Overview of IT Industry:
Module : 1

1. What is Software? What is Software Engineering?

- **Software:**

  ✓ Software is a collection of instructions, data, or programs used to operate computers and execute specific tasks.

  ✓ It can be categorized into system software (which manages hardware and the system), and application software (which helps users perform tasks). Unlike hardware, software is intangible and is developed by writing code using various programming languages.

- **Software Engineering:**

  ✓ Software engineering is the systematic application of engineering principles to the development, operation, and maintenance of software.

  ✓ It involves designing, developing, testing, and maintaining software in a way that ensures reliability, efficiency, scalability, and maintainability.

  ✓ The goal is to create high-quality software within budget and time constraints.

2. Explain types of software:

There are mainly two broad categories of software:

1. **System Software:**

- **Definition:**

  ✓ System software is a type of computer program that is designed to run a computer's hardware and application programs. If we think of the computer system as a layered model, the system software is the interface between the hardware and user applications. The operating system is the best-known example of system software. The OS manages all the other programs in a computer.

  ✓ In short, system software helps run the computer hardware and system.

  ✓ **Examples:**

    ▪ **Operating Systems (OS):** e.g., Windows, Linux, macOS

    ▪ **Device Drivers:** Enable communication between hardware and the OS

    ▪ **Utility Software:** Disk management, antivirus, etc.

2. **Application Software:**

- **Definition:**

- ✓ Application software is a type of computer program that performs a specific personal, educational, and business function. Each application is designed to assist end-users in accomplishing a variety of tasks, which may be related to productivity, creativity, or communication.
- ✓ In short, application software is used to perform specific tasks for the user.
- ✓ **Examples:**
    - ▪ **Word Processing Software:** MS Word, Google Docs
    - ▪ **Web Browsers:** Google Chrome, Firefox
    - ▪ **Multimedia Software:** Adobe Photoshop, VLC Media Player

Other classifications of software include:

- **Programming Software:**
    - ✓ Tools to assist developers in writing and testing programs, e.g., compilers, debuggers, and text editors.
- **Middleware:**
    - ✓ Software that connects different applications or services, facilitating communication and data management.

3. What is SDLC (Software Development Life Cycle)?

- ✓ SDLC is a structured process used by software developers to design, develop, and test software. It ensures the software meets the needs of users and operates efficiently. SDLC consists of several phases, each of which plays a critical role in creating a high-quality software product.

Phases of SDLC:

1. **Requirement Analysis:**
    - ✓ In this phase, the requirements of the software are gathered from client (users, customers, etc.).
    - ✓ Understand what the end-user needs from the software.
    - ✓ Deliverable: Software Requirement Specification (SRS) document.
2. **Planning:**

- ✓ The project is planned, and resources (time, budget, team, tools) are allocated.
- ✓ Risk management and feasibility studies are also performed.
- ✓ Deliverable: Project plan, risk analysis report.

3. **Design:**

- ✓ High-level design (architecture) and low-level design (detailed design) are created.
- ✓ This phase involves designing how the system will be structured and how different components will interact.
- ✓ Deliverable: Design documents, database schema, flowcharts.

4. **Development (Coding):**

- ✓ Actual coding of the software begins based on the design documents.
- ✓ Developers work on writing code for different modules of the software.
- ✓ Deliverable: Source code.

5. **Testing:**

- ✓ The software is tested to find defects and ensure it meets the requirements.
- ✓ Different testing methods include unit testing, integration testing, and user acceptance testing.
- ✓ Deliverable: Test reports, bug reports.

6. **Deployment:**

- ✓ The software is released to the user or client for use in the production environment.
- ✓ Deliverable: Deployed software, deployment documentation.

7. **Maintenance:**

- ✓ After deployment, the software might need updates, bug fixes, or new features, which are managed in the maintenance phase.
- ✓ Deliverable: Updated software, issue logs.

4. What is a DFD (Data Flow Diagram)?

- ✓ A **Data Flow Diagram (DFD)** is a graphical representation of the flow of data through a system.
- ✓ It is used to visualize how data is processed by a system in terms of inputs and outputs. DFDs help in understanding the functions of a system and how data moves between different entities, processes, and data stores.

