Assignment - 01

WHAT IS SOFTWARE? WHAT IS SOFTWARE ENGINEERING?

1. WHAT IS SOFTWARE?

SOFTWARE: Software is a set of instructions or programs that tells a computer or other electronic device what to do. It includes all the code and data that is required to run an application or system as well as any associated documentation and user interfaces.

Software can be categorised into two main types:
SYSTEM SOFTWARE AND APPLICATION SOFTWARE.
System software includes the operating system, device drivers and other programs that manage and control the hardware resources of a computer system.
Application software on the other hand, includes programs that are designed to perform specific tasks or functions, such as word processing, spreadsheets, multimedia editing and gaming.

Software can be written in a variety of programming languages, including java, c++, python and java script among others. It can be installed on a computer or other electronic device through a variety of methods, such from a physical disc or download from the internet.

Software is a critical component of modern technology, enabling the development of everything from smartphones and gaming consoles to business applications and scientific simulations. The quality and functionality of software is constantly improving, driven by advances in technology, changing user needs, and new software development methodologies and techniques.

2. WHAT IS SOFTWARE ENGINEERING?

Software engineering is the process of designing, developing, testing and maintaining software using a systematic and disciplined approach. It involves applying engineering principles and methods to software development, with the goal of creating high-quality, reliable and efficient software.

SOFTWARE ENGINEERING ENCOMPASSES A RANGE OF ACTIVITIES, INCLUDING: -

1. **REQUIREMENTS GATHERING: -** Defining and Documenting the requirements and specifications for the software applications or system.
2. **DESIGN:** - Creating a detailed plan and architecture for the software application or system, including the user interface, data structures and algorithms.
3. **IMPLEMENTATION:** - Writing, testing and debugging the code for the software application or system.
4. **TESTING:** - Testing the software application or system to ensure that it meets the requirements and specifications, and fixing any defects or errors that are identified.

5. **Maintenance:** Updating and maintaining the software application or system over time to ensure that it continues to meet changing needs of users and the business.

Software engineering also involves collaboration and communication between different stakeholders, including developers, project managers, quality assurance testers and end-users.

Effective software engineering requires a strong understanding of software development principles and best practices, as well as expertise in programming languages, software development tools and project management methodologies.

Assignment – 02

Q2. **Types of software**

SOFTWARE

- **Software: -** Software is a set of instructions, data or Programs used to operate computers and specific tasks. It is opposite of hardware, which describes the physical aspects of computers. Software is a generic term used to refer to applications, scripts and programs that run on a device.

Now that we know the meaning of software we shall discuss "TYPES OF SOFTWARE" following are the types of software :-

- **APPLICATION SOFTWARE**
- **SYSTEM SOFTWARE**
- **DRIVER SOFTWARE**
- **MIDDLLEWARE SOFTWARE**
- **PROGRAMMING SOFTWARE**

Above mentioned are 5 types of software, we will now understand each type of software in detail.

APPLICATION SOFTWARE

Application software is also known as "apps" it is a computer generated programs designed to perform specific tasks or activities for users. Unlike system software, which is essential for the functioning of a computer, application software is optional and serves a specific purpose based on the needs of users.

There are various types of application software, including productivity software such as word processors, spreadsheets and presentation software as well as multimedia software, graphic design software and gaming software.

Here are few popular e.g. of application software:-

- Microsoft Office Suite
- Adobe
- Creative Suite
- Spotify
- Zoom

Application software can be installed on a computer, mobile device or accessed through the internet via a web application. It has greatly increased the efficiency and convenience of tasks such as communication, data analysts and entertainment.

SYSTEM SOFTWARE

System Software is a type of computer program that is designed to manage and control the hardware and software resources of computer system. It is essential for the functioning of a computer and provides the foundation on which application software runs.

System software is typically divided into two categories: -

1. **OPERATING SYSTEM: -** An operating system (OS) is a type of system software that manages computer hardware and software resources and provides common services for computer programs. E.G., Windows, macOS, etc.
2. **UTILITY SOFTWARE: -** utility software is type of system software that provides tools to optimize and maintain the computer system.

