CLASS DIAGRAM

A collection of classes, interfaces, affiliations, collaborations, and restrictions are portrayed in a class diagram. It also is referred to as "structural diagram."

Class diagrams is static diagrams. It represents the application's static view. Class diagrams are used to create executable code for software applications as well as for visualizing, explaining, and documenting diverse elements of systems.

The characteristics and functions of a class are described in a class diagram, along with the restrictions placed on the system. Because they are the only UML diagrams that can be directly mapped with object-oriented languages, class diagrams are frequently used in the modeling of object-oriented systems.

CLASS:

A class is a design or a set of principles for creating a specific type of object (creating objects a particular data structure), providing initial values for state (member variables or attributes), and implementations of behavior (member functions or methods)). It is the fundamental idea behind object-oriented programming.

To identify the candidates as a class:

Physical entities, or a library of tangible objects, and devices with which the program interacts are required in order to classify the candidates: Books, automobiles, pressure sensors, etc.

Attributes:

The information or structure of a class is represented by an attribute. It can be recognized using any of these three methods.

Examining the entry requirements, meeting the prerequisites, and using domain knowledge

Demonstrating method:

A process connected to a message and an object is called a method. Method overriding, which enables the use of the same name for several distinct classes, is one of the most prominent components that a method offers.

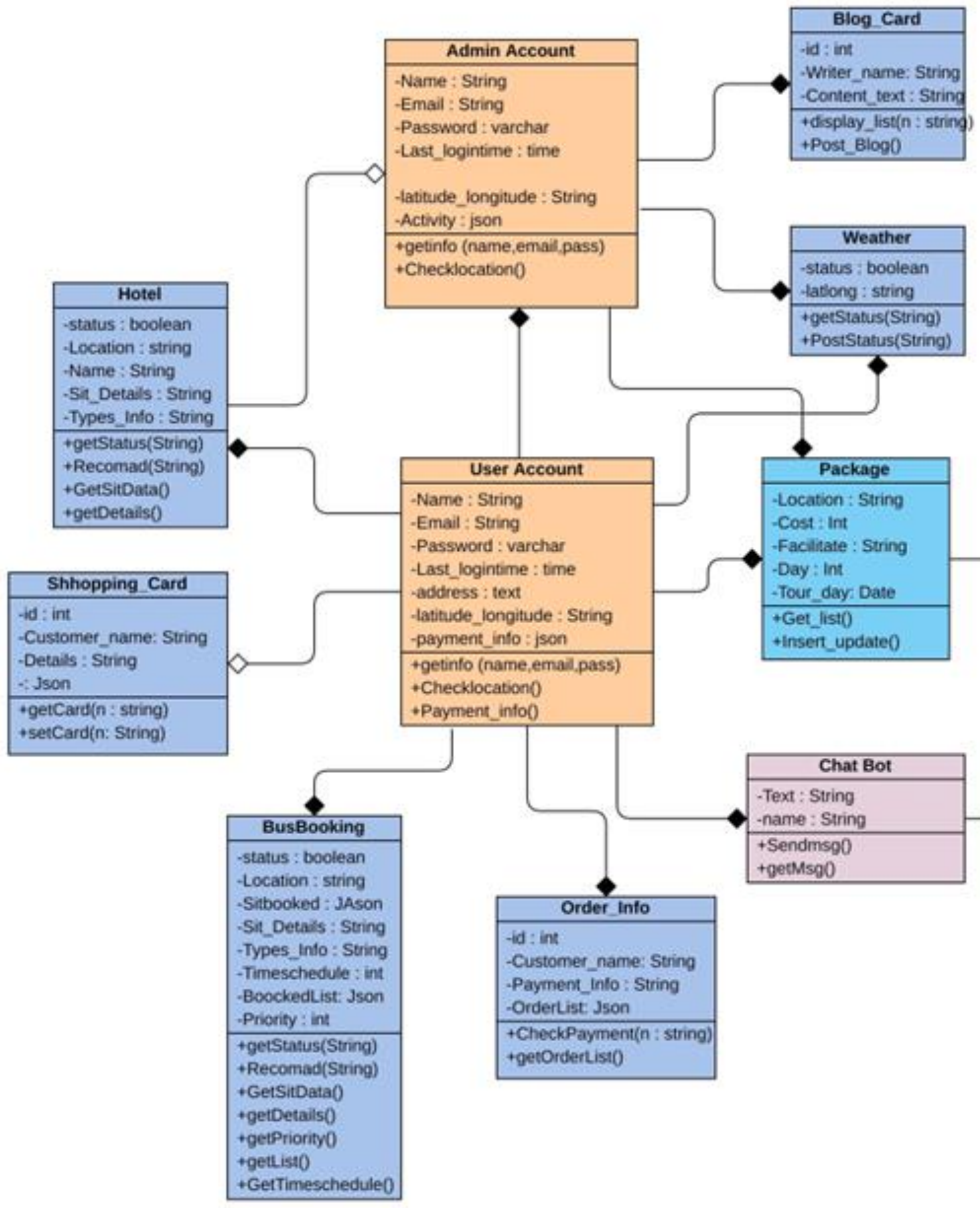
Instance & Inheritance:

A unique object formed from a certain class is referred to as an instance.

Inheritance is a technique by which one class obtains a property from another. A child, for example, inherits the attributes/characteristics of his or her parents.

Classes are used to generate and manage new objects, as well as to provide inheritance—a major component of object-oriented programming and a method for code reuse

We may utilize the existing class's properties and functions by using inheritance. As a result, inheritance improves reusability. An inheritance connection, from the domain's perspective, arranges classes into hierarchical structures. This enables us to model a term hierarchy using generalizations and specializations.



References:

- https://www.tutorialspoint.com/uml/uml_class_diagram.htm#:~:text=Class%20diagram%20is,a%20structural%20diagram.
- https://brilliant.org/wiki/classes-oop/#:~:text=In%20object%2Doriented,of%20reusing%20code.

SEQUENCE DIAGRAM

Sequence diagrams depict interactions between classes as a series of messages sent over time. They are also known as event diagrams. A sequence diagram is an effective tool for visualizing and validating various runtime scenarios. These can assist forecast how a system will act and identify responsibilities that a class may need to have while modeling a new system.

It illustrates the explicit sequence of messages sent between objects during a specific interaction. Because sequence diagrams highlight the time-based ordering of activity among a group of objects, they are extremely useful for comprehending real-time specifications and complicated use cases.

Elements of a Sequence Diagram:

- Actor
- Object
- Lifeline
- Focus of Control
- Message
- Object Destruction

An actor:

- Is a person or system that derives benefit from and is external to the system.
- Participates in a sequence by sending and/or receiving messages.
- Is placed across the top of the diagram.

An object:

- Participates in a sequence by sending, and/or receiving messages.
- Is placed across the top of the diagram.

A lifeline:

- Denotes the life of an object during a sequence.
- Contains an X at the point at which the class no longer interacts.

A focus of control:

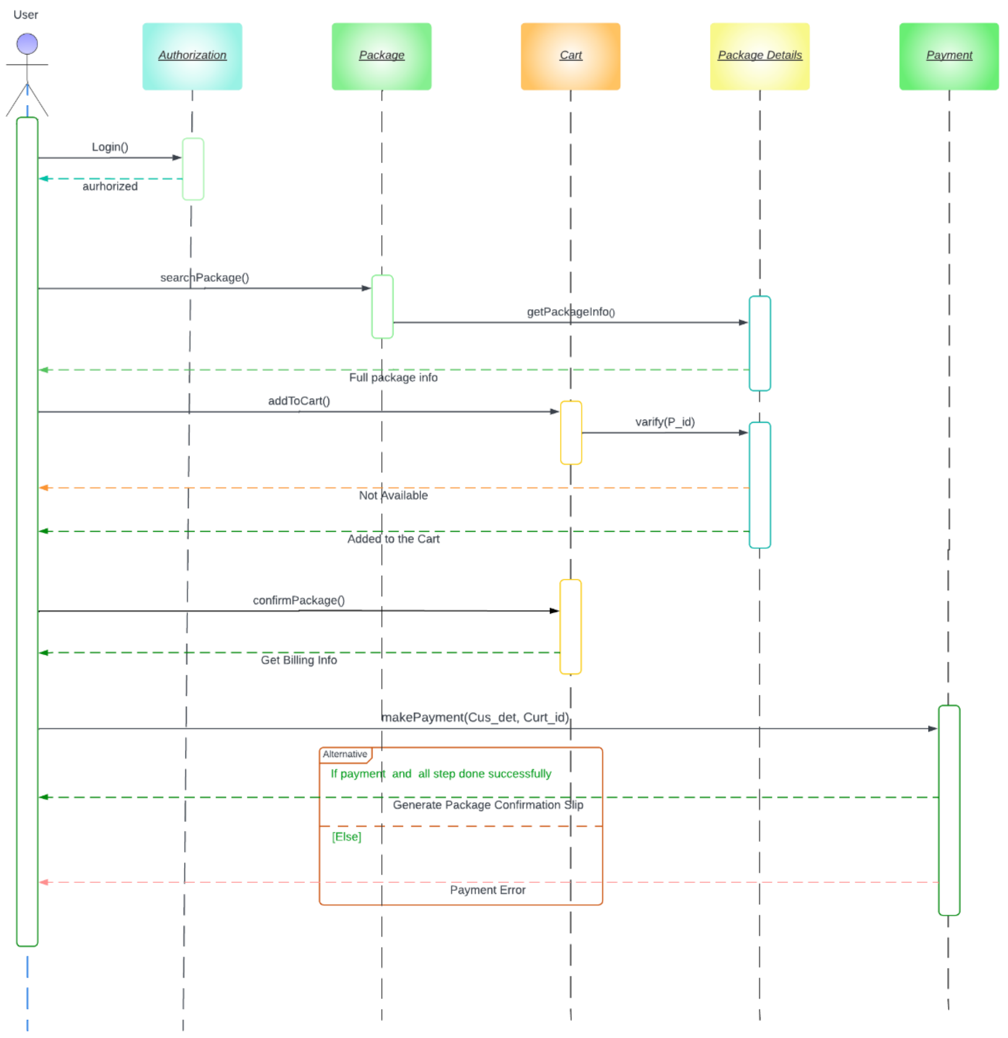
- Is a long narrow rectangle placed atop a lifeline.
- Denotes when an object is sending or receiving messages.

A message:

- Conveys information from one object to another one.
- Object destruction:
- An X is placed at the end of an object's lifeline to show that it is going out of existence.

References:

- https://www.smartdraw.com/sequence-diagram/
- System Analysis and Design 5th Edition Alan Dennis, Barbara Haley Wixom, Roberta M. Roth



<u>Sequence diagrams for package booking</u>

STATE DIAGRAM

Specifically, a state diagram describes the behavior of a single object in response to a series of events in a system. Sometimes it's also known as a Harel state chart or a state machine diagram. This UML diagram models the dynamic flow of control from state to state of a particular object within a system. The behavioral state machine diagram shows the different states that a single instance of a class passes through during its life in response to events, along with responses and actions.

State:

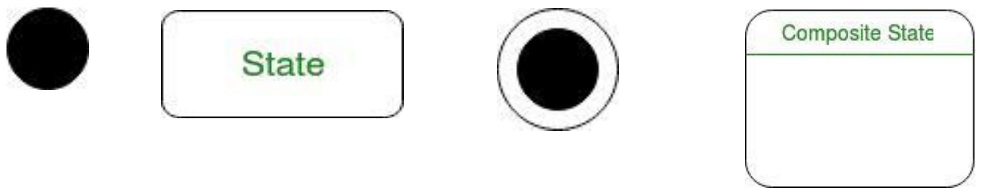
A state is a set of values that describes an object at a specific point in time, and it represents a point in an object's life in which it satisfies some condition, performs some action, or waits for something to happen.