- **Key components of DFD:**

1. **External Entities:** Represent the sources or destinations of data (e.g., customers, suppliers).
2. **Processes:** Show what happens to the data as it moves through the system (e.g., processing orders, user authentication).
3. **Data Stores:** Represent where data is stored (e.g., databases).
4. **Data Flows:** Indicate the direction of data movement between entities, processes, and data stores.

- **Types of DFD:**

1. **Context-Level DFD (Level 0 DFD):**
    - **Definition:** It is the simplest, high-level diagram that provides an overview of the entire system.
    - **Purpose:** Shows the system as a single process with interactions from external entities (e.g., users, customers, or external systems).
    - **Example:** A "Customer" interacts with an "Online Shopping System" to place an order, and the system responds with confirmation.

Key Feature: No detailed breakdown of internal processes.

2. **Level 1 DFD:**
    - **Definition:** This diagram breaks down the main process from the Level 0 DFD into sub-processes to provide more detail.
    - **Purpose:** Helps to understand how data flows between different processes within the system.

- **Example:** In an "Online Shopping System," Level 1 DFD might show sub-processes like "Search Products," "Place Order," and "Process Payment."

Key Feature: Detailed view of how the system works internally.

3. **Level 2 (and more detailed levels):**

    - **Definition:** These diagrams further break down sub-processes from Level 1 into even more detailed steps.
    - **Purpose:** Provide in-depth understanding of complex processes within the system.
    - **Example:** "Process Payment" could be broken down into "Verify Payment" and "Process Transaction."

- DFD for flipcart :

1. Level 0 (Context-Level DFD):

This shows a high-level overview of the system.

 External Entity:

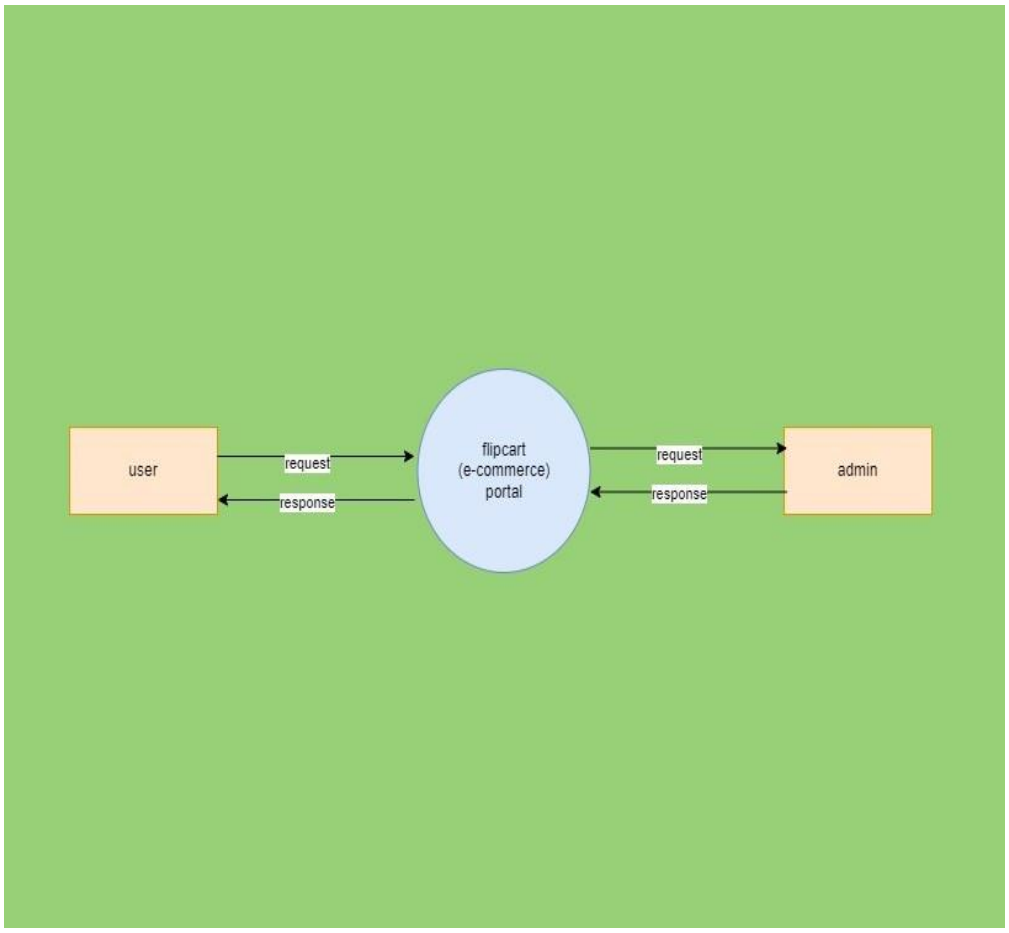
- **User**
- **admin**

 Processes:

- **user**
    - login/register
- **admin**
    - login

 Data Flows:

- user → login/register
- admin → login
- Flipkart → for the login check the name and password. If Any new user do the registration so portal save the data.



