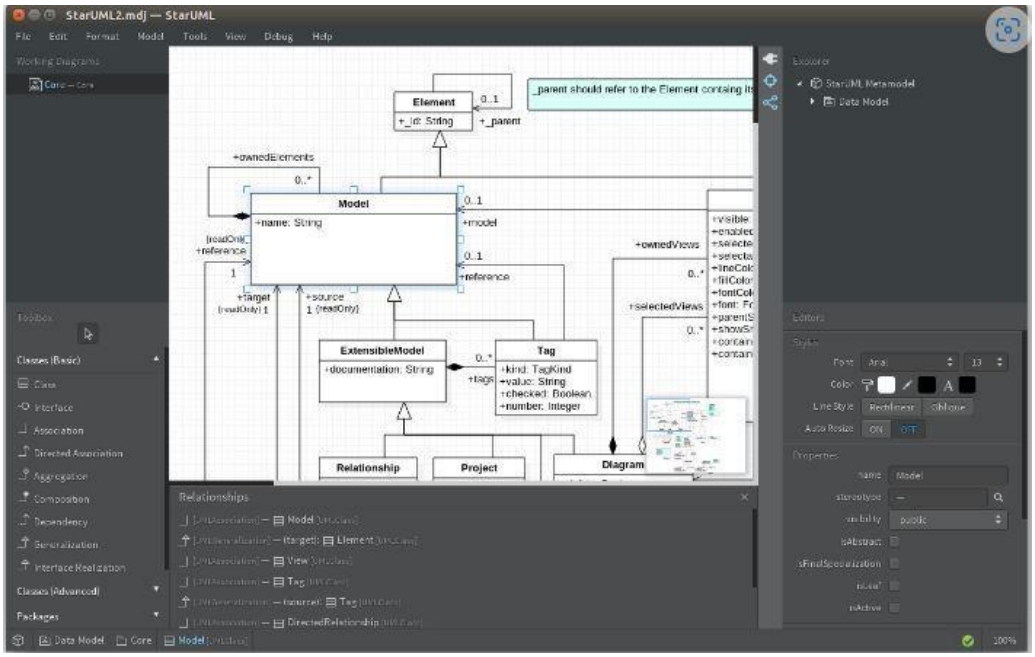
STAR UML

- StarUML is a software modeler for agile and concepts modeling for macOS windows and different types of Linux(Ubuntu, Debian, Redhat, fedora).
- It is one of the best UML software that provides eleven types of diagram.
- it is support code generation for various programming languages such as Java, C#, C++, and python, it also has Open APIs

Features:

- This UML diagram software allows you to discover and install third-party extensions.
- No limit to using this commercial software for evaluation.
- Offers pre-built templates for Class Diagram
- It provides multiple diagram options cloud storage

- Supports JSON, PNG, HTML and JPEG file formats
- It Provides import/export options for PDF, XML, PNG, JPEG, HTML and SVG
- Support programming language like C, C++, C#, Java and Python
- Provides Custom Uml Profile, Auto Update, Model-driven Development, Fast Modeling, Dark And Light Themes and Code Markdown Support
- Offers Multi-file support, Drag and Drop, and Print option



Limitations

- Ascending Degree of Complexity
- Architecture – Indifferent Design
- Support for new languages in case of reverse engineering
- Classes with template parameters do not export class as a template class

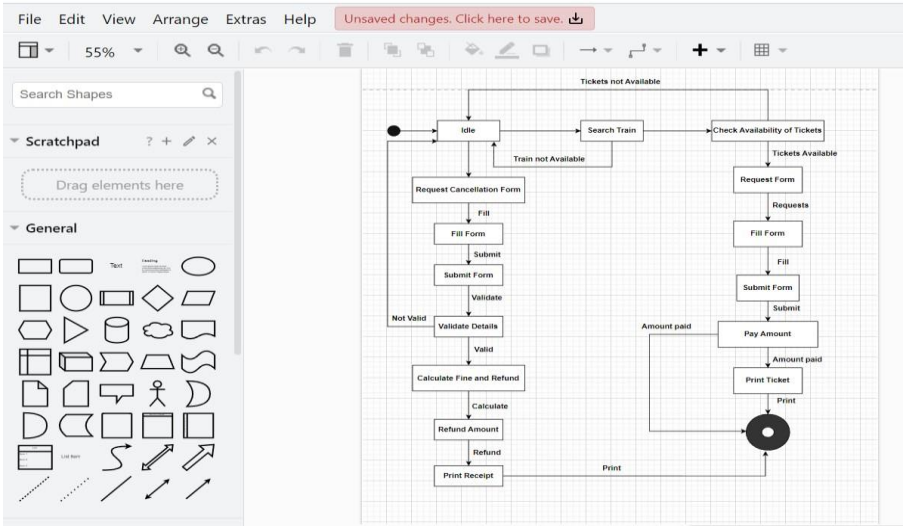
DRAW.IO

- Draw.IO is a free online UML tool.

- It is one of the best UML tools that allows users to create and manage the drawing easily these tools.
- A lot of the wide and early share is available with this tool.

Features:

- No limit on the number of sizes
- Templates are present in the software design itself.
- This free UML diagram tool allows you to save the model in your preferred location
- Offers pre-built templates for diagram
- Seamlessly integrates with Aha, Atom, Bioiocons, BookStack, Docstell, FOSWiki,
    - Grafana, Growi, JupyterLab, Lark, LumApps, Nextcloud, Nuclino, Redmine and Sandstorm
- It provides multiple diagram options cloud storage
- Supports PNG, JPEG, SVG, JSON, XML, CSV and PDF file formats
- It Provides import/export options for PNG, JPEG, SVG, JSON, XML, CSV and PDF
- Offers firewall based encryption
- Provides Keep your diagram data secure, Diagram wherever you want, Collaborate in real-time with shared cursors, Easy-to-use diagram editor, and Many advanced tools
- Offers Version history, Drag and Drop, and Print option



Limitations

- As you work in a browser, the app seems to lag if you work on it for a while. ▪ The app appears to glitch from time to time if you work for longer periods of time,
- Making you lose some of your work of you're not careful.

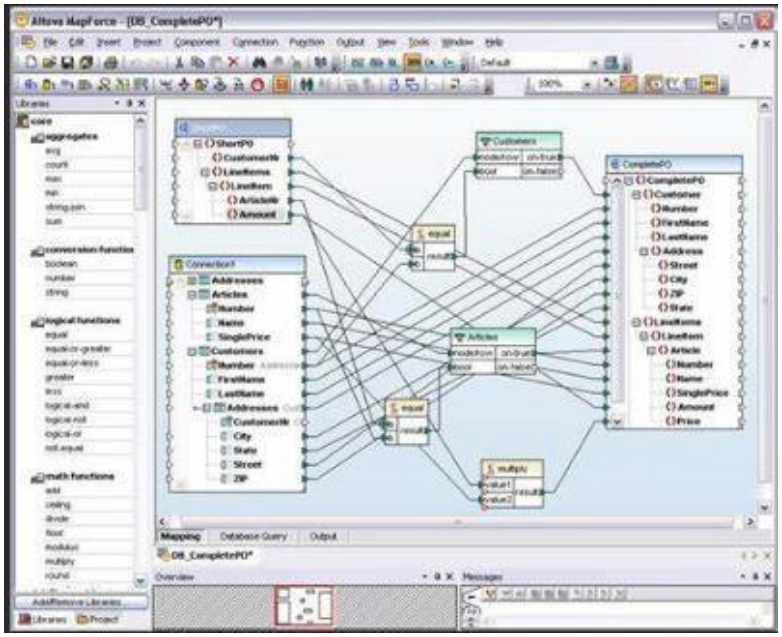
- Exporting your work in the form you would like can be challenging.

VISUAL PARADIGM

- Visual Paradigm is a software design tool that is tailored for engine software projects.
- This UML editor tool helps the software development team to model business information systems and development processes.

Features:

- It offers a complete tool like for process analysis, system design, database design, etc.
- Offers user story feature to capture and maintain users' needs.
- Offers pre-built templates for Social Media, Events, Business, Marketing, Documents,
    - School, Personal, Infographics, Posters and Gift Card
- Seamlessly integrates with Eclipse, NetBeans, IntelliJ IDEA, Visual Studio and Android Studio
- Free plan offers 1GB cloud storage
- It provides multiple diagram options 1GB cloud storage
- Supports JPG, PNG, SVG, TIFF, EMF, PDF and DOC file formats
- It Provides import/export options for JPG, PNG, SVG, TIFF, EMF, PDF and XML
- Offers 256-bit SSL encryption
- Provides Visual Modeling, Enterprise Architecture, Business Analysis & Design,
    - Project Management, Agile & Scrum Development, Online Diagram Tool, Spreadsheet Tool, and Team Collaboration
- Offers Multi-file support, Drag and Drop, and Print option



Limitations:

- When you move projects from Visual Paradigm into Sparx Systems Enterprise Architect via XMI files, you lose all your notes from the elements, so this is an area for improvement in Visual Paradigm.
- TOGAF support feature is not available.
- ArchiMate support should be added.