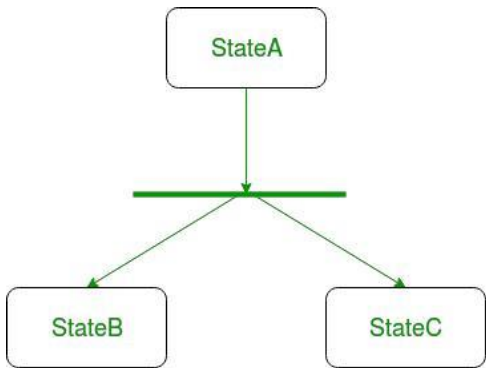
Event:

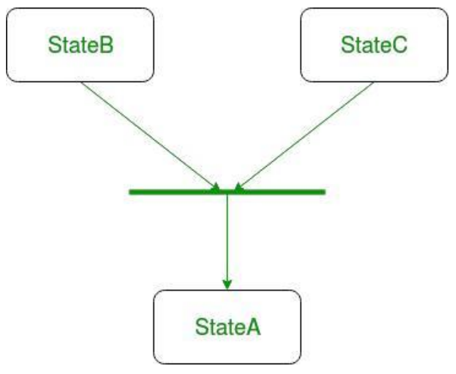
An event is something that takes place at a certain point in time and changes a value(s) that describes an object, which in turn changes the object's state.

Basic Components of State Diagram:

- **Initial state –** We use a black filled circle represent the initial state of a System or a class.
- **Transition –** We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.
- **State –** We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.
- **Fork –** We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent state and outgoing arrows towards the newly created states. We use the fork notation to represent a state splitting into two or more concurrent states.





