Practical 9

Behavioural View diagram

Aim: Design the behavioural view diagram for the selected system. Use following diagrams
1. Interaction Diagrams a. Sequence Diagrams b. Collaboration Diagram
2. State–Chart Diagrams
3. Activity Diagrams

Behavioural diagrams are used to visualize, specify, construct, and document the dynamic aspects of a system. It shows how the system behaves and interacts with itself and other entities (users, other systems). They show how data moves through the system, how objects communicate with each other, how the passage of time affects the system, or what events cause the system to change internal states. Since behaviour diagrams illustrate the behaviour of a system, they are used extensively to describe the functionality of software systems. As an example, the activity diagram describes the business and operational step-by-step activities of the components in a system. In other words, a behavioural diagram shows how the system works 'in motion', that is how the system interacts with external entities and users, how it responds to input or event and what constraints it operates under.

9.1 Interaction Diagrams

The purpose of interaction diagrams is to visualize the interactive behaviour of the system Sequence and collaboration diagrams are used to capture the dynamic nature but from a different angle.

Terminology of Interaction diagram:
- An interaction diagram contains lifelines, messages, operators, state invariants and constraints.

Lifeline:

A lifeline represents a single participant in an interaction. It describes how an instance of a specific classifier participates in the interaction. A life line represents a role that an instance of the classifier may play in the interaction.

Various attributes of a lifeline,

❖ **Name**
- It is used to refer to the life line within a specific interaction.
- The name of a life line is optional.

❖ **Type**
- It is the name of a classifier of which the life line represents an instance.

❖ **Selector**
- It is a Boolean condition which is used to select a particular instance that satisfies the requirement.
- Select or attribute is also optional.

Messages

- A message is a specific type of communication between two lifelines in an interaction. A message involves following
- A call message which is used to call an operation.
- A message to create an instance.
- A message to destroy an instance.
- For sending a signal.

Message Type	Description	Example
Synchronous	Sender waits for receiver to complete before continuing	objectA.method()
Asynchronous	Sender continues without waiting for response	sendMessage()
Return	Represents return of control or data from receiver to sender	return value
Create	Indicates creation of a new object	new Object()
Delete	Indicates termination of an object	object.destroy()
Self-Message	An object sends a message to itself	this.doSomething()
Found Message	Message originates outside the system or is unidentified	External API call
Lost Message	Message is sent but the destination is unknown or not modeled	Log message not saved

State in variants and constraints

- When an instance or a lifeline receives a message, it can cause it to change the state.
- A state is a condition or a situation during a life time of an object at which it satisfies some constraint, performs some operations, and waits for some event.
- In an interaction diagram, not all messages cause changes to change the state of an instance.

Some messages do not have the values of some attributes. It has no side effects on the state of an object.

Operator

- An operator specifies an operation on how the operands are going to be executed.
- The operators in UML support operations on data in the form of branching as well as an iteration. Various operators can be used to ensure the use of iteration and branching in the UML model.

Iteration

- In an interaction diagram, we can also show iteration using an iteration expression.
- An iteration expression consists of an iteration specifier and an optional iteration clause. There is no pre-specified syntax for UML iteration.

Branching

- In an interaction diagram, we can represent branching by adding guard conditions to the messages. Guard conditions are used to check if a message can be sent forward or not.
- A message is sent forward only when its guard condition is true.

- A message can have multiple guard conditions, or multiple messages can have the same guard condition.
- Branching in UML is achieved with the help of alt and opt, operators.

Types of Interaction diagram:

- Different types of interaction diagrams defined in UML:
- Sequence diagram
- Collaboration diagram
- Timing diagram

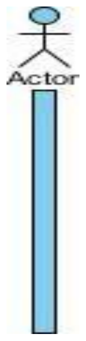
9.1.1 Sequence diagram

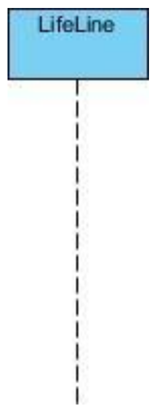
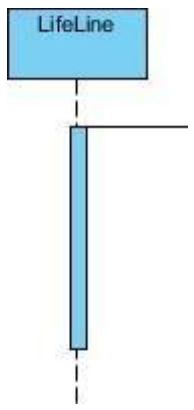
Sequence Diagrams are interaction diagrams that detail how operations are carried out.They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