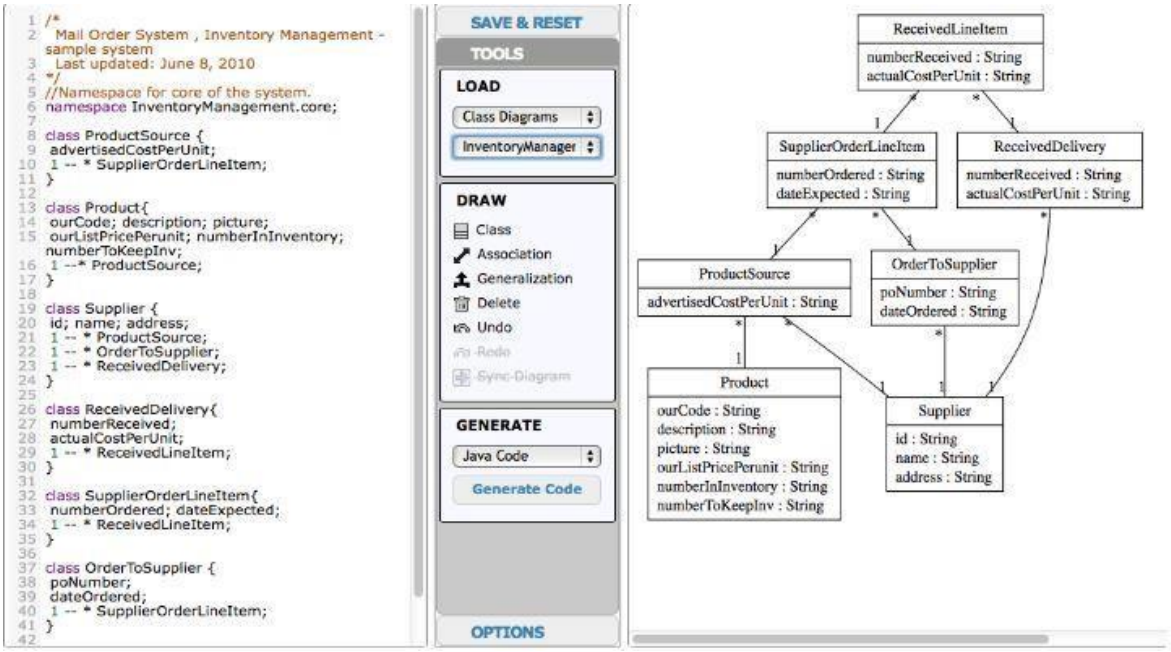
UMPLE

- Umple is an open-source modelling tool for software developers and students to make UML in the fastest way in their classroom.
- Its works online as an eclipse plugin and as a stand-alone command-line Jar.
- Umple is a model-oriented programming technology that adds UML associations and state machines to Java and PHP.
- It is used to draw UML diagrams, embeds models in code, and generate a complete system.

Features:

- Simple and saves time
- It makes you model in UML textually.
- You can generate top quality code from class diagrams and state diagrams
- You can add a little bit of Umple code into an existing Java, PHP, or Ruby system ➢

  Umple works like a pre-processor



Limitations:

- Making it possible to develop software simultaneously with a textual and visual ▪ representation of high-level abstractions.
- Reducing the volume of code that needs to be written due to code from abstractions.
- Umple is explicitly not a single new programming language instead, it consists of a set of
- features that extend multiple existing programming languages.

Why Star UML is preferred over the above other four tools?

Star UML is faster, flexible, and can be extended to accommodate other codes in the diagram. The extensive features make the users fall in love. If mistakes happen, we can undo and make adjustments. This feature is not applicable in some UML tools.

Anyone can understand the framework, and hence the architecture can be modified for the extensive use of the software. Performance and security can be tracked easily with the help of Star UML. Documentation is provided with proper guidelines to improve the business processes. Star UML is the visual language that communicates information to the users in a diagrammatic manner.

The tools in Star UML help to know the requirements in the system and apply the design patterns so that proper analysis can be done to understand and modify the diagrams. These tools are open source and for more high requirements, tools can be purchased from the software vendors.

Result:

Hence the study of different open-source to draw UML diagrams has been studied successfully.

EXPERIMENT NO-2

UML PACKAGE DIAGRAM

AIM: To draw a UML package diagram for Airline Reservation System using Star UML.

DESCRIPTION:

A package diagram is a type of Unified Modelling Language (UML) diagram that shows the organization and structure of related components in a software system. The purpose of a package diagram is to provide a high-level view of the system's components and how they are related to one another.

In a package diagram, the components of the system are grouped into packages, which are represented as rectangles with a tab. The name of the package is written inside the rectangle, and the packages can be nested within one another to indicate their relationships. Packages can contain classes, interfaces, and other packages, and the relationships between packages can be shown using directed arrows.

A package diagram can also show dependencies between packages, which indicate that one package relies on another to function properly. These dependencies are represented by directed arrows pointing from the dependent package to the package it depends on.

PURPOSE:

Package diagrams are useful for modeling the structure and organization of a system, and they provide a high-level view that can be easily understood by stakeholders and other team members. They are also useful for identifying potential problems in the design early on and making changes before they become more complex and costly to resolve.

HOW TO DRAW:

Identify the system's components: Analyze the requirements of the system and identify the major components or functionalities that need to be modeled.

Determine the packages: Group related components into packages, where each package represents a functional unit of the system.

Create class diagrams for each package: Create class diagrams for each package, including classes, their attributes, and relationships between classes.

Represent packages and dependencies: Use UML notation to represent the packages and their dependencies on one another. The package symbol is a rectangle with a tab, and dependencies between packages are represented by directed arrows pointing from the dependent package to the package it depends on.

Add optional components: If necessary, add additional components, such as interfaces, to the packages to further refine the design.

Review and refine the diagram: Review the diagram to ensure that it accurately represents the system and its components, and make any necessary revisions.

Document the diagram: Document the diagram with meaningful names and descriptions to help others understand the design

Package diagram for airline reservation system

A package diagram for a airline reservation service might include the following packages and classes:

Order Package:

- Order class: to manage passenger bookings, including details such as plans taken, coupons applied, payment information, etc.

Reservation Package:

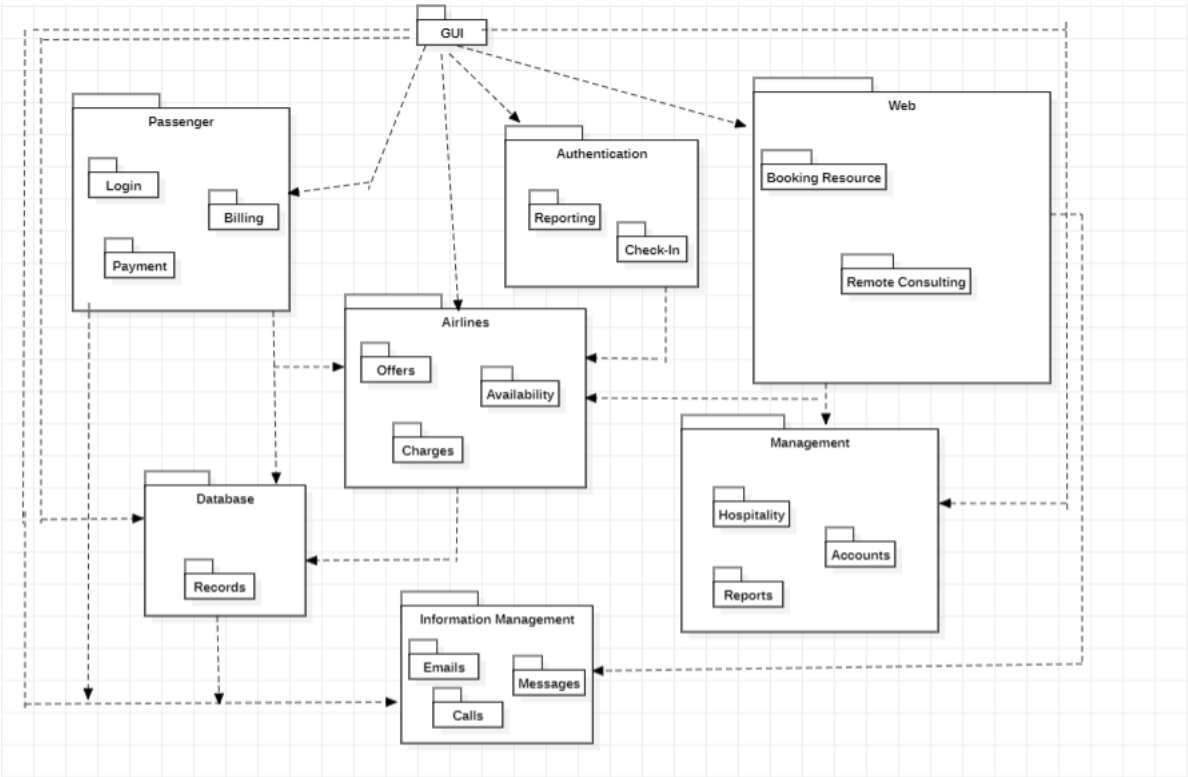
- Reservation class: to manage the reservation process, including details such as reservation status, registration ID, etc.

Payment Package:

- Payment class: to manage payment processing, including details such as payment method, payment status, etc.
- Credit Card class: to store information about a customer's credit card, including name, number, expiration date, etc.

Customer Package:

- Customer class: to store information about customers, including name, contact information, order history, etc.
- These packages and classes can be connected with various relationships, such as inheritance, aggregation, and association, to model the relationships and interactions between different parts of the system.



PACKAGE DIAGRAM FOR AIRLINE RESERVATION SYSTEM

RESULT:

Thus, we have successfully drawn the UML package diagram for AIRLINE RESEVATION SYSTEM.

EXPERIMENT NO-3

Package diagram with drill down feature

AIM: To draw a UML package diagram with drill down features for Airline Reservation system using Star UML.