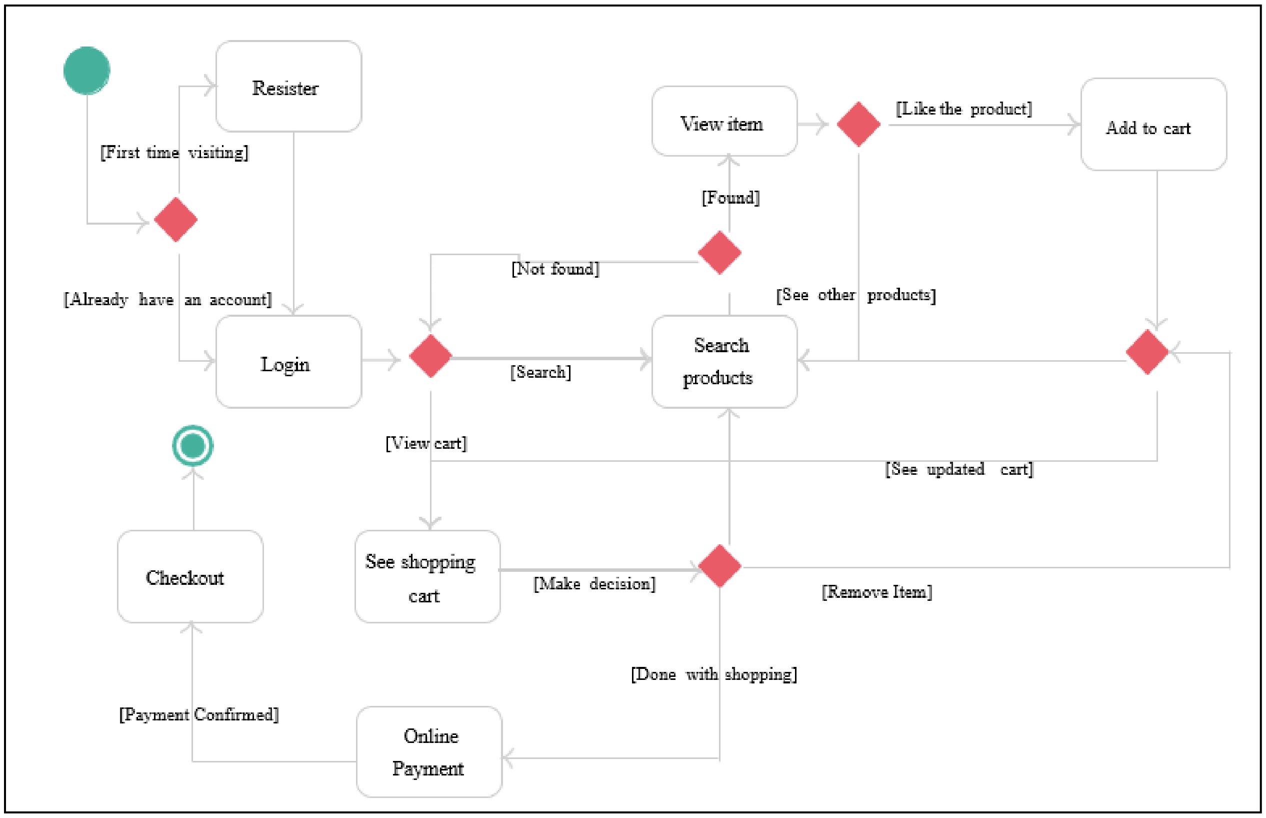


Steps to draw a state diagram –

1. Identify the initial state and the final terminating states.
2. Identify the possible states in which the object can exist (boundary values corresponding to different attributes guide us in identifying different states).
3. Label the events which trigger these transitions.

References:

- https://www.smartdraw.com/state-diagram/#:~:text=Specifically%20a%20state%20diagram%20describes,particular%20object%20 within%20a%20system.
- https://www.geeksforgeeks.org/unified-modeling-language-uml-state-diagrams/
- System Analysis and Design 5th Edition Alan Dennis, Barbara Haley Wixom, Roberta M. Roth
- https://en.wikipedia.org/wiki/State_diagram