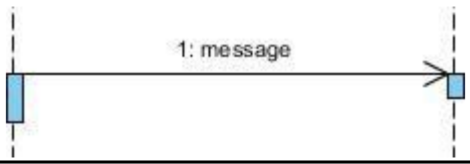
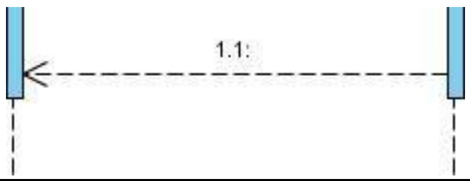
Purpose of Sequence Diagram

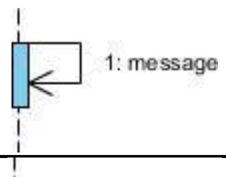
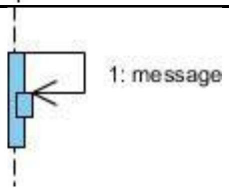
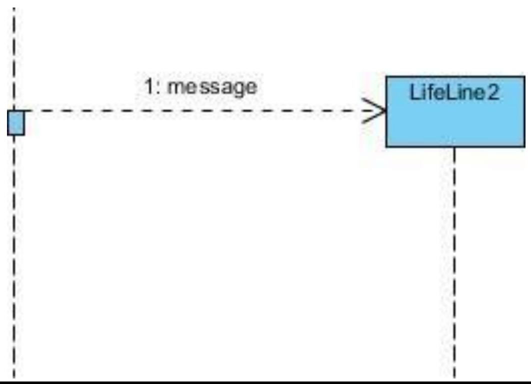
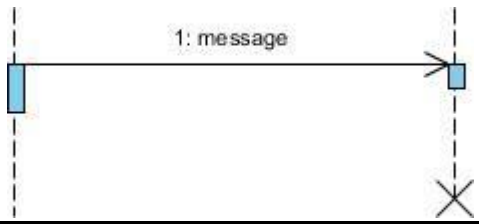
- Model high-level interaction between active objects in a system
- Model the interaction between object instances within a collaboration that realizes a use case
- Model the interaction between objects within a collaboration that realizes an operation
- Either model generic interactions (showing all possible paths through the interaction) or specific instances of a interaction (showing just one path through the interaction)

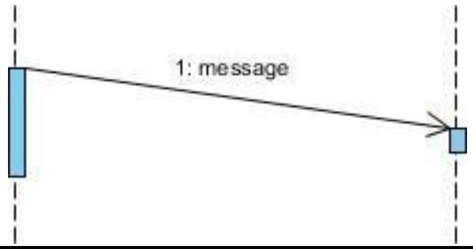
Sequence Diagram Notation

Notation	**Symbol**
Actor ● A type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data) ● External to the subject (i.e., in the sense that an instance of an actor is not a part of the instance of its corresponding subject). ● Represent roles played by human users, external hardware, or other subjects.	

Lifeline ● A lifeline represents an individual participant in the Interaction.	
Activations ● A thin rectangle on a lifeline) represents the period during which an element is performing an operation. ● The top and the bottom of the of the rectangle are aligned with the initiation and the completion time respectively	

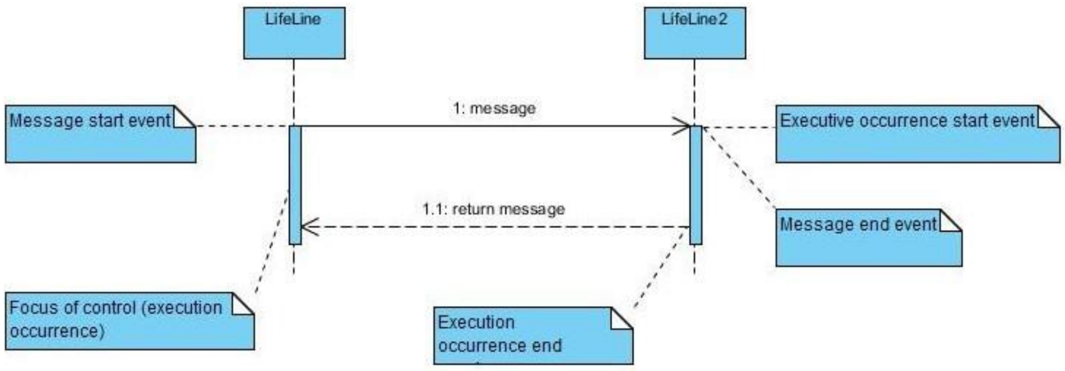
Call Message ● A message defines a particular communication between Lifelines of an Interaction. ● Call message is a kind of message that represents an invocation of operation of target lifeline.	
Return Message ● A message defines a particular communication between Lifelines of an Interaction. ● Return message is a kind of message that represents the pass of information back to the caller of a corresponded former message.	

