Module-1

Q-1. What is software? What is software engineering?

Ans:

Software is a collection of data, instructions, or programs that are used to operate computers and execute specific tasks. Unlike hardware, which refers to the physical components of a computer, software is intangible and is created through the process of programming. Software can be categorized into several types:

1.System Software: This includes operating systems (like Windows, macOS, Linux), device drivers, and utility programs that manage and support the computer hardware.

2.Application Software: These are programs designed to perform specific tasks for users, such as word processors, web browsers, and games.

3.Middleware: This software acts as a bridge between system software and application software, facilitating communication and data management.

4.Development Software: Tools used by developers to create, debug, maintain, or otherwise support other programs and applications (e.g., compilers, debuggers, and Integrated Development Environments - IDEs).

What is Software Engineering?

Software Engineering is a systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software. It involves applying engineering principles to software development in order to create high-quality software products that are reliable, maintainable, and meet user requirements. Key aspects of software engineering include:

1.Requirements Analysis: Understanding and documenting what the users need from the software.

2.Design: Planning the software architecture and how the software components will interact.

3.Implementation (Coding): Writing the actual code that makes the software function.

4.Testing: Verifying that the software works as intended and finding and fixing bugs.

5.Maintenance: Updating and fixing the software after it has been deployed.

6.Project Management: Planning, scheduling, and managing resources to ensure that the software project is completed on time and within budget.

7.Quality Assurance: Ensuring that the software development process and the final product meet predefined standards and requirements.

Principles of Software Engineering

Software engineering is guided by several principles to ensure the creation of high-quality software:

1.Modularity: Breaking down software into smaller, manageable, and reusable components.

2.Abstraction: Simplifying complex systems by focusing on the essential features and ignoring the unnecessary details.

3.Encapsulation: Hiding the internal workings of a module and exposing only what is necessary for other parts of the software.

4.Separation of Concerns: Dividing a software system into distinct features that overlap in functionality as little as possible.

5.Maintainability: Designing software in a way that makes it easy to update and fix.

6.Scalability: Ensuring the software can handle increased loads by adding resources.

7.Reliability: Building software that operates correctly under specified conditions.

8.Usability: Creating software that is easy for users to understand and interact with.

9.Efficiency: Optimizing software performance in terms of speed, memory usage, and other resources.

Q-2. Explain types of software

Ans:

Software can be broadly categorized into several types based on its purpose and functionality. Here are the main types of software:

1.System Software:

Operating Systems: Manage the hardware and software resources of the computer. Examples include Windows, macOS, Linux, and Android.

Device Drivers: Enable the operating system to communicate with hardware devices like printers, graphics cards, and network adapters.

Utility Programs: Perform maintenance tasks to ensure the smooth operation of the computer, such as antivirus software, disk cleanup tools, and file management utilities.

2.Application Software:

Productivity Software: Includes applications that help users perform specific tasks efficiently, such as word processors (Microsoft Word), spreadsheets (Excel), and presentation software (PowerPoint).

Database Software: Manages and organizes data, enabling users to store, retrieve, and manipulate information. Examples include Oracle, MySQL, and Microsoft Access.

Multimedia Software: Allows users to create, edit, and play audio, video, and graphic files. Examples include Adobe Photoshop, VLC Media Player, and Final Cut Pro.

Web Browsers: Enable users to access and navigate the internet. Popular browsers include Google Chrome, Mozilla Firefox, and Safari.

Communication Software: Facilitates communication between individuals and groups. Examples include email clients (Outlook, Gmail), instant messaging apps (WhatsApp, Slack), and video conferencing tools (Zoom, Microsoft Teams).

Games: Interactive software designed for entertainment. Examples range from simple mobile games like Candy Crush to complex PC and console games like The Witcher 3 and Fortnite.

3.Middleware:

Application Servers: Provide an environment for running specific applications. Examples include Apache Tomcat and Microsoft Internet Information Services (IIS).

Database Middleware: Facilitates communication and data management between databases and applications. Examples include Oracle Fusion Middleware and IBM WebSphere.

Message-Oriented Middleware (MOM): Allows applications to communicate asynchronously. Examples include RabbitMQ and Apache Kafka.

4.Development Software: Integrated Development Environments (IDEs): Provide comprehensive facilities to programmers for software development, including code editors, debuggers, and build automation tools. Examples include Visual Studio, Eclipse, and IntelliJ IDEA.