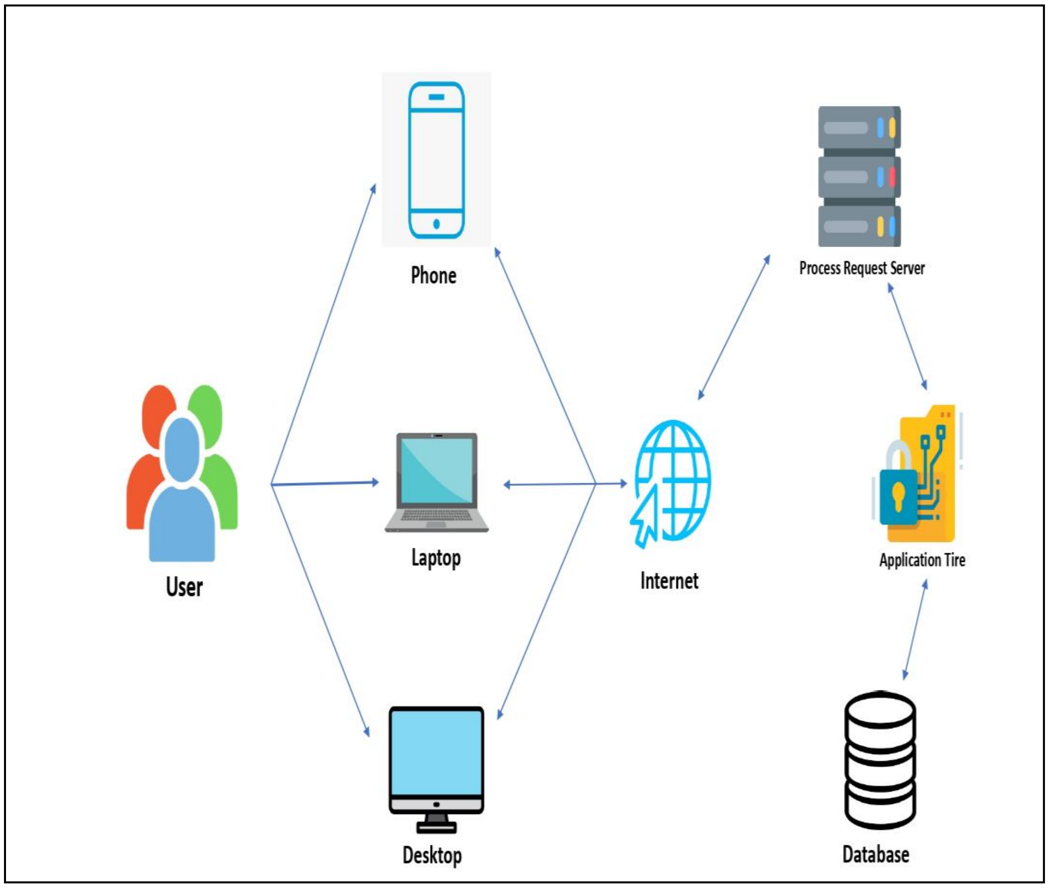
DEPLOYMENT DIAGRAM

What is a Deployment Diagram?

A UML deployment diagram is a diagram that shows the configuration of run-time processing nodes and the components that live on them. Deployment diagrams are a kind of structured diagrams used in modeling the physical aspects of a system or a platform. It shows how the end-users are accessing the system and how the system is serving the service. It only covers the basic structured view of the system without going into details.

Where and when to use Deployment diagram:

- When we need to show, with what the newly added system will interact or integrate with.
- How large or vast the system needs to be.
- Who/what will interact with the system.
- What kind of protocols does the system need to go through.
- What kind of HW/SW does the system need to function properly.
- Who/What can access which part and how'd they access it.



References:

- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/https://en.wikipedia.org/wiki/Deployment_diagram
- Deployment Diagrams - CS dept of Loyla Marymount University https://cs.lmu.edu/~ray/notes/deploymentdiagrams/

CRC Diagram

CRC Diagram – Class Responsibility Collaboration Diagram It represents each classes responsibilities and with which class it collaborates to complete it's responsibilities.

Each CRC card has three components: 1. On top of the card, the class name 2. On the left, the responsibilities of the class 3. On the right, collaborators (other classes) with which this class interacts to fulfill its responsibilities.

Shopping_Card	
• View card • Remove item • Add item	- User - Package_info - Package_Details

Package	
• Add to card • View details	- Bus_Booking - Hotel_Booking - Package_Details - Weather_Info

Blog_Card	
• Tourist spot info • Weather info • Available package	- Weather_Detail - Package - Tourist_Spot

Review	
• View Review • User Feedback • Package Review	- User - Package

Use Case Diagram

Use Case Diagram:

• It only summarizes some of the relationships between use cases, actors, and systems.

• It does not show the order in which steps are performed to achieve the goals of each use case.

• A use case diagram should be simple and contains only a few shapes.

Use Case: According to Wikipedia, In software and systems engineering, a use case is a list of actions or event steps typically defining the interactions between a role and a system to achieve a goal. The actor can be a human or other external system.

Actor:

• Actors may represent roles played by human users, external hardware, or other systems. Actors do not necessarily represent specific physical entities but merely particular facets of some entities that are relevant to the specification of its associated use cases.

• An Actor models a type of role played by an entity that interacts with the subject, but which is external to the subject.