Self Message ● A message defines a particular communication between Lifelines of an Interaction. ● Self message is a kind of message that represents the invocation of message of the same lifeline.	
Recursive Message ● A message defines a particular communication between Lifelines of an Interaction. ● Recursive message is a kind of message that represents the invocation of message of the same lifeline. It's target points to an activation on top of the activation where the message was invoked from.	
Create Message ● A message defines a particular communication between Lifelines of an Interaction. ● Create message is a kind of message that represents the instantiation of (target) lifeline.	
Destroy Message ● A message defines a particular communication between Lifelines of an Interaction. ● Destroy message is a kind of message that represents the request of destroying the lifecycle of target lifeline.	
Duration Message ● A message defines a particular communication between Lifelines of an Interaction.	

• Duration message shows the distance between two time instants for a message invocation.	
Note A note (comment) gives the ability to attach various remarks to elements. A comment carries no semantic force, but may contain information that is useful to a modeler.	

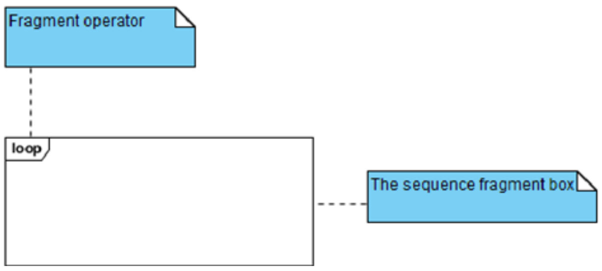
Message and Focus of Control

- An Event is any point in an interaction where something occurs.
- Focus of control: also called execution occurrence, an execution occurrence
- It shows as tall, thin rectangle on a lifeline)
- It represents the period during which an element is performing an operation. The top and the bottom of the rectangle are aligned with the initiation and the completion time respectively.



Sequence Fragments

- Sequence fragments make it easier to create and maintain accurate sequence diagrams
- A sequence fragment is represented as a box, called a combined fragment, which encloses a portion of the interactions within a sequence diagram
- The fragment operator (in the top left cornet) indicates the type of fragment
- Fragment types: ref, assert, loop, break, alt, opt, neg



Operator	Fragment Type
alt	Alternative multiple fragments: only the one whose condition is true will execute.
opt	Optional: the fragment executes only if the supplied condition is true. Equivalent to an alt only with one trace.
par	Parallel: each fragment is run in parallel.
loop	Loop: the fragment may execute multiple times, and the guard indicates the basis of iteration.
region	Critical region: the fragment can have only one thread executing it at once.
neg	Negative: the fragment shows an invalid interaction.
ref	Reference: refers to an interaction defined on another diagram. The frame is drawn to cover the lifelines involved in the interaction. You can define parameters and a return value.
sd	Sequence diagram: used to surround an entire sequence diagram.

9.1.2 Collaboration Diagram (Communication diagram)

Collaboration diagrams (known as Communication Diagram) are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.

- A Collaboration is a collection of named objects and actors with links connecting them. They collaborate in performing some task.
- A Collaboration defines a set of participants and relationships that are meaningful for a given set of purposes
- A Collaboration between objects working together provides emergent desirable functionalities in Object-Oriented systems
- Each object (responsibility) partially supports emergent functionalities
- Objects are able to produce (usable) high-level functionalities by working together
- Objects collaborate by communicating (passing messages) with one another in order to work together

Notations of Collaboration Diagram:

Objects

An object is represented by an object symbol showing the name of the object and its class underlined, separated by a colon:

<u>Object_name : class_name</u>

You can use objects in collaboration diagrams in the following ways:

- Each object in the collaboration is named and has its class specified
- Not all classes need to appear
- There may be more than one object of a class
- An object's class can be unspecified. Normally you create a collaboration diagram with objects first and specify their classes later.
- The objects can be unnamed, but you should name them if you want to discriminate different objects of the same class.