DESCRIPTION:

In UML, drill down feature allows the user to navigate from a higher-level view of a system to a lower-level view of the system's components. It provides a way to explore the details of a system by breaking down a complex diagram into smaller and more detailed diagrams. This is particularly useful when dealing with complex systems or large diagrams, as it allows users to focus on specific parts of the diagram while ignoring the rest. By drilling down into a diagram, users can gain a deeper understanding of the system and its components.

Purpose:

Drilldown is a technique used to navigate and explore complex information or data in a hierarchical structure. It involves starting with a high-level view of the data and then progressively "drilling down" to view more detailed information. This technique is often used in data analysis, software development, and business intelligence to analyse and understand complex systems.

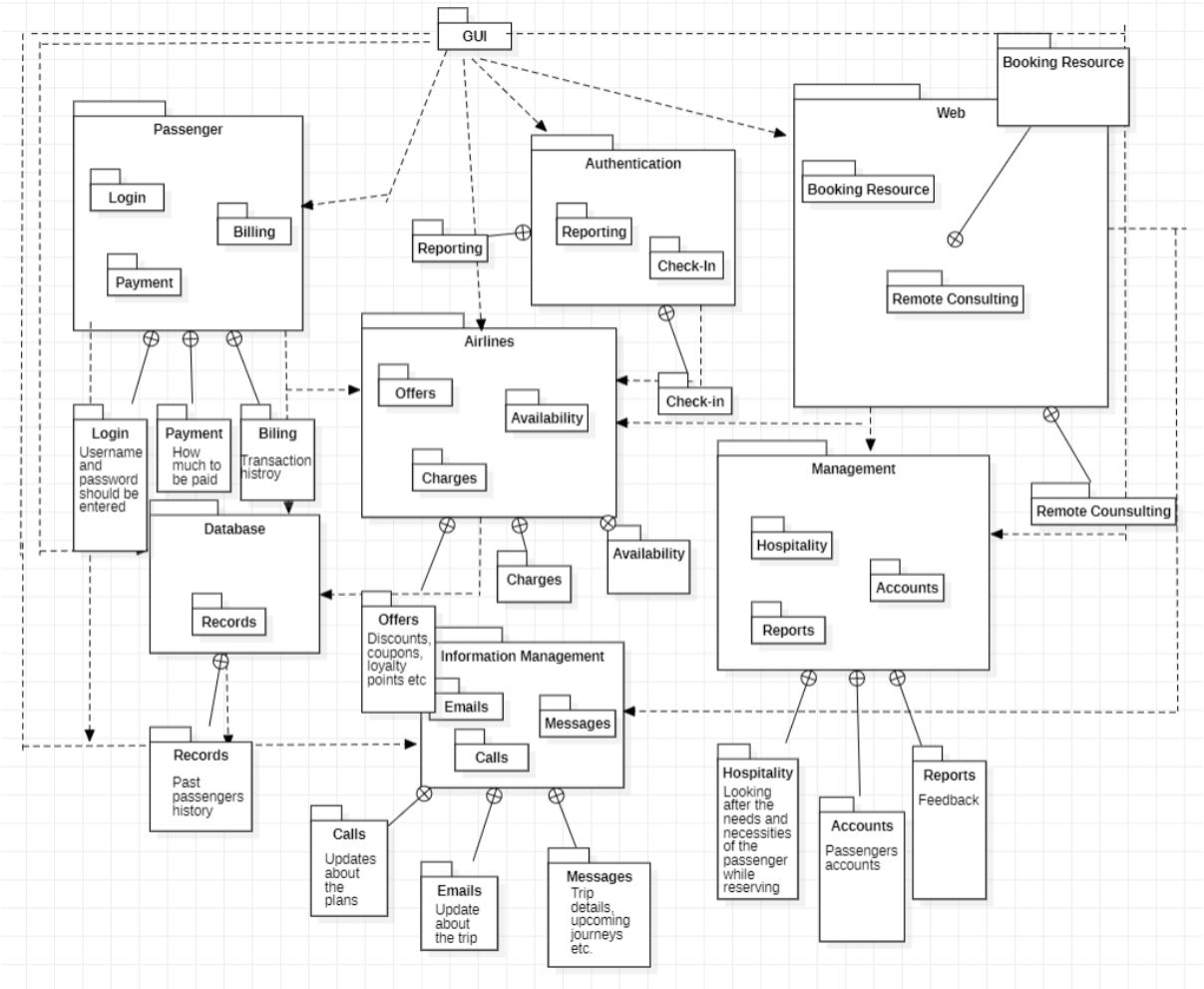
Drill down is an analytics capability that allows users to instantly shift from an overview of data to a more detailed and granular view within the same dataset they are analysing by clicking on a metric in a dashboard or report. It enables everyone to explore specific information in a report from different angles by stepping down from one level of a predefined data hierarchy to the next.

For example, a tabular report on sales performance results by region can help a user better understand why one region outperformed another by drilling down into a specific region. By clicking on a specific region, the report is automatically refreshed with further detail updated – in this case, to display all countries within that region and their various sales performances. This allows the user to go deeper – or ‘drill down’ into more particular layers of data they’re analysing.

HOW TO DRAW:

- Start by drawing the main package that you want to represent in your diagram.
- Identify the sub-packages or modules that make up the main package.
- Draw each sub-package as a smaller package inside the main package, using a nesting or containment relationship between them.
- Add any relevant classes or interfaces to the sub-packages, and draw these inside the sub-package boxes.

- Identify any drill down features that you want to include in your diagram. These could be methods, attributes, or other elements of the classes or interfaces.
- Draw these drill down features as additional boxes inside the sub-packages or classes, with a dependency or association relationship to the main package or class.
- Use appropriate labels and annotations to make the diagram clear and easy to understand.
- Repeat the process for any additional levels of drill down that you want to include in your diagram, nesting sub-packages or classes inside each other as needed.



RESULT:

Thus, we have successfully drawn the UML package diagram with drill down features for Airline Reservation System.

EXPERIMENT NO-4

STATE AND ACTIVITY DIAGRAM

AIM: To study about state and activity diagrams.

An airlines reservation system involves a complex set of interactions between the user, the airline company, and various external systems. Both activity diagrams and state chart diagrams can be used to model the behavior of such a system, but they focus on different aspects of the system's behavior.

Activity Diagram :-

An activity diagram for an airline reservation system could model the flow of activities or actions involved in making a reservation, such as selecting the departure and arrival airports, choosing a date and time, selecting a seat class, and paying for the reservation. The diagram would include symbols for actions, decisions, and other elements, and would show the sequence of actions that the user takes to make the reservation. The diagram could also show alternative paths, such as if a flight is not available on the selected date, or if the user decides to change the reservation after it has been made.

State Chart :-

A state chart diagram for an airline reservation system would focus on the various states that the system can be in, and the transitions between those states. The diagram would show the different states that the system can be in, such as "Idle", "Booking", "Ticketing", and "Cancelled". It would also show the events or conditions that trigger state transitions, such as a user selecting a flight, or a payment being processed. The diagram would also show the actions or behaviors that occur when the system enters each state, such as updating the flight availability or sending a confirmation email.

Summary of both Diagrams:-

In summary, both activity diagrams and state chart diagrams can be used to model an airline reservation system, but they focus on different aspects of the system's behavior. An activity diagram focuses on the sequence of actions that the user takes to make a reservation, while a state chart diagram focuses on the states and transitions of the system as a whole. Both types of diagrams are useful for understanding and designing complex systems.

RESULT:

Case study on state chart and Activity diagram of AIRLINR RESEVATION SYSTEM was successfully done

EXPERIMENT NO-5

UML STATECHART DIAGRAM

AIM:

To draw a UML state chart diagram for Airline Reservation System using Star UML.

DESCRIPTION:

A state chart diagram is a type of UML (Unified Modeling Language) diagram used to represent the behavior of a system or object. It shows the various states of an object or system and the transitions between these states in response to events or conditions.

Purpose:

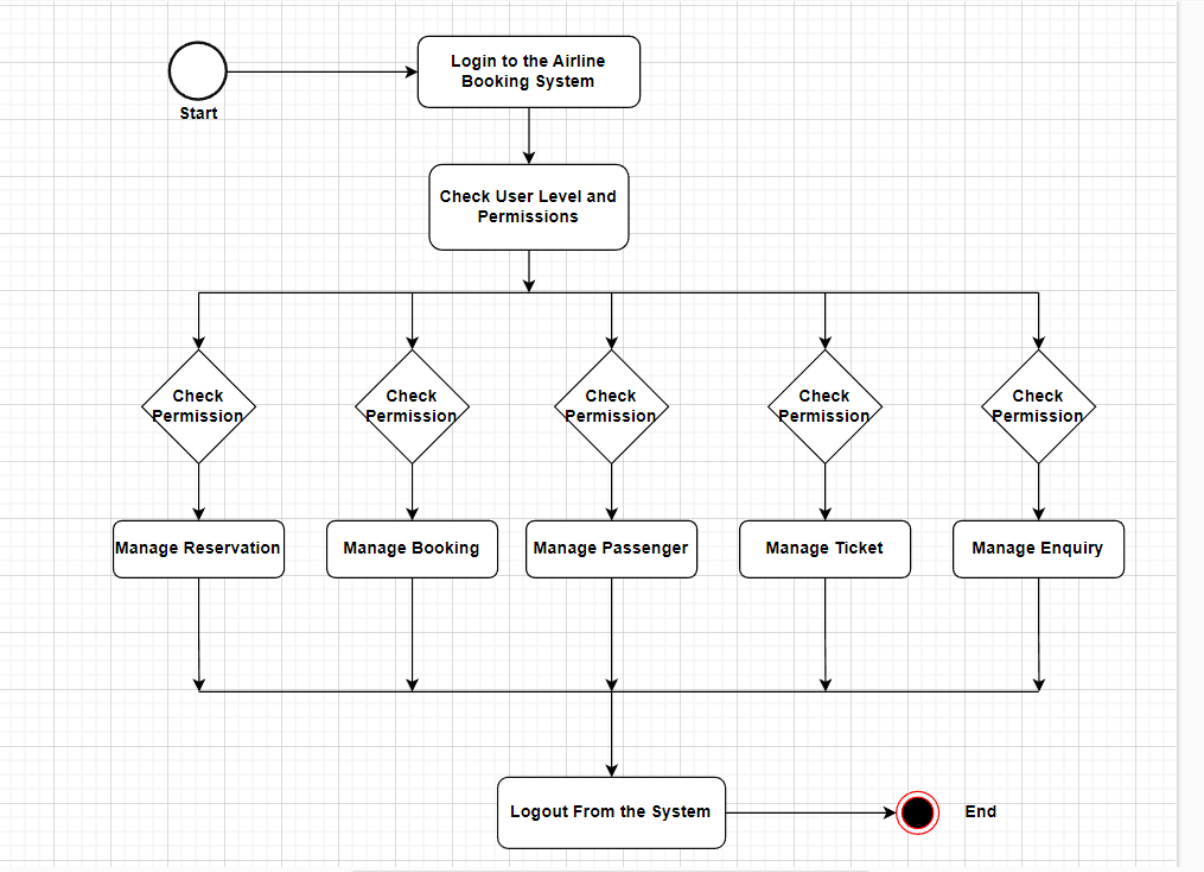
This is the Login Activity Diagram of Airline Booking System, which shows the flows of Login Activity, where admin will be able to login using their username and password. After login, users can manage all the operations on Airlines Booking, Booking Enquiry, Airline Enquiry, Passenger, Ticket Booking. All the pages such as Airline Enquiry, Passenger, Ticket Booking are secure and users can access these pages after login. The diagram below helps demonstrate how the login page works in an Airline Booking System. The various objects in the Passenger, Airlines Booking, Booking Enquiry, Airline Enquiry, and Ticket Booking page-interact over the course of the Activity, and users will not be able to access this page without verifying their identity.

HOW TO DRAW:

- Identify the object or entity: The first step is to identify the object or entity that you want to model with the state chart diagram. This could be a system, a device, or an entity such as a user or a customer.
- Identify the states: Once you have identified the object or entity, the next step is to identify the different states that it can be in. States represent different conditions or modes that the object or entity can be in, such as "idle," "processing," or "complete."
- Identify the events: The next step is to identify the events that cause the object or entity to transition from one state to another. Events represent external stimuli or inputs that cause the object or entity to change its state, such as a user clicking a button or a system receiving a message.
- Define the transitions: Once you have identified the states and events, the next step is to define the transitions between the states. Transitions represent the movement from one state to another, and are triggered by events. Transitions can have conditions or actions associated with them, which describe what happens when the transition occurs.
- Draw the diagram: The final step is to draw the state chart diagram, using UML notation to represent the states, events, and transitions. The state chart diagram should be easy to read and understand, and should accurately represent the behavior of the object or entity being modeled.

The main components of a state chart diagram are:

- States: These are the different conditions or situations that an object or system can be in.
- Each state is represented by a rectangle with a name.
- Transitions: These are the movements or changes from one state to another in response to an event or condition. They are represented by arrows with labels that indicate the trigger for the transition.
- Events: These are the occurrences or stimuli that cause a transition from one state to another. They are represented by small circles along the arrows.
- Actions: These are the operations or activities that occur when a transition takes place. They are represented by labels on the arrows or on the state itself.
- Guards: These are the conditions that must be true for a transition to occur. They are represented by labels on the arrows or on the state itself.
- Start and End Points: These indicate the beginning and end of the state chart diagram. The start point is represented by a filled circle and the end point is represented by a filled circle with a ring around it.



RESULT:

Thus, we have successfully drawn the UML state chart diagram for Airline Reservation System.

EXPERIMENT NO-6

UML ACTIVITY DIAGRAM

AIM:

To draw a UML activity diagram for Airline Reservation System using Star UML.

DESCRIPTION:

An activity diagram is a type of UML (Unified Modeling Language) diagram used to model the flow of activities or actions within a system or process. It represents the steps or activities that are required to achieve a particular goal, and shows the flow of control between them.

Purpose:

A state diagram, sometimes known as a state machine diagram, is a type of behavioural diagram in the Unified Modelling Language (UML) that shows transitions between various objects. Here, each state diagram typically begins with a dark circle that indicates the initial state and ends with a bordered circle that denotes the final state. However, despite having clear start and end points, state diagrams are not necessarily the best tool for capturing an overall progression of events. Rather, they illustrate specific kinds of behaviour in particular, shifts from one state to another.

State diagrams of airline reservation systems mainly depict states and transitions. States are represented with rectangles with rounded corners that are labelled with the name of the state. Transitions are marked with arrows that flow from one state to another, showing how the states change.

The detailed state chart diagram of airline reservation system has been given in the UML Diagram below.

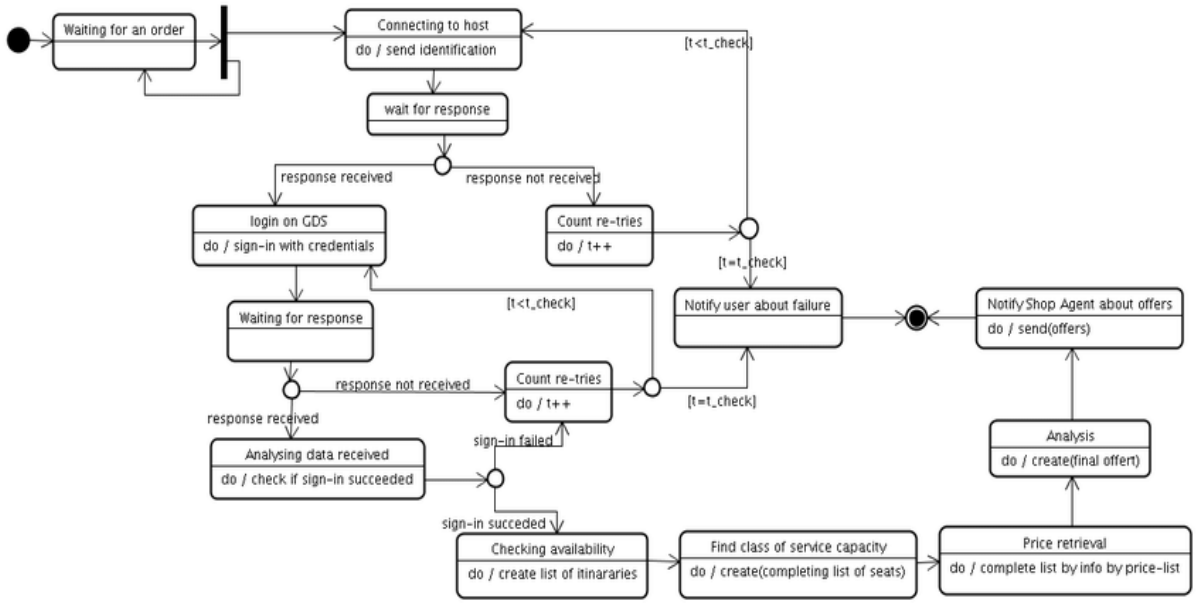
HOW TO DRAW:

- Identify the purpose and scope of the activity diagram. Determine what process or workflow you want to model and what the specific goals of the diagram are.
- Identify the actors and objects involved in the process or workflow. Actors are external entities that interact with the system, while objects are internal entities that perform activities.
- Create a start node and an end node for the diagram. The start node represents the beginning of the process or workflow, while the end node represents the final outcome or goal.
- Identify the different activities or actions that need to be performed in order to achieve the goal of the process. These can be represented as nodes in the diagram.
- Connect the nodes with arrows to show the flow of activities. Use decision nodes to represent conditional branches in the workflow, and merge nodes to represent the merging of multiple branches.
- Add swimlanes to the diagram to represent different roles or departments involved in the process.
- Use annotations and notes to provide additional information or clarification on the activities or nodes.

- Review and refine the diagram to ensure that it accurately represents the process or workflow and achieves the intended purpose.

The main components of an activity diagram are:

- Activities: These are the specific actions or tasks that need to be performed. They are represented by rounded rectangles with descriptive labels inside.
- Control Flow: This is the directional flow of the activities, represented by arrows or lines that connect them. The arrows show the sequence in which the activities are performed. Decisions: These are points in the process where the flow can diverge based on a certain condition. They are represented by diamond shapes with labels that describe the condition. Merges: These are points where different paths in the process converge back into a single flow. They are represented by diamond shapes with multiple incoming arrows and a single outgoing arrow.
- Forks: These are points where the process splits into multiple concurrent paths. They are represented by a vertical bar with multiple outgoing arrows.
- Joins: These are points where the concurrent paths merge back into a single flow. They are represented by a vertical bar with multiple incoming arrows.
- Swimlanes: These are used to organize the activities based on the roles or responsibilities of different individuals or groups. They are represented by columns or rows with labels at the top.
- Together, these components form a visual representation of the steps or activities required to achieve a specific goal, and show how the different activities are connected and flow into each other.



RESULT: Thus, we have successfully drawn the UML activity diagram for Airline Reservation System.

EXPERIMENT NO-7

UML USE-CASE DIAGRAM

AIM: To draw a UML use case diagram for Airline Reservation System using Star UML.