Version Control Systems: Help manage changes to source code over time, facilitating collaboration among developers. Examples include Git and Subversion (SVN).

Compilers and Interpreters: Translate code written in high-level programming languages into machine code that can be executed by the computer. Examples include GCC (GNU Compiler Collection) and the Python interpreter.

5.Embedded Software:

Firmware: Specialized software programmed directly into hardware devices to control their functions. Examples include the software running on microcontrollers in washing machines, traffic lights, and medical devices.

Real-Time Operating Systems (RTOS): Manage hardware resources in real-time systems where timing is critical. Examples include VxWorks and FreeRTOS.

6.Enterprise Software:

Enterprise Resource Planning (ERP): Integrates core business processes like finance, HR, and supply chain management. Examples include SAP ERP and Oracle ERP Cloud.

Customer Relationship Management (CRM): Manages a company’s interactions with current and potential customers. Examples include Salesforce and Microsoft Dynamics CRM.

Supply Chain Management (SCM): Manages the flow of goods, information, and finances related to a product or service. Examples include SAP SCM and Oracle SCM.

7.Open Source Software:

Characteristics: Software for which the source code is freely available and can be modified and redistributed. Examples include the Linux operating system, the Apache HTTP Server, and the LibreOffice suite.

8.Proprietary Software:

Characteristics: Software that is owned by an individual or company, with restrictions on its use, modification, and distribution. Examples include Microsoft Windows, Adobe Photoshop, and most commercial software products.

Q-3. What is SDLC? Explain each phase of SDLC

Ans:

The Software Development Life Cycle (SDLC) is a structured process used by software engineers and developers to design, develop, test, and deploy high-quality software. It provides a framework for planning, creating, testing, and maintaining software applications. The SDLC is typically divided into several distinct phases, each with its own set of activities and deliverables.

Phases of the SDLC

1.Planning:

>Objective: To determine the scope, objectives, and feasibility of the project.

>Activities:

-Define project goals and objectives.

-Conduct a feasibility study (technical, operational, and financial).

-Identify project risks and develop mitigation strategies.

-Prepare a project plan, including timelines, resources, and budget.

>Deliverables: Project plan, feasibility study report, risk assessment.

2.Requirements Analysis:

>Objective: To gather detailed information about the software requirements.

>Activities:

-Conduct stakeholder interviews and workshops.

-Gather and document functional and non-functional requirements.

-Create use cases and user stories.

-Validate requirements with stakeholders.

>Deliverables: Requirements specification document, use cases, user stories.

3.Design:

>objective: To create a blueprint for the software that will meet the requirements.

>Activities:

-Define the overall system architecture.

-Design system components, modules, and interfaces.

-Create data models and database schemas.

-Develop detailed design specifications.

>Deliverables: System architecture document, design specifications, data models.

3.Implementation (Coding):

>Objective: To translate design specifications into functional code.

>Activities:

- Set up the development environment.
- Write code according to design specifications.
- Implement unit tests and perform code reviews.
- Integrate different modules and components.

>Deliverables: Source code, unit test results, code review reports.

4.Testing:

>Objective: To verify that the software works as intended and is free of defects.

>Activities:

- Develop test plans and test cases.
- Perform various types of testing (unit, integration, system, acceptance).
- Identify and log defects.

- Retest after defects are fixed.

>Deliverables: Test plans, test cases, defect reports, test results.

5.Deployment:

>Objective: To release the software to users and make it operational.

>Activities:

- Prepare deployment plans and user documentation.
- Set up the production environment.
- Install and configure the software.
- Conduct user training and support.

>Deliverables: Deployment plan, user manuals, installed software.

6.Maintenance:

>Objective: To provide ongoing support and enhancements to the software.

>Activities:

- Monitor the software for bugs and issues.
- Provide regular updates and patches.
- Implement new features and enhancements.

- Conduct performance tuning and optimization.

>Deliverables: Updated software, maintenance reports, enhancement documentation.