Actors

Normally an actor instance occurs in the collaboration diagram, as the invoker of the interaction. If you have several actor instances in the same diagram, try keeping them in the periphery of the diagram.

- Each Actor is named and has a role
- One actor will be the initiator of the use case

Links

Links connect objects and actors and are instances of associations and each link corresponds to an association in the class diagram Links are defined as follows:

- A link is a relationship among objects across which messages can be sent. In collaboration diagrams, a link is shown as a solid line between two objects.
- An object interacts with, or navigates to, other objects through its links to these objects.
- A link can be an instance of an association, or it can be anonymous, meaning that its association is unspecified.
- Message flows are attached to links, see Messages.

Messages

A message is a communication between objects that conveys information with the expectation that activity will ensue. In collaboration diagrams, a message is shown as a labeled arrow placed near a link.

- The message is directed from sender to receiver
- The receiver must understand the message
- The association must be navigable in that direction

Steps for Creating Collaboration Diagrams

1. Identify behavior whose realization and implementation is specified
2. Identify the structural elements (class roles, objects, subsystems) necessary to carry out the functionality of the collaboration
    - Decide on the context of interaction: system, subsystem, use case and operation
3. Model structural relationships between those elements to produce a diagram showing the context of the interaction
4. Consider the alternative scenarios that may be required ● Draw instance level collaboration diagrams, if required.

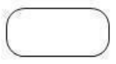
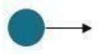
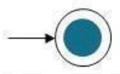
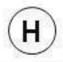
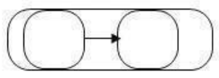
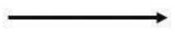
- Optionally draw a specification level collaboration diagram to summarize the alternative scenarios in the instance level sequence diagrams

9.2 State–Chart Diagrams

Purpose of State-Chart diagram:

State-chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State-chart diagram is to model lifetime of an object from creation to termination.

Notations of State-Chart diagram:

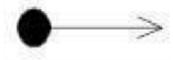
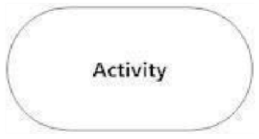
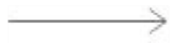
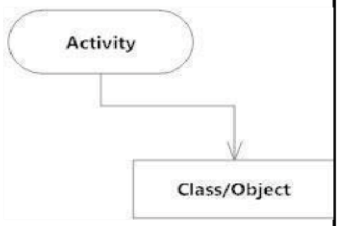
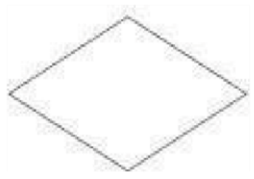
Elements	Description	UML Notation
States	The set of properties and values describing the object in a well defined instant.	
Initial State	The initial state shows the starting point or first activity of the flow.	
Final State	The final state corresponding completion transition on the enclosing state.	
History State	A history state is used to remember the previous state of a state machine when it was interrupted	
Composite State	A composite state is a state that contains other states (sub-states).	
Transition	A transition is a directed relation between a source state and a destination state.	

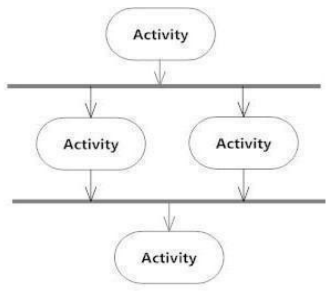
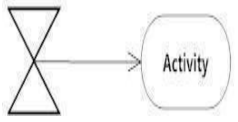
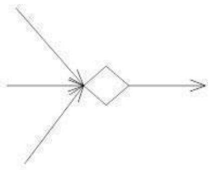
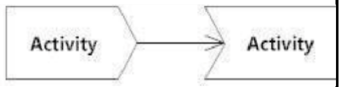
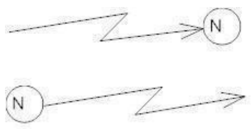
9.3 Activity Diagram

Purpose of Activity Diagram:

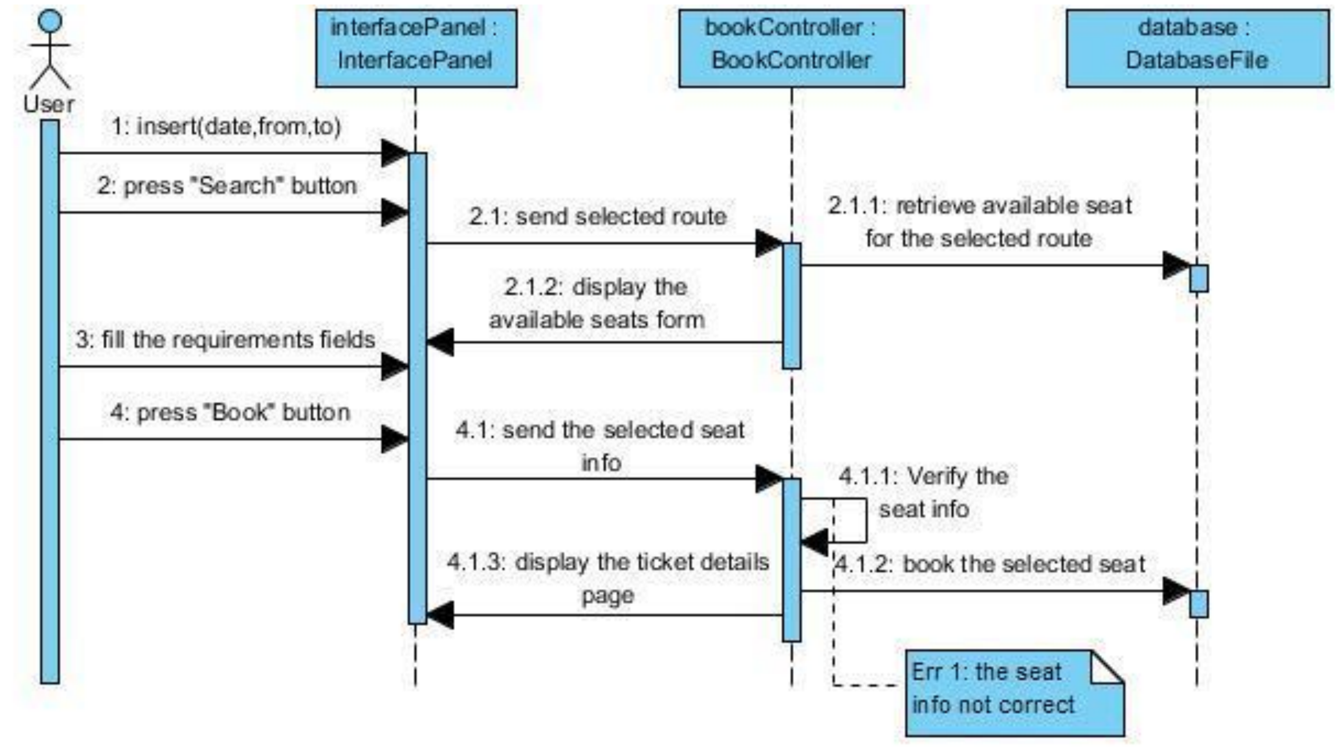
An activity diagram shows business and software processes as a progression of actions. These actions can be carried out by people, software components or computers. Activity diagrams are used to describe business processes and use cases as well as to document the implementation of system processes.

Notations of Activity Diagram:

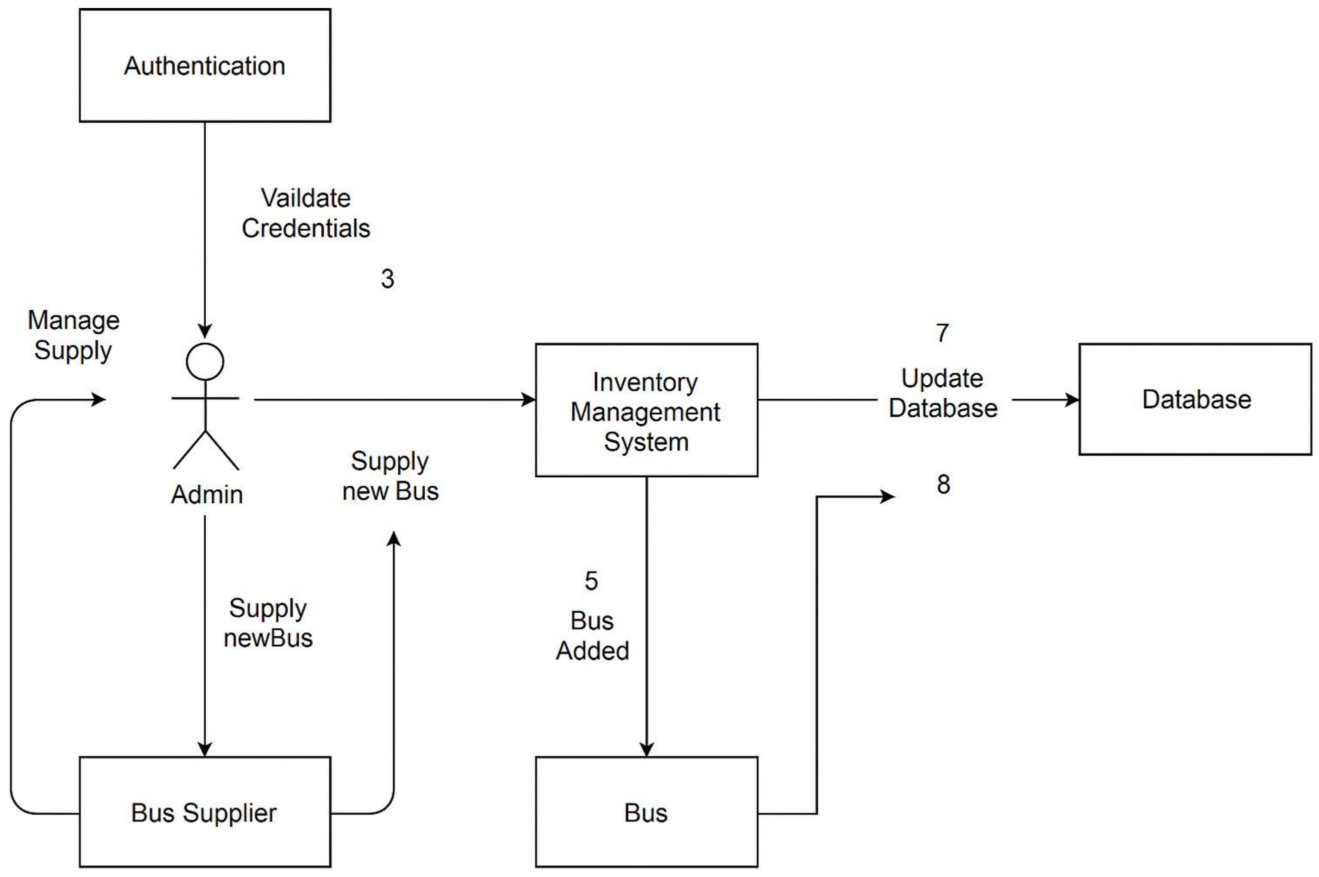
Notation Description	**Visual Representatio n**
Initial State or Start Point ● A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram.	
Activity or Action State ● Anactionstaterepresentsthenon-interruptibleactionofobjects.Youcandrawanactionstateusingarectanglewith rounded corners.	
Action Flow ● It Action flows, also called edges and paths, illustrate the transitions from one action state to another. They are usually drawn with an arrowed line.	
Object Flow ● Object flow refers to the creation and modification of objects by activities. An object flow arrow from anactiontoanobjectmeansthattheactioncreatesorinfluencestheobject.	
Decisions and Branching ● A diamond represents a decision with alternate paths. When an activity requires a decision prior to moving on to the next activity, add a diamond between the two activities.	

Synchronization ● A join node joins multiple concurrent flows back in to a single outgoing flow. ● A fork and join mode used together are often referred to as synchronization.	
Time Event ● This refers to an event that stops the flow for a time; an hour glass depicts it.	
Merge Event ● A merge event brings together multiple flows that are not concurrent.	
Sent and Received Signals ● They appear in pairs of sent and received signals, because the state can't change until a response is received.	
Interrupting Edge ● They An event, such as a cancellation, that interrupts the flow denoted with a lightning bolt.	