2. Level 1 DFD:

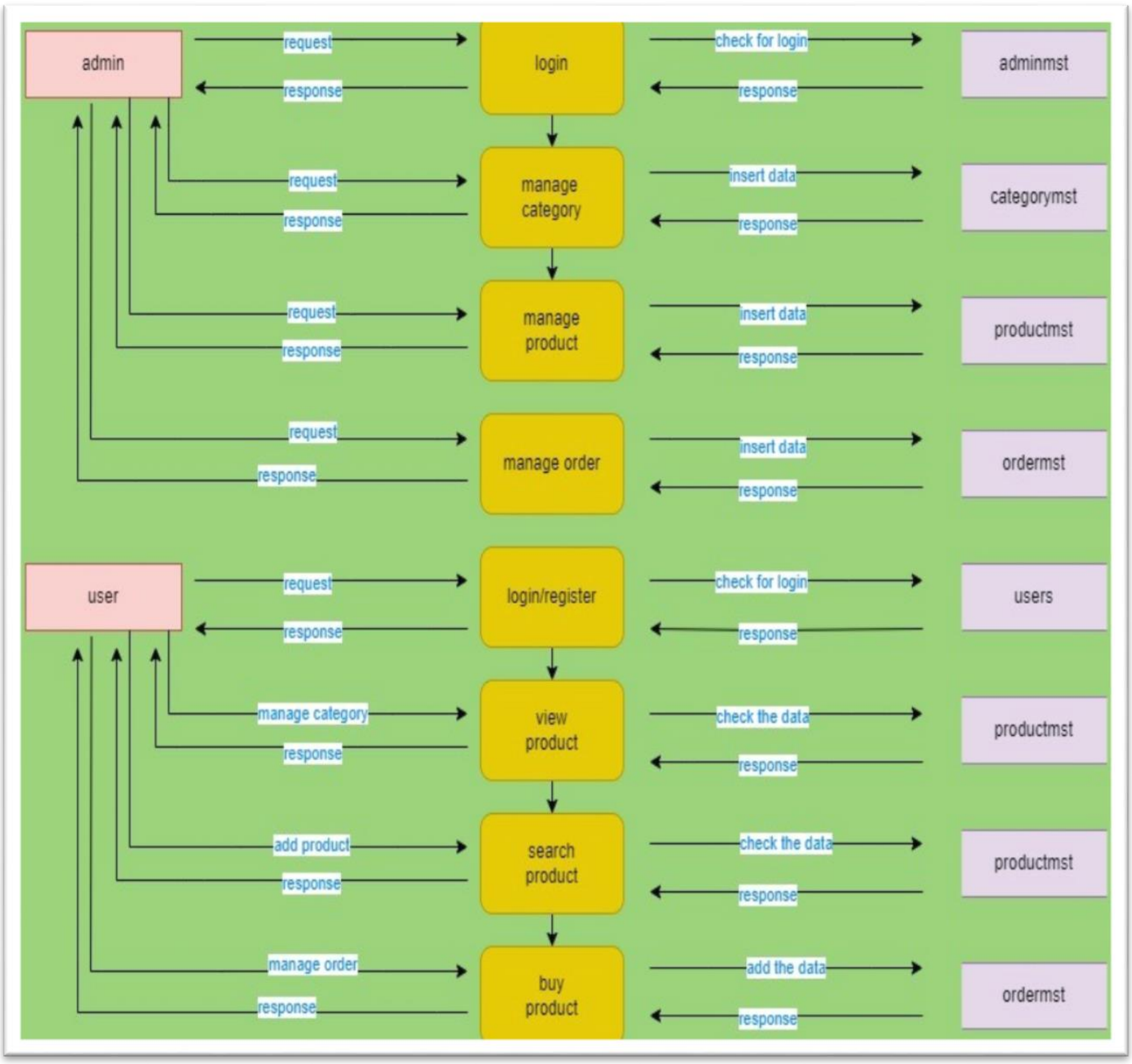
This breaks down the main processes into sub-processes.