Q-4 What is DFD? Create a DFD diagram on Flipkart

Ans:

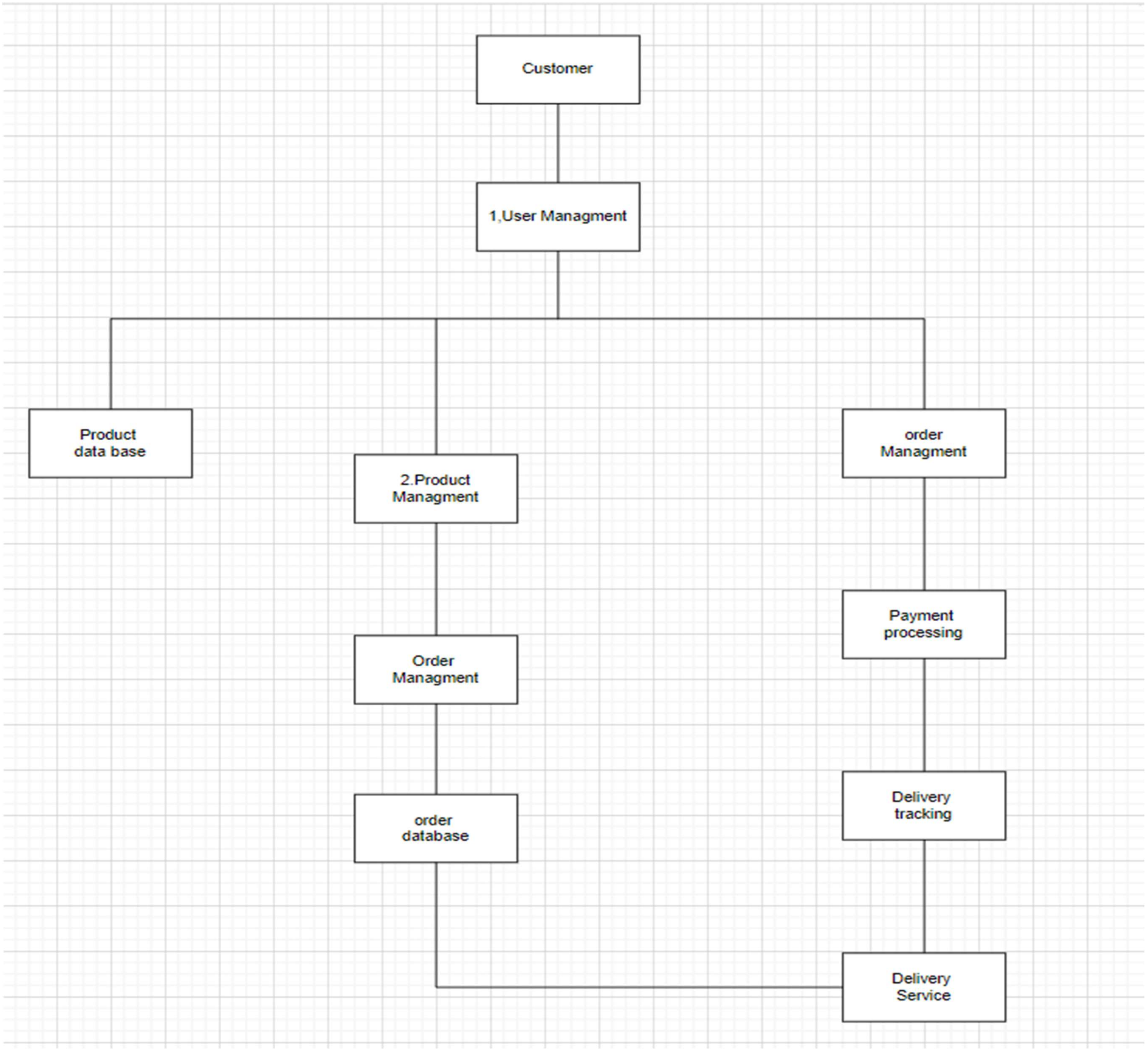
A Data Flow Diagram (DFD) is a graphical representation of the flow of data within a system. It shows how data moves from input to output through various processes and data stores. DFDs help in understanding the functionality of a system and are used for analyzing and designing the system's data flow. Key components of a DFD include:

- Processes: Represented by circles or rounded rectangles, they indicate where data is processed.

- Data Flows: Represented by arrows, they show the direction of data movement between processes,data stores, and external entities.

- Data Stores: Represented by open-ended rectangles, they indicate where data is stored.

- External Entities: Represented by rectangles, they show the sources or destinations of data outside the system.