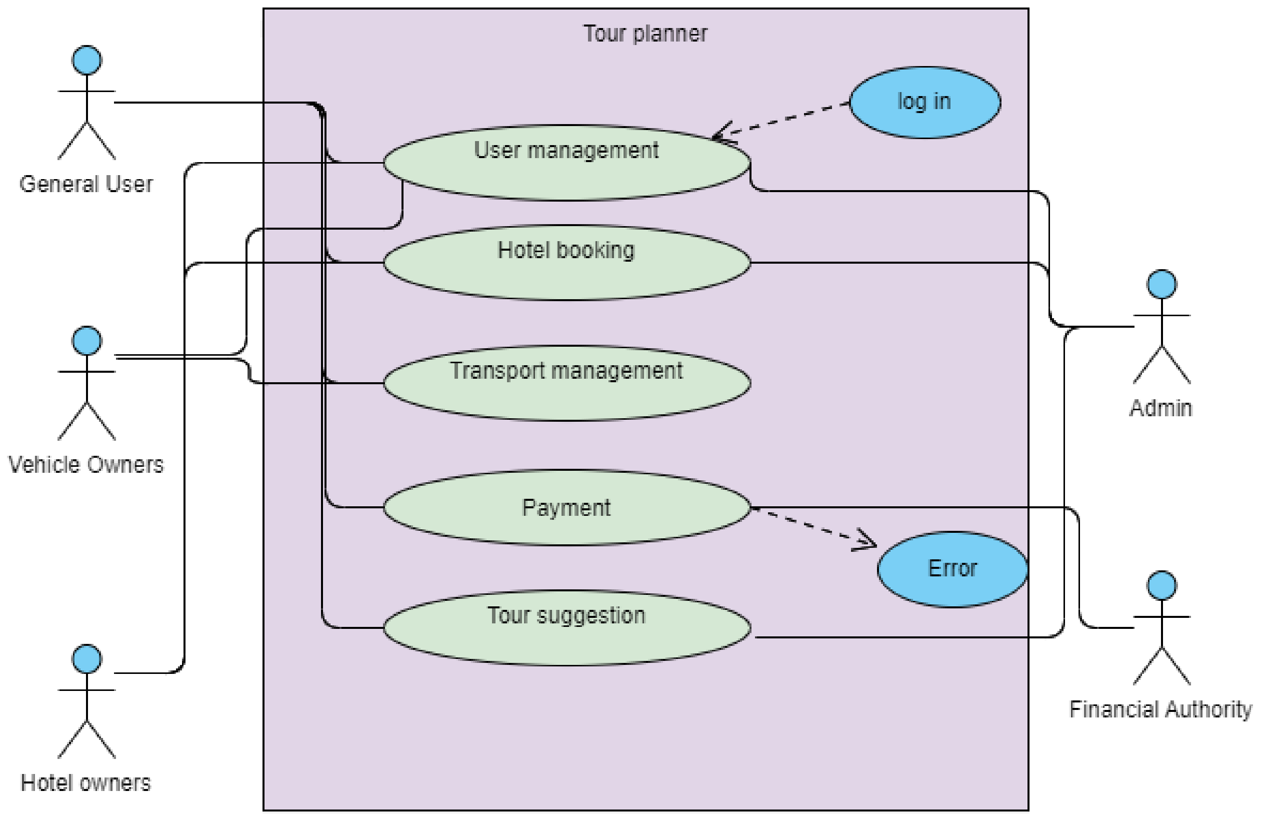
System Boundary:

• A system boundary is a rectangle that you can draw in a use-case diagram to separate the use cases that are internal to a system from the actors that are external to the system. A system boundary is an optional visual aid in the diagram; it does not add semantic value to the model.

References:

• "UML 2 Specification". Retrieved July 4, 2012.

• System Analysis and Design 5 th Edition Alan Dennis, Barbara Haley Wixom, Roberta M. Roth


Tour planner
log in
User management
Hotel booking
Transport management
Payment
Error
Tour suggestion
General User
Vehicle Owners
Hotel owners
Admin
Financial Authority

Contribution of each member

Name	ID	Contribution
Hasibul Hasan Rupok	011201002	State diagram, CRC diagram, report writing
Mahmudul Hasan Bipul	011201025	Sequence Diagram
Sanjida Jannat Anannaya	011201440	Deployment diagram
Muntasir Jahid Ayan	011201439	Sequence Diagram
Shah Newaz Aziz	011201437	Class diagram