- **Processes:**
    - Admin:
        - login
        - **manage Products**
        - **manage category**
        - **manage order**
    - user:
        - login/register

        - view product
        - search product
        - buy product

- **Data Stores:**
    - Admin
        - Category database (for add the product category into database).
        - Product database (for add the product into database)
        - Order database(for manage order status)
    - User:
        - User database (for login check the data into database/for register to store the data into database given by user).
        - order database (product details stores when user buy the product).
        - Product database (any user search the product database given the information).

- **Data Flows:**
    - Admin:
        - changes→ category database
        - changes→ product database
        - manage→ order database
    - User:
        - Login/register → user database
        - View product → product database
        - Search Products → Product Database
        - Buy product → order database


admin
request
response
login
check for login
response
adminmst
request
response
manage category
insert data
response
categorymst
request
response
manage product
insert data
response
productmst
request
response
manage order
insert data
response
ordermst
user
request
response
login/register
check for login
response
users
manage category
response
view product
check the data
response
productmst
add product
response
search product
check the data
response
productmst
manage order
response
buy product
add the data
response
ordermst

3.Level 2 DFD (Detailed Breakdown of Processes)

Level 2 further breaks down individual processes from **Level 1** for more data.

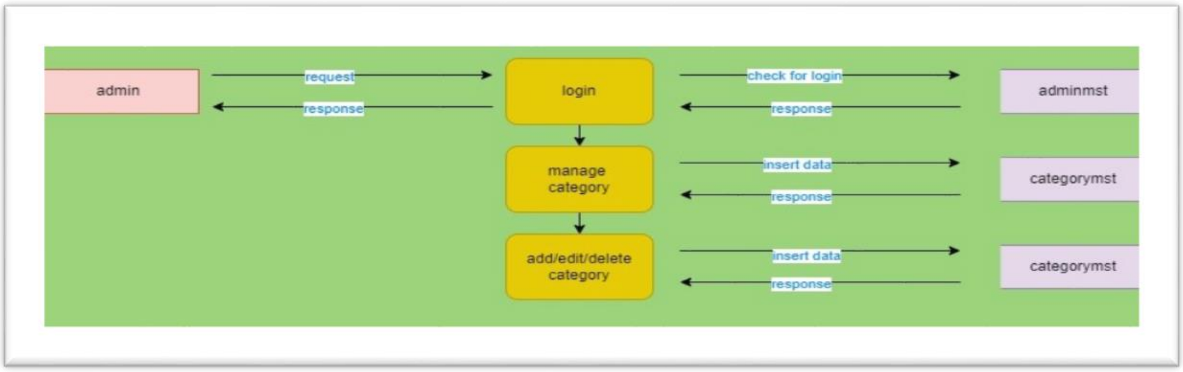
- **Processes:**
  - Admin:
    - **Manage Products:**
      - manage product>add/update/delete the product
    - **Manage categories:**
      - Manage category > add/update/delete the category
    - **Manage orders**:
      - Manage order > view order > confirm/cancel order > dispatch order
  - user:

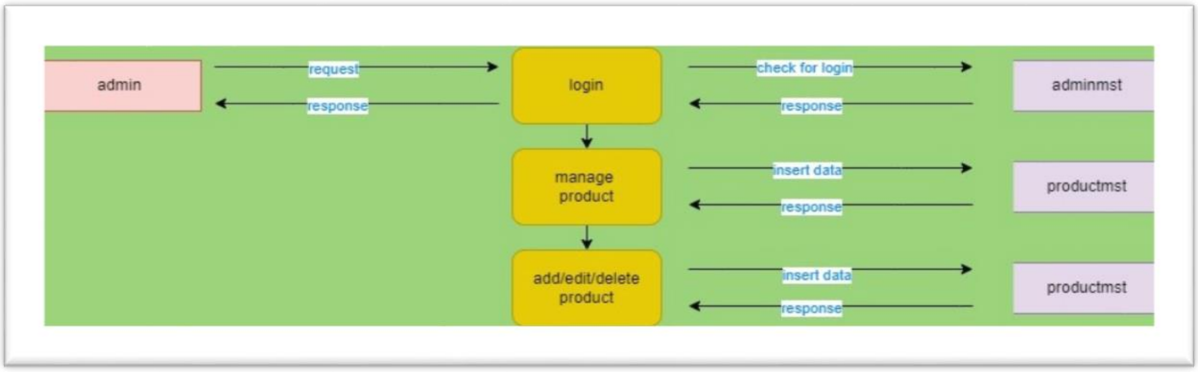
    login/register > view the product > search product > buy product
- **Data Stores:**
  - Admin
    - Category database (for add the product category into database).
    - Product database (for add the product into database)
    - Order database(for manage order status)
  - User:
    - User database (for login check the data into database/for register to store the data into database given by user).
    - order database (product details stores when user buy the product).
    - Product database (any user search the product database given the information).
- **Data Flows:**
  - Admin:
    - Admin login → Add/update/delete category → category database

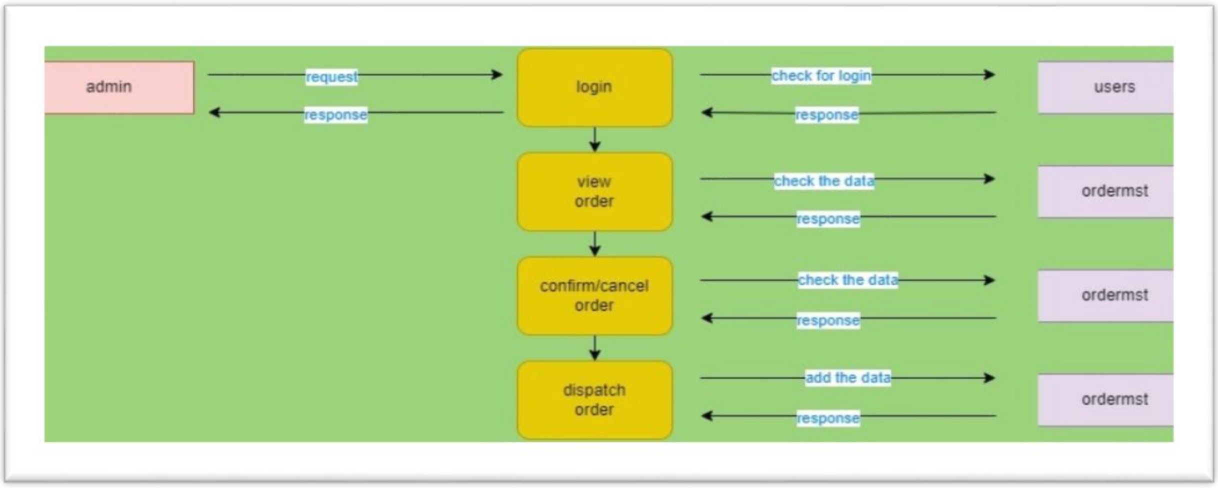
- Admin login → Add/update/delete product → product database
- Admin login → Manage order → view order → confirm/cancel order → dispatch order→ order database

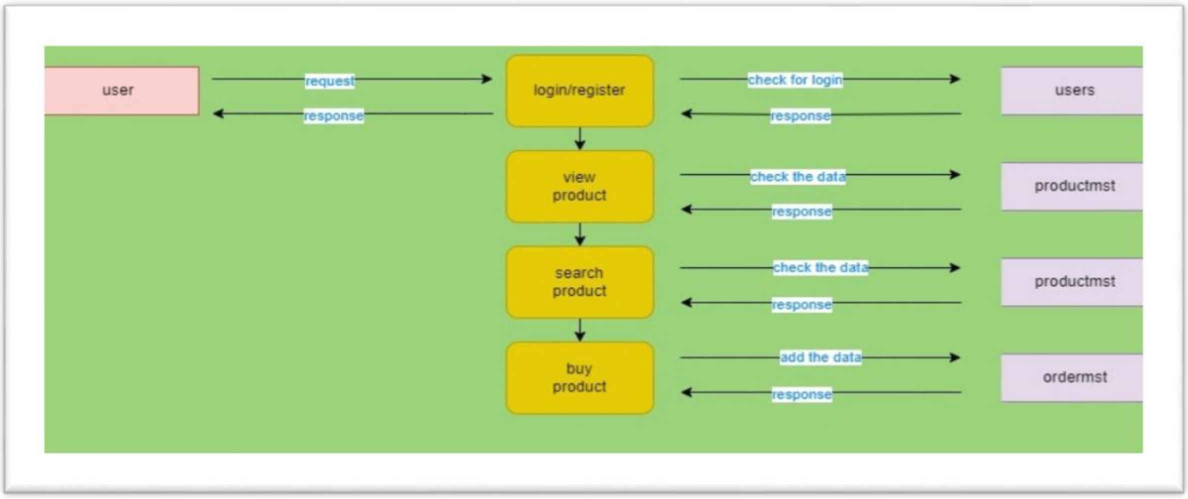
➢ User:

- Login/register → view product → search product → buy product







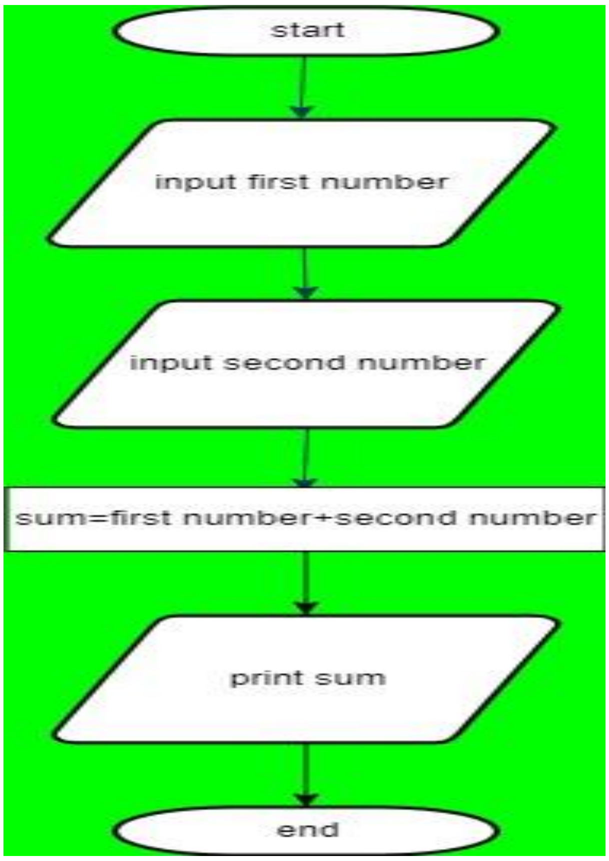


4. What is Flow chart? Create a flowchart to make addition of two numbers.

- A Flowchart is a diagram that visually represents the sequence of steps to solve a problem or perform a process.

- It uses symbols to represent different types of actions or steps, connected by arrows to show the flow of control.

- Common symbols include:
  - **Oval**: Start or End of the process.
  - **Rectangle**: A process or task to be performed.
  - **Diamond**: A decision point (Yes/No or True/False).
  - **Parallelogram**: Input or Output operations.



Here's the logic for a flowchart that performs the addition of two numbers:

1. **Start**: The process begins.
2. **Input First Number**: Get the first number from the user.
3. **Input Second Number**: Get the second number from the user.
4. **Add the Two Numbers**: Perform the addition of the two numbers.
5. **Display the Result**: Output the sum of the two numbers.
6. **End**: The process ends