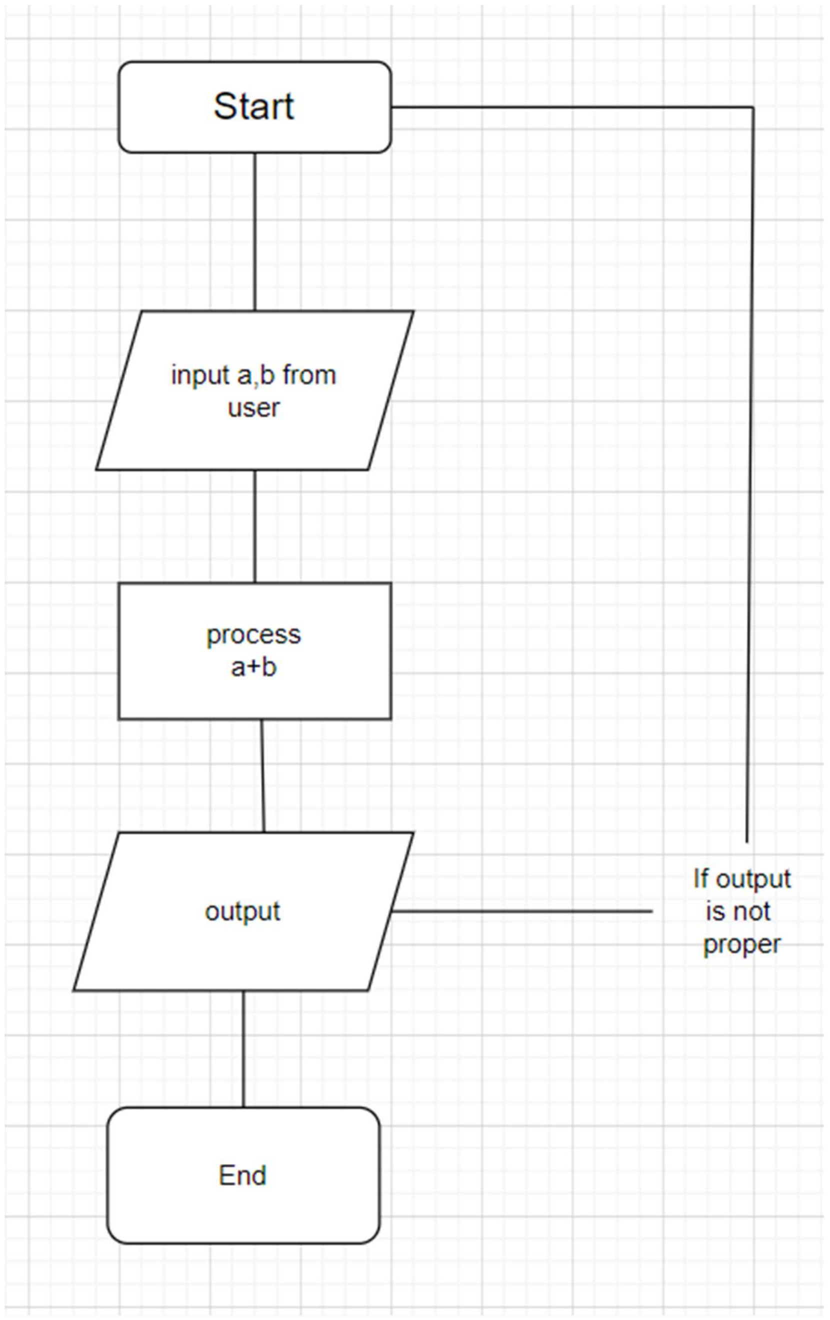

Customer
1,User Managment
Product data base
2.Product Managment
order Managment
Order Managment
Payment processing
order database
Delivery tracking
Delivery Service

Q-5 What is Flow chart? Create a flowchart to make addition of two numbers

Ans

A flowchart is a graphical representation of a process or algorithm, showing the steps as various types of boxes and their order by connecting them with arrows. Flowcharts are used to visualize the sequence of steps and decision points in a process, making it easier to understand, analyze, and communicate how a process works. Common symbols used in flowcharts include:

- Oval: Represents the start or end of a process.
- Rectangle: Represents a process step or an action.
- Diamond: Represents a decision point that can lead to different paths.
- Parallelogram: Represents input or output operations.
- Arrows: Indicate the flow of the process.