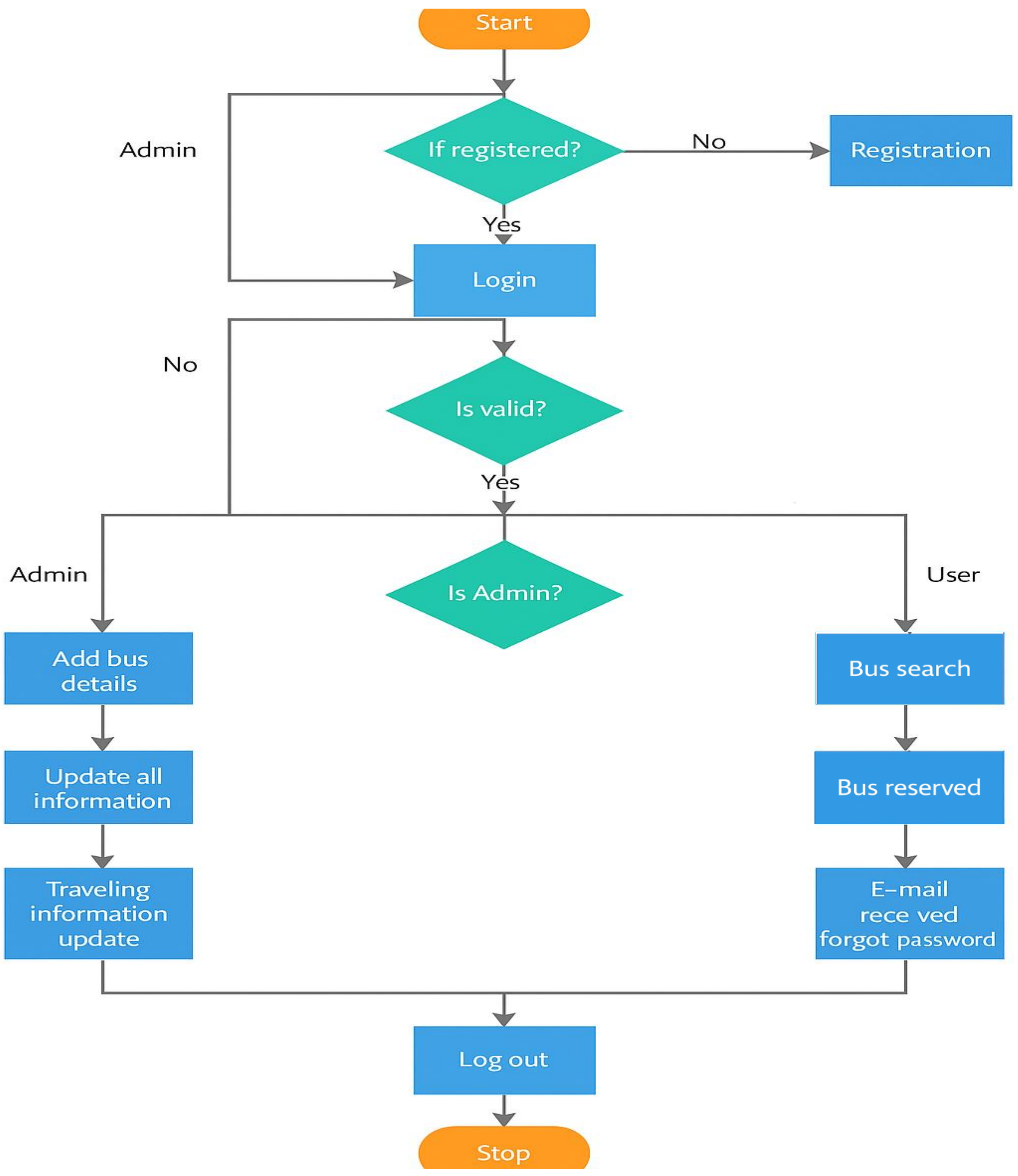
Sequence Diagram of Bus Management System


User
interfacePanel : InterfacePanel
bookController : BookController
database : DatabaseFile
1: insert(date,from,to)
2: press "Search" button
2.1: send selected route
2.1.1: retrieve available seat for the selected route
2.1.2: display the available seats form
3: fill the requirements fields
4: press "Book" button
4.1: send the selected seat info
4.1.1: Verify the seat info
4.1.2: book the selected seat
4.1.3: display the ticket details page
Err 1: the seat info not correct