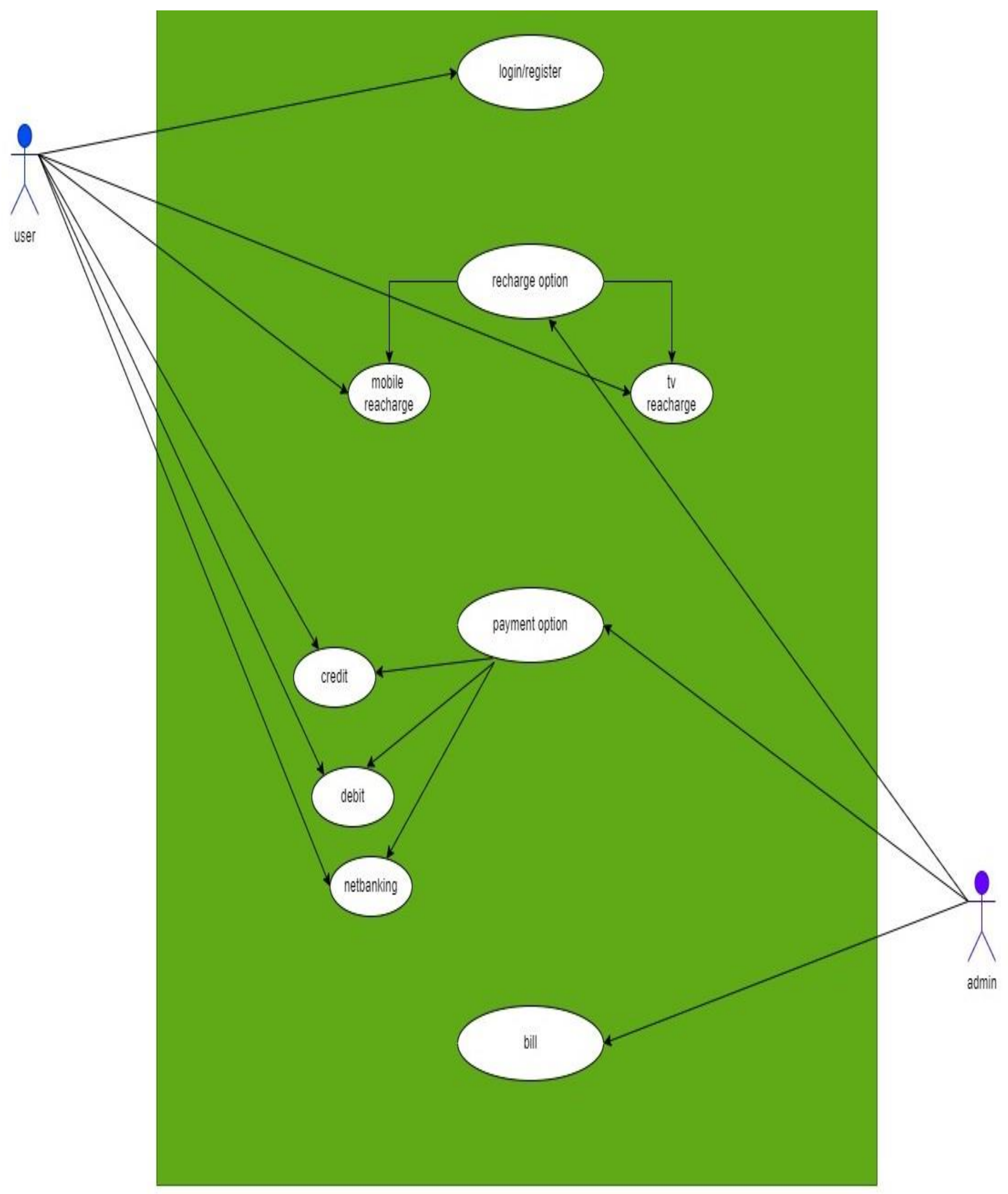
5. What is a Use Case Diagram?

- ✓ A **Use Case Diagram** is part of the **Unified Modeling Language (UML)** and is used to represent the interactions between users (actors) and a system. It describes **what** the system does, without explaining **how** it does it, focusing on the user's perspective and their interactions with the system.

- **Components of a Use Case Diagram:**

1. **Actors**: Represent entities (people, organizations, or systems) that interact with the system. There are two types:
    - **Primary Actor**: The user or entity that initiates an interaction with the system.
    - **Secondary Actor**: Systems or entities that provide support to the main functionality.
2. **Use Cases**: Represent the specific actions or functions that the system performs in response to a request from the actor. These are usually written as simple tasks or goals.
3. **System Boundary**: Defines the scope of the system, showing what is inside the system (use cases) and what lies outside (actors).
4. **Relationships**:
    - **Association**: A line connecting an actor to a use case, showing interaction.
    - **Include**: A use case that is always part of another use case (reusable behavior).
    - **Extend**: A use case that may extend the behavior of another use case.
    - **Generalization**: Represents inheritance between use cases or actors.

- Create a use-case on bill payment on paytm:


login/register
user
recharge option
mobile reacharge
tv reacharge
payment option
credit
debit
netbanking
admin
bill