DESCRIPTION:

A use case diagram is a visual representation of the functional requirements of a system. It is a type of UML (Unified Modelling Language) diagram that shows the actors (users) and use cases (functionality) of a system and how they interact with each other. The main purpose of a use case diagram is to describe the functional requirements of a system in a clear and concise way.

PURPOSE:

This system is the subsystem of the airline reservation system. The actors are passengers, admins, and the banks that are the organizations. The passenger is concerned with multiple use cases like login, check for availability, book ticket, etc. The book ticket use case is in relation to the choose seat use case. The admin cancels tickets, updates flight schedules. The bank sees the payment use cases.

The UML Use Case Diagram designed used as one of the Methodologies on Airline Reservation System development. It shows what the system processes and how it works. Users who write computer programs and users who use the Airline Reservation System will also be able to find their way around the system because of the correct labelling.

HOW TO DRAW:

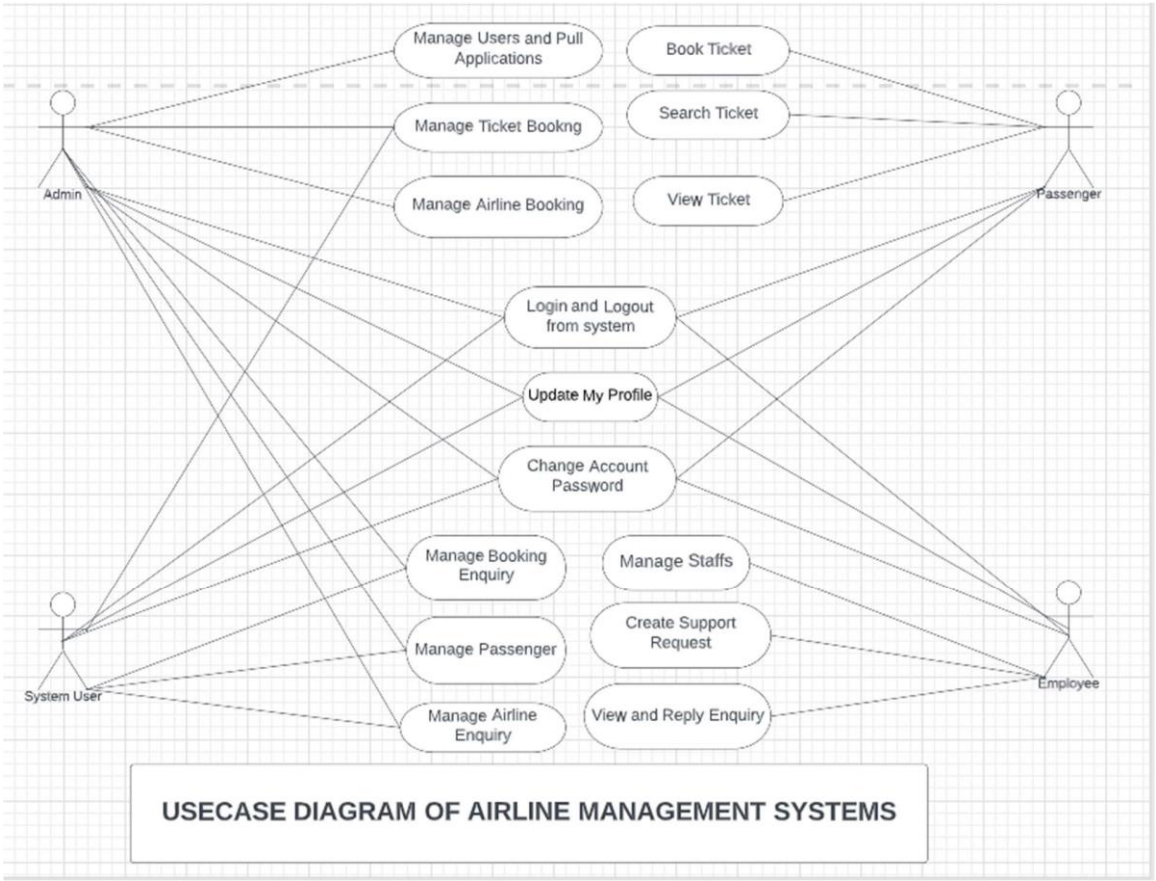
- Identify the system boundaries: The first step is to identify the boundary of the system being modelled. This involves identifying the actors that interact with the system and defining the scope of the system.
- Identify the actors: The next step is to identify the actors that interact with the system. Actors can be users, external systems, or other entities that interact with the system.
- Identify the use cases: The next step is to identify the use cases that the system must support. Use cases represent the functionality provided by the system to the actors.
- Connect the actors and use cases: Use lines to connect the actors and use cases in the diagram. These lines represent the interactions between the actors and the use cases. Use cases can be associated with one or more actors.
- Include or extend relationships: If some use cases are related to other use cases, you can indicate this using the include or extend relationship. The include relationship indicates that one use case includes the functionality of another use case. The extend relationship indicates that one use case extends the functionality of another use case.
- Refine the use case diagram: Once you have the basic structure of the use case diagram in place, you can refine it by adding more detail. This might involve adding additional actors or use cases, or adding more information to existing use cases.

The main components of a use case diagram include:

- Actors: Actors are the users or external systems that interact with the system being modelled. Actors are represented by stick figures on the diagram.
- Use Cases: Use cases are the specific functions or tasks that the system performs. Use cases are represented by ovals on the diagram.
- Relationships: Relationships describe how actors and use cases interact with each other.

There are three types of relationships in a use case diagram:

- Association: An association is a connection between an actor and a use case. It shows that the actor is involved in the use case.
- Extend: An extend relationship shows that a use case can be extended with additional functionality. This is represented by an arrow with an open arrowhead.
- Include: An include relationship shows that a use case includes another use case as a step in its process. This is represented by an arrow with a closed arrowhead.



RESULT: Thus, we have successfully drawn the UML use case diagram for Airline Reservation System.

UML CLASS DIAGRAM

AIM: To draw a UML class diagram for Airline Reservation System using Star UML.

DESCRIPTION:

A class diagram is a type of UML (Unified Modelling Language) diagram that describes the structure of a system by showing the classes in the system and the relationships between them. A class is a blueprint for creating objects that have properties and methods, and the relationships between classes describe how the objects interact with each other.

PURPOSE:

The purpose of a class diagram is to provide a high-level view of the structure and behavior of a system or application, and to illustrate the relationships between classes and objects within the system. Some of the key purposes of a class diagram include:

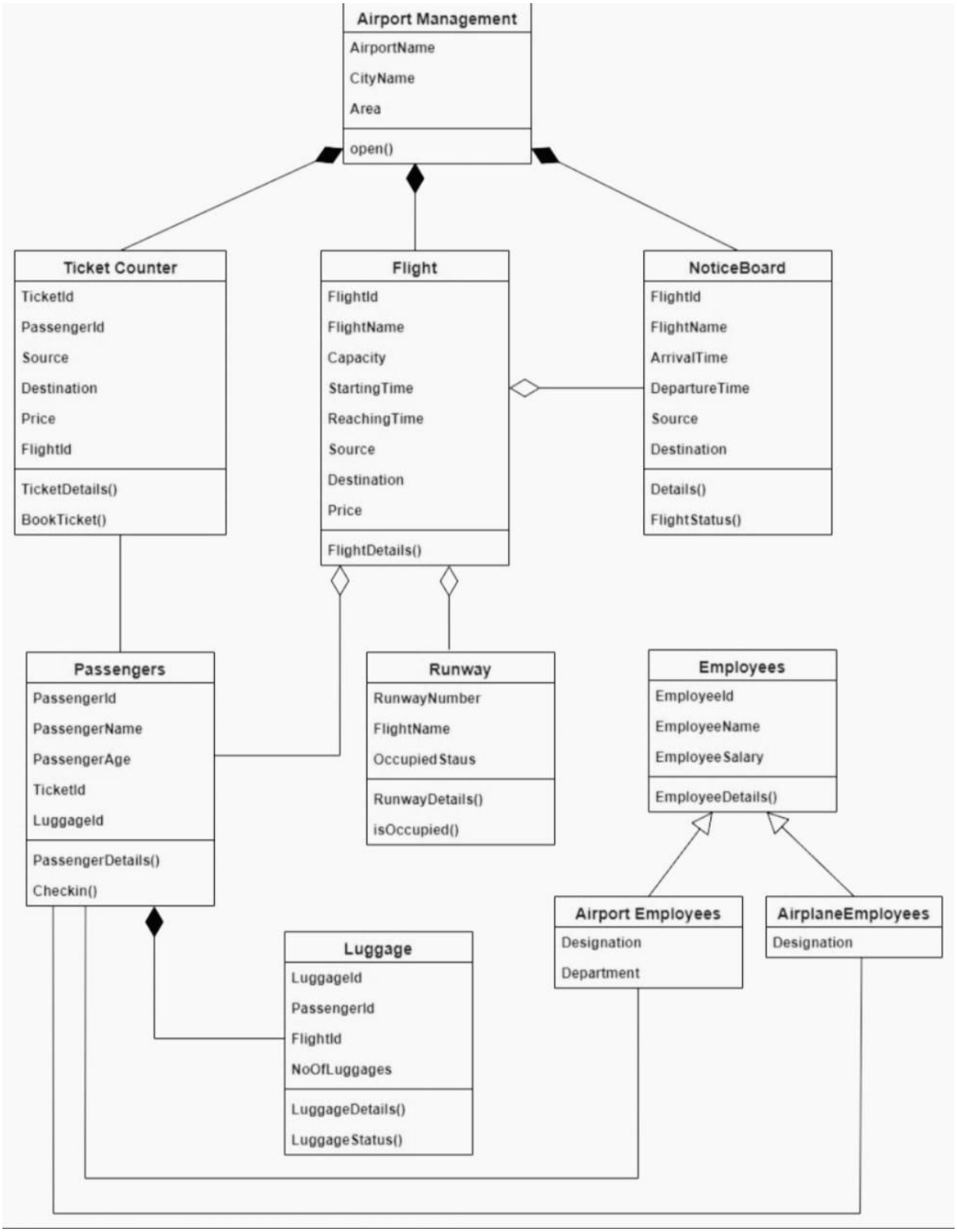
- Modeling system structure
- Visualizing system behavior
- Communication tool

HOW TO DRAW:

The airport is a complex system in which thousands of domestic and international flights function every day and it needs proper planning and execution to make it a management system.