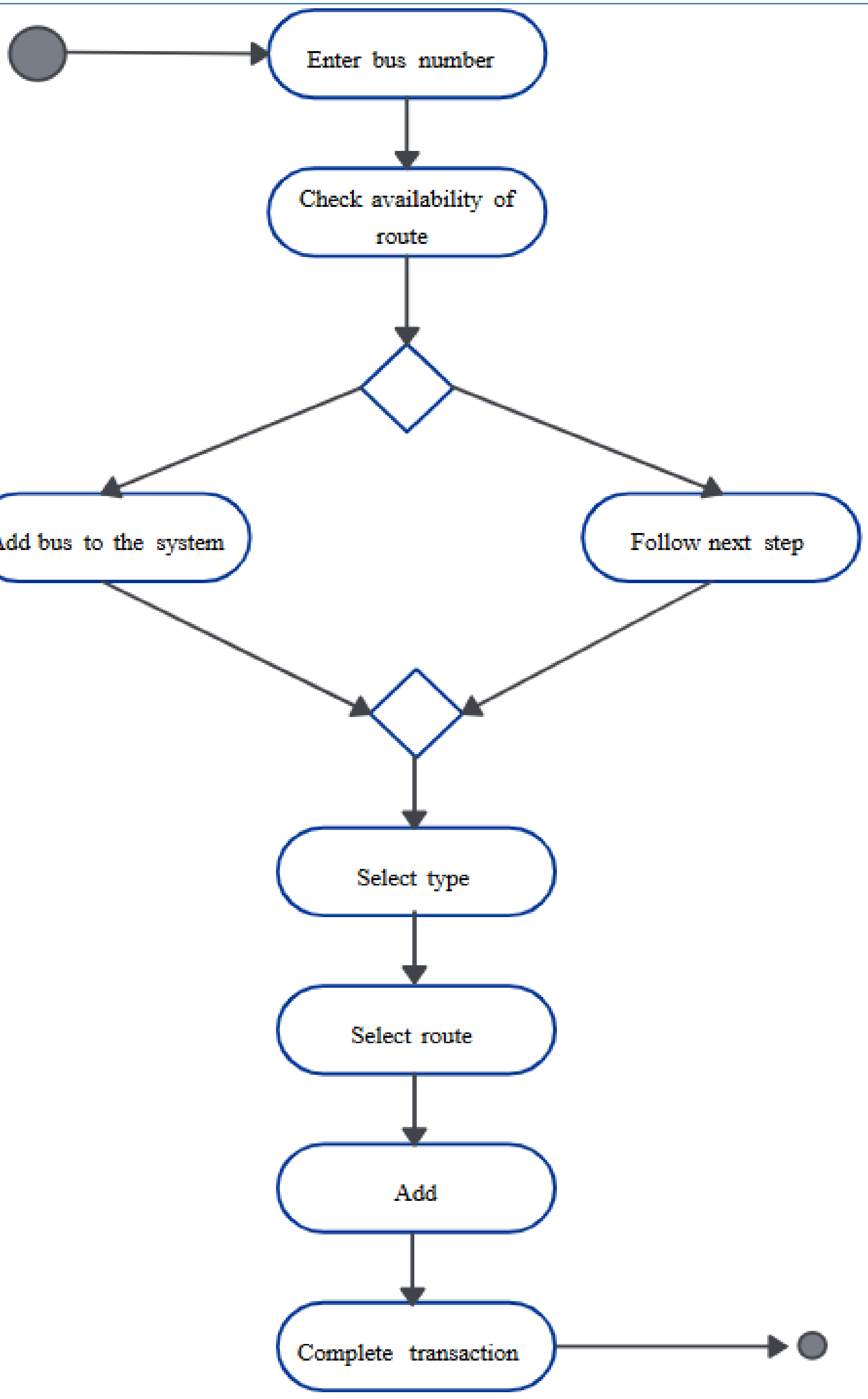
Collaboration Diagram for Bus Management system



State chart Diagram for Bus Management system



Activity Diagram for Bus Management System


Enter bus number
Check availability of route
Add bus to the system
Follow next step
Select type
Select route
Add
Complete transaction