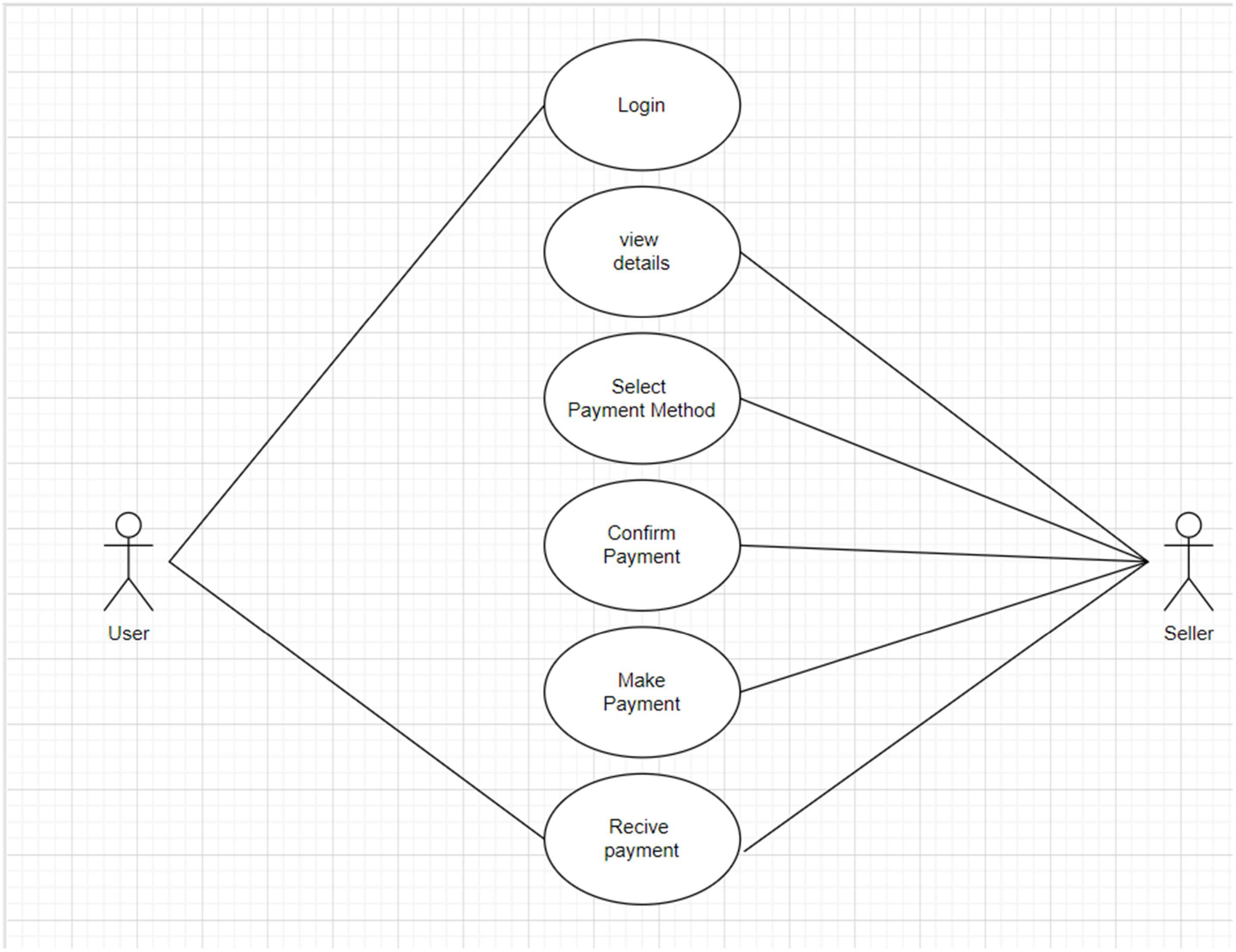

Start
input a,b from user
process a+b
output
If output is not proper
End

Q-6 What is Use case Diagram? Create a use-case on bill payment on paytm.

Ans

A use case diagram is a type of behavioral diagram defined by the Unified Modeling Language (UML) that visually represents the interactions between users (actors) and a system (use cases) to achieve a specific goal. It helps to capture the functional requirements of a system and provides a high-level view of how the system is used by its users. Key elements of a use case diagram include:

- Actors: Represent the users or other systems that interact with the system. They are usually depicted as stick figures.
- Use Cases: Represent the functions or services provided by the system. They are depicted as ovals.
- System Boundary: Represents the boundary of the system, encapsulating all the use cases. It is depicted as a rectangle.
- Relationships: Depict the interactions between actors and use cases, as well as between use cases themselves. Relationships include associations (lines), generalizations, inclusions, and extensions.


Login
view details
Select Payment Method
Confirm Payment
Make Payment
Recive payment
User
Seller