Classes:

- **Airport Management** – This class contains the overall details of the airport.
- **Ticket Counter** – It allows the passengers to buy the ticket and do pay for a ticket.
- **Flight** – This contains all the flight details in an Airport.
- **Employees** – Employees can be two types – Airport and Airplane Employee. This class is the parent class of two subclasses – Airport Employee and Airplane Employee
- **Airport Employees** – This class is the child class of Employees. It describes the employees working in the airport. It contains their designation and departments like customs, ticket, food, etc.
- **Airplane Employees** – This class is the child class of Employee and contains the designation like Air Hostess, Pilot, etc. It indicates the employees working inside airplanes.
- **Runway** – This contains the runway details and it also tells whether the particular runway is occupied by any flights.
- **Passengers** – This class contains the passenger details.
- **Notice Board** – This class contains the current flights and the flights which are yet to arrive and depart on a particular day.
- **Luggage** – This class holds the details of luggage for a particular passenger



UML CLASS DIAGRAM FOR AIRLINE RESERVATION SYSTEM

RESULT:

Thus, we have successfully drawn the UML class diagram for Airline Reservation System.

EXPERIMENT NO-8

UML COMPONENT DIAGRAM

AIM: To draw a UML component diagram for Airline Reservation System using Star UML.

DESCRIPTION:

A component diagram is a type of UML diagram that shows the structural relationship between the components in a system. It is used to model the physical components of a system and the dependencies among them.

PURPOSE:

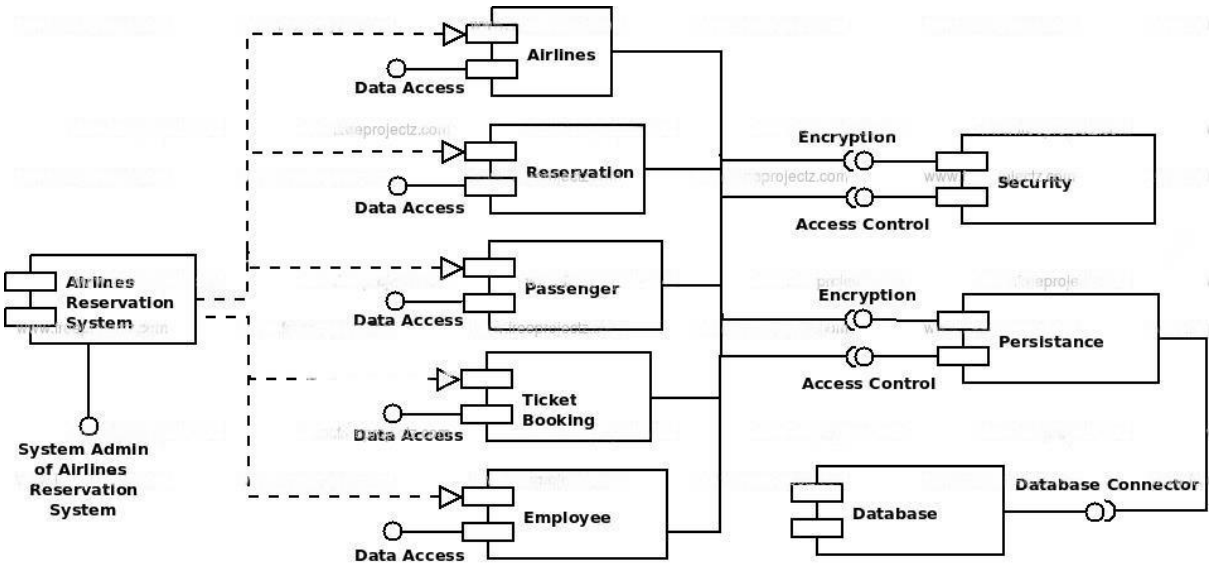
This is a Component diagram of Airline Booking System which shows components, provided and required interfaces, ports, and relationships between the Booking Enquiry, Passenger Reservation, Airlines Booking, Airline Enquiry and Ticket Booking. This type of diagrams is used in Component- Based Development (CBD) to describe systems with Service-Oriented Architecture (SOA). Airline Booking System UML component diagram, de- scribes the organization and wiring of the physical components in a system.

HOW TO DRAW:

- Identify the components: The first step is to identify the different components of the system or application that you want to model. These may include software components such as classes, modules, or libraries, or physical components such as servers, databases, or hardware devices.
- Determine the relationships: Once you have identified the components, the next step is to determine how they are related to each other. This may include dependencies, associations, or interfaces between components.
- Draw the components: The next step is to draw the components on the diagram, using the appropriate UML notation. For software components, this may include using class symbols or interface symbols, while physical components may be represented by cloud symbols or server symbols.
- Connect the components: Once you have drawn the components, the next step is to connect them with the appropriate relationships. This may include using dependency arrows, association lines, or interface symbols.
- Label the diagram: Finally, you should label the diagram with appropriate names, descriptions, and other information to help users understand the purpose and functionality of each component.

The main symbols used in a component diagram are:

- Component: It is represented as a rectangle with two parts, one part containing the name of the component and the other containing the keyword "component". A component is a modular, deployable and replaceable part of a system that encapsulates its implementation and exposes a set of interfaces.
- Interface: It is represented as a small circle on the edge of a component. An interface defines a set of operations that a component provides or requires from other components.
- Dependency: It is represented as an arrow between two components, indicating that one component depends on another component. A dependency can be either a requirement or a
- realization. A requirement is a dependency in which one component needs the services of another component to function properly. A realization is a dependency in which one component implements the services provided by another component.
- Connector: It is represented as a line between two ports or interfaces, indicating that they are connected. A connector can be either a delegation or an assembly. A delegation is a connector in which one component delegates some of its operations to another component. An assembly is a connector in which one component is composed of several sub-components.
- Package: It is represented as a folder-like icon, used to group related components together.



Component Diagram of Airlines Reservation System

RESULT: Thus, we have successfully drawn the UML component diagram for Airline Reservation System.

EXPERIMENT NO-9

UML INTERACTION DIAGRAM

AIM: To draw a UML interaction diagram for Airline Reservation System using Star UML.

DESCRIPTION:

An interaction diagram is a graphical representation of the interactions that take place between objects in a system or between actors in a use case. It shows how objects or actors collaborate with each other to achieve a specific goal or complete a certain task. Interaction diagrams are a part of the Unified Modelling Language (UML) and are commonly used during the design phase of software development to visualize and communicate the behaviour of a system.

PURPOSE:

The purpose of interaction diagrams in UML is to model the dynamic behavior of a system or application by showing how objects interact with each other over time. Interaction diagrams provide a graphical representation of the flow of messages and actions between objects, and they can help developers better understand the logic and behavior of a system.

There are two types of interaction diagrams in UML: sequence diagrams and communication diagrams. Sequence diagrams focus on the time-based sequence of messages exchanged between objects or actors, while communication diagrams illustrate the interactions as a set of objects and the messages that they exchange.

Interaction diagrams help to visualize and analyse the behaviour of a system, identify potential problems or bottlenecks, and improve communication among team members. They are also useful for testing and verifying system behaviour before implementation.

The Interaction Diagram depicts the process interactions arranged in time sequence in the field of software engineering. It depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionality.

This UML interaction diagram of Airlines Reservation System which shows the interaction between the objects of Ticket Booking, Airlines, Passenger, Reservation, Employee. The instance of class objects involved in this UML Sequence Diagram of Airline Reservation System are as follows:

- Booking System
- Flight
- Customer Database
- Reservation System

HOW TO DRAW:

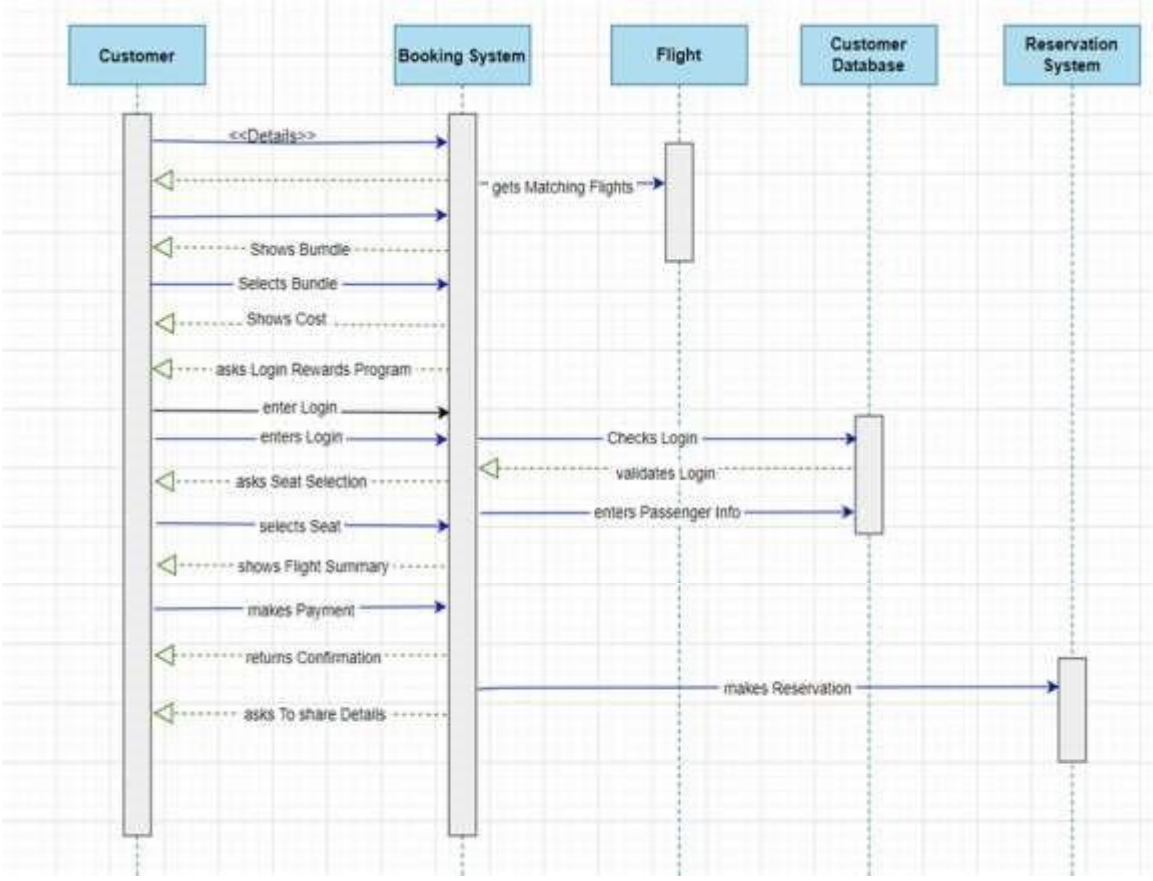
Sequence Diagram:

- Identify the objects: The first step is to identify the objects that will be included in the sequence diagram. These are typically the objects that are involved in the interaction that you want to model.

- Identify the messages: Next, you need to identify the messages that are exchanged between the objects. These messages represent the actions that the objects perform during the interaction.
- Draw the lifelines: Each object in the sequence diagram is represented by a vertical line called a "lifeline". Draw a lifeline for each object, and label each lifeline with the name of the corresponding object.
- Add the messages: Draw horizontal arrows between the lifelines to represent the messages that are exchanged between the objects. Label each arrow with the name of the message.

The components of each type of interaction diagram are as follows:

- Lifeline: A vertical line that represents an object or participant involved in the sequence of actions in the system.
- Execution Occurrence: A rectangle that represents the time interval during which an object performs a particular action.
- Message: A horizontal arrow that represents a communication between two lifelines, indicating the direction of the message and its name.



UML INTERACTION FOR AIRLINE RESERVATION SYSTEM

RESULT: Thus, we have successfully drawn the UML interaction diagram for Airline Reservation System

EXPERIMENT NO-10

UML COLLABORATION DIAGRAM

AIM: To draw a UML collaboration diagram for Airline Reservation System using Star UML.

DESCRIPTION:

A collaboration diagram is a type of interaction diagram in UML (Unified Modeling Language) that depicts the interactions and relationships among objects participating in a particular scenario or use case.

It visually represents the interactions between objects and how they collaborate to accomplish a specific task or functionality. Collaboration diagrams are also known as communication diagrams as they show how objects communicate with each other to achieve a common goal.

PURPOSE:

The purpose of a collaboration diagram in UML is to illustrate the interactions and relationships between objects or components in a system. It shows the various objects or components involved in a particular scenario and how they communicate with each other to achieve a common goal.

The collaboration diagram is often used to model the dynamic behaviour of a system, showing how objects collaborate with each other in order to complete a particular task or scenario. It allows developers and stakeholders to visualize the flow of control and communication between different objects or components, which can help them to identify potential issues and improve the overall design of the system.

In a collaboration diagram, the objects are represented as boxes, and the messages exchanged between them are represented as arrows. The sequence of messages is shown by numbering the arrows. Additionally, the lifelines of the objects are shown, which represents the time period during which the object is active or involved in the collaboration.

Collaboration diagrams are useful in understanding complex systems, analysing object interactions, identifying communication gaps, and for designing and testing software applications. They provide a clear picture of the objects involved in the system and how they interact with each other, making them an important tool for software developers and system architects.

HOW TO DRAW:

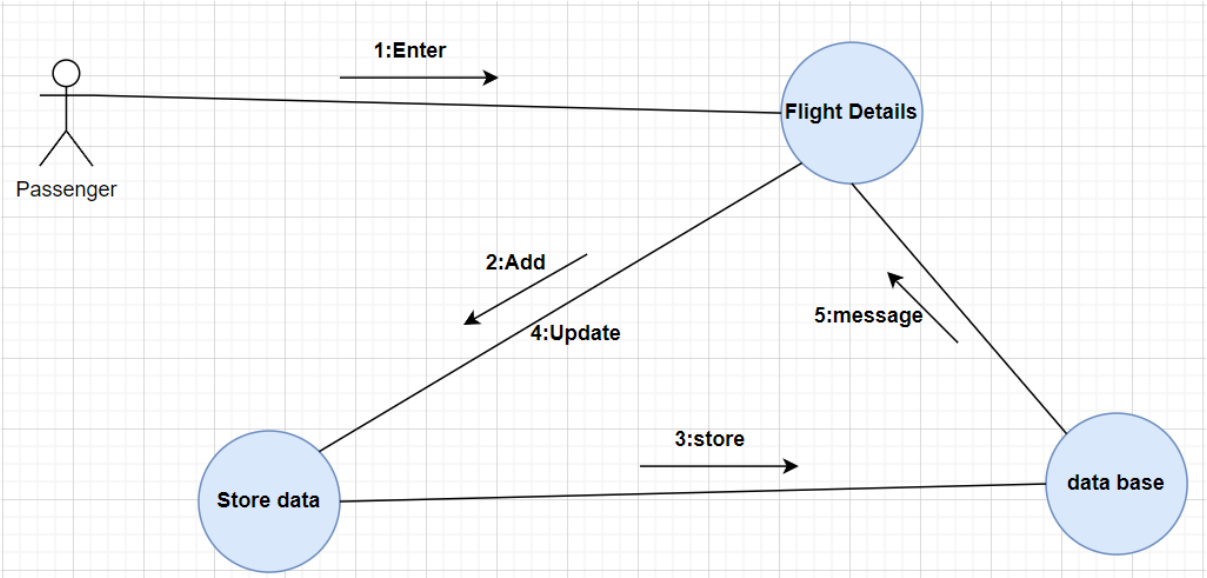
Identify the objects: The first step is to identify the objects that will be included in the collaboration diagram. These are typically the objects that are involved in the interaction that you want to model.

Draw the objects: Draw a box for each object, and label each box with the name of the corresponding object.

Add the messages: Draw lines between the boxes to represent the messages that are exchanged between the objects. Label each line with the name of the message.

The components of each type of interaction diagram are as follows:

- Object: A rectangular box that represents an object or participant involved in the collaboration.
- Link: A line that connects two objects and represents a communication or association between them.
- Message: A numbered arrow that represents the order and direction of messages being sent between objects.
- Multiplicity: A notation that represents the number of instances of an object that participate in the collaboration.
- Role Name: A name assigned to an object to help identify its purpose or role in the collaboration.



UML COLLABORATION DIAGRAM FOR AIRLINE RESERVATION SYSTEM

RESULT: Thus, we have successfully drawn the UML collaboration diagram for Airline Reservation System.

EXPERIMENT NO-11

UML DEPLOYMENT DIAGRAM

AIM: To draw a UML deployment diagram for Airline Reservation System using Star UML.

DESCRIPTION:

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships. It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.

PURPOSE:

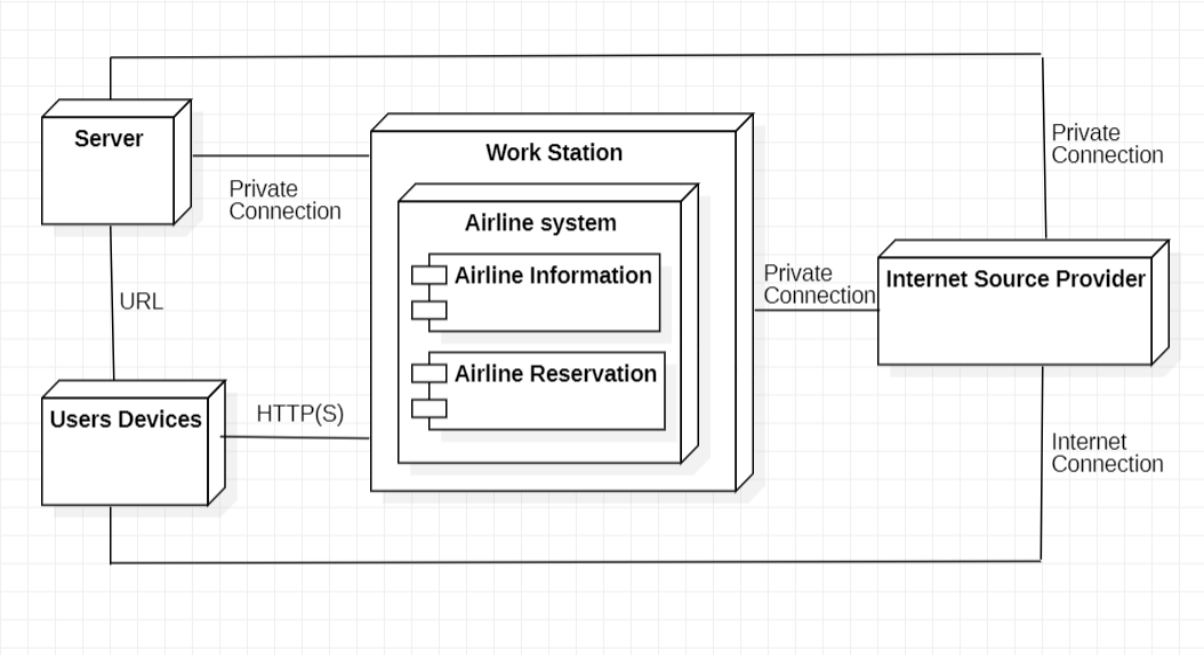
In this deployment diagram, the user's device and workstation communicate with the web server over HTTP to access the airline reservation system. The web server communicates with the application server using RMI/IIOP to execute the business logic of the system. The ISP provides the internet connectivity for the system and communicates with the airline reservation system using HTTP. The airline reservation system includes the application server, EJBs, and the database server, which store and retrieve data related to airline reservations.

This deployment diagram shows how the various components of the airline reservation system are physically deployed on the target environment, including the hardware and software components and their interconnections. It is a useful tool for software architects and developers to ensure that the physical architecture of the system is well-structured, efficient, and easily maintainable.

HOW TO DRAW:

- Identify the software components and hardware nodes: Start by identifying the software components (such as applications, databases, middleware, etc.) and hardware nodes (such as servers, routers, switches, etc.) that will be involved in the deployment.
- Draw the nodes: Use rectangles to represent the hardware nodes and place them on the diagram.
- Add the components: Use component symbols (rectangles with a smaller rectangle inside) to represent the software components and place them on the diagram. Label each component with a descriptive name and include any necessary details about its functionality.
- Add the deployment relationships: Use lines to show the deployment relationships between the components and the nodes. These relationships can be either direct (when a component is deployed on a specific node) or indirect (when a component is deployed on multiple nodes or on a cluster).

Components of deployment diagram:
Nodes: Nodes represent the physical hardware devices or software environments on which the system components are deployed.
Components: Components represent the software artifacts that are deployed to the nodes.
Artifacts: Artifacts are the physical files, libraries, or other software elements that make up a component
Deployment Relationships: Deployment relationships show how the components are deployed onto the nodes. These relationships include associations, dependencies, and generalizations.
Communication Paths: Communication paths show how nodes communicate with each other over the network.



UML DEPLOYMENT DIAGRAM FOR AIRLINE RESERVATION SYSTEM

RESULT: Thus, we have successfully drawn the UML DEPLOYMENT diagram for Airline Reservation System.

EXPERIMENT NO-12

UML SEQUENCE DIAGRAM WITH THREADS

AIM: To draw a UML sequence diagram with threads for Airline Reservation system using Star UML.

DESCRIPTION:

In a sequence diagram, a thread represents a separate flow of execution or control within the system being modelled. Each thread typically corresponds to a separate unit of functionality or process within the system and can interact with other threads and objects in the system through method calls and message exchanges. Threads are often used in sequence diagrams to illustrate concurrent or parallel processing, where multiple threads are executing simultaneously and interacting with each other and the rest of the system. Each thread is represented as a separate vertical line on the diagram, with arrows or messages indicating the flow of control and communication between threads and objects.

PURPOSE:

The sequence diagram with threads for an airline reservation system describes the flow of events and interactions that take place when a user makes a reservation. The diagram is divided into several threads that handle different tasks, such as searching for flights, reserving seats, and updating flight availability.

The main thread controls the overall process and begins when the user selects a flight. The system then creates a separate thread to search for available flights on the selected dates and destination. Once the search is complete, the system creates another thread to display the available flight options to the user.

Once the reservation is complete, the system creates another thread to update the availability of seats on the selected flight. A confirmation page is displayed to the user, and the system creates a thread to send a confirmation email to the user. Finally, the reservation process is completed, and the threads are terminated.

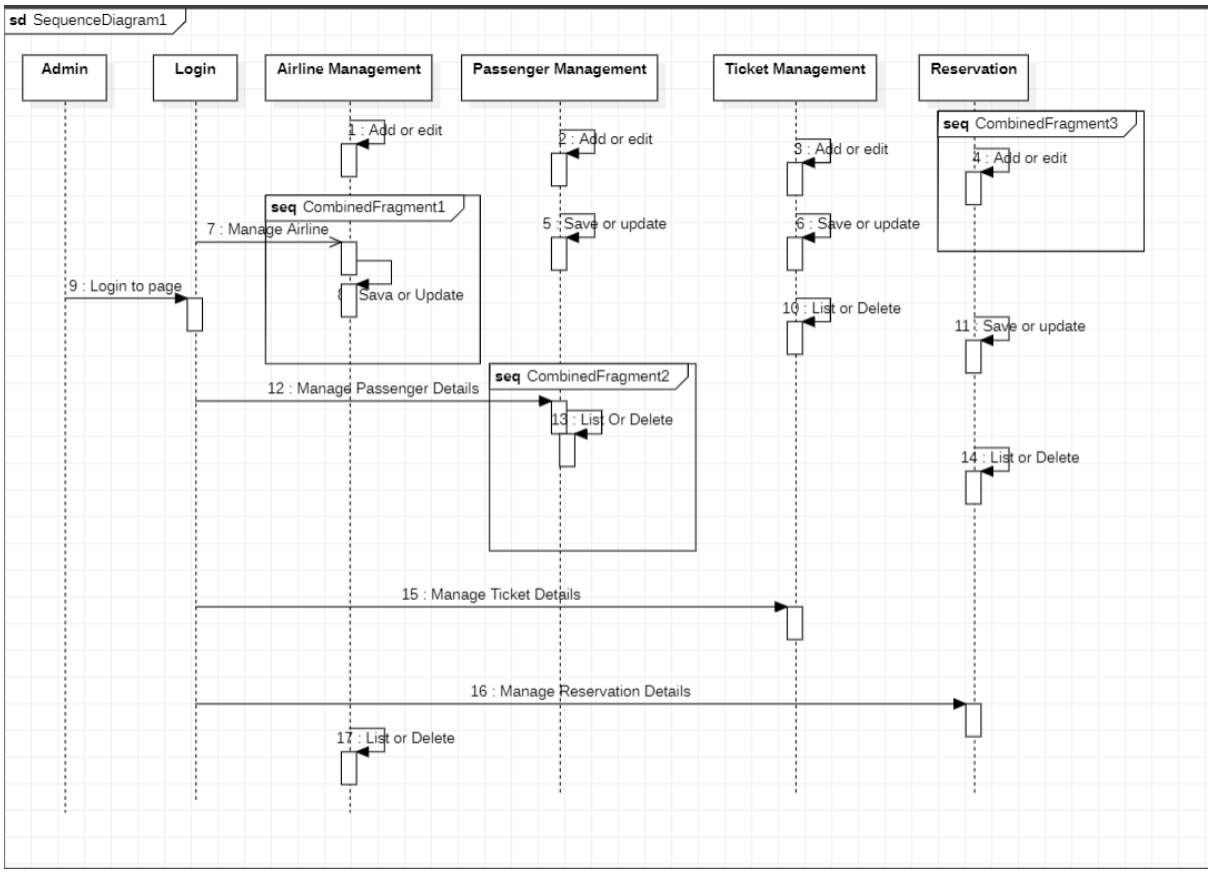
Overall, the sequence diagram with threads for an airline reservation system helps to illustrate how a complex process, such as booking a flight, can be broken down into smaller, manageable tasks that can be handled concurrently by different threads. This approach can help to improve the performance and efficiency of the system and provide a better user experience for the customer.

HOW TO DRAW:

- Identify the different threads that are involved in the scenario you are modeling. For example, if you are modeling a banking system, you may have separate threads for the customer, the teller, and the ATM.
- Draw a vertical line for each thread, running from the top of the diagram to the bottom.
- Identify the messages or actions that are associated with each thread. For example, if the customer wants to withdraw money from their account, they may send a message to the teller thread, which will in turn send a message to the banking system thread.
- Draw horizontal arrows that cross the thread lines to represent each message or action. Label each arrow with a description of the message or action.
- Use dotted lines to represent asynchronous messages or actions, which do not block the thread while they are being processed.
- Add any additional components, such as external systems, databases, or middleware, that are involved in the scenario.

The components of a sequence diagram can include:

- Objects: Objects represent the different entities or actors that are involved in the scenario. For example, in a banking system, you may have objects for the customer, the teller, and the ATM.
- Messages: Messages represent the communication between the different objects. Each message is labeled with a description of the action being performed.
- Activation boxes: Activation boxes represent the period of time during which an object is active and processing a message. The top and bottom of the box represent the start and end of the processing period.
- Lifelines: Lifelines represent the existence of an object over time. Each object is associated with a lifeline, which runs vertically through the diagram.
- Combined fragments: Combined fragments are used to represent alternative or parallel paths in the sequence diagram. They are represented by boxes with different types of markers, such as "alt" or "par".
- Constraints: Constraints are used to specify any conditions or limitations on the behavior of the system or objects in the scenario. They can be represented by annotations on the diagram.



UML SEQUENCE DIAGRAM WITH THREADS FOR AIRLINE RESERVATION SYSTEM

RESULT: Thus, we have successfully drawn the UML sequence diagram with threads for Airline Reservation System.