System software is responsible for managing computer resources such as memory, storage and processing power. It also provides essential services such as file management, security and network connectivity. Without systems software, application software would not be able to run on a computer.

*** SOME KEY FEATURES OF SYSTEM SOFTWARE INCLUDE: -**

1. **Memory Management: -** System software manages computer memory and allocates memory resources to running programs.
2. **Device Drivers:** - System software includes drivers, which are programs that allows the computer to communicate with hardware devices such as printers, scanners and cameras.
3. **User Interface:** - The system software provides a user interface that allows users to interact with the computer system.
4. **Security:** - System software includes security features such as firewalls, antivirus software and encryption tools to protect the computer system from malware, viruses and other threats.

Overall, system software plays a critical role in the functioning and performance of a computer system, and it is essential for the smooth operation of both systems and application software.

* DRIVER SOFTWARE *

Driver software: - Driver software is a type of program that allows the operating system to communicate with hardware devices such as printers, scanners, keyboards and mice. Without drivers, the operating system would not be able to recognise or use hardware devices.

Driver software is typically provided by the hardware manufacturer and it's specific to a particular device and operating system. When a new driver is installed, the operating system will typically search for and install the appropriate driver software automatically.

Driver software acts a translator between the operating system and hardware device. It provides a standardized interface for the operating system to communicate with the hardware, regardless of the specific hardware device or manufacturer.

Overall, driver software plays a critical role in enabling the operating system to communicate with hardware devices. It ensures that hardware devices are recognised and can be used by the operating system, and it provides a standardized interface for communication between two.

MIDDLEWARE SOFTWARE

Middleware software is a type of computer program that sits between different software applications and facilitates communication between them it acts as a mediator, enabling different software applications to exchange data and interact with each other, even if they are running on different platforms or using different programming language

Middleware software can be thought of as a “glue” that connects different software applications together.

SOME COMMON EXAMPLES OF MIDDLEWARE SOFTWARE INCLUDE: -

- Application software: Application servers provides a platform for running web applications and enable communication between web servers and web database servers.

SOME KEY FEATURES OF MIDDLEWARE SOFTWARE INCLUDE: -

1. Compatibility: Middleware software is designed to be compatible with different software applications, platforms and programming languages.
2. Scalability: It allows it to handle large volumes of data and support a large number of software applications.
3. Security: Middleware software provides security features such as authentication and encryption to ensure the confidentiality and integrity of data exchanged between software applications.
4. Performance: it ensures that data exchange between software applications is fast and efficient.

Overall, Middleware software plays a critical role in enabling different software applications to communicate and exchange data with each other. It provides standardized interface for communication and data exchange, regardless of the specific software applications involved and enables organizations to integrate different software system into a cohesive whole.

PROGRAMMING SOFTWARE

Programming software is a type of computer software that is used by programmers and developers to create, debug, and maintain other software programs and applications. Programming software typically includes tools such as text editors, integrated development environments (IDEs), compilers and debuggers.

Text editors are simple tools that allows programmers to create and edit source code, while IDEs are more comprehensive programming environments that provide integrated tools for coding, testing and debugging. Compilers are programs that translate source code into executable code that can be run on a computer or other device, and debuggers are tools that help programmers identify and fix errors in their code.

Programming software is critical for software development, as it enables programmers and developers to write and test software code efficiently and effectively. The availability of powerful programming software has played a key role in the rapid development of modern computing and technology, enabling developers to create increasingly sophisticated software applications and system.

Assignment - 03

Q1. What is SDLC?

SDLC stands for Software Development Life Cycle. It is a structured approach used to plan, develop, test, and maintain software systems. The SDLC consists of several distinct phases, each with its own objectives, activities, and deliverables. Here is an explanation of each phase of the SDLC:

1. REQUIREMENT GATHERING
2. ANALYSIS
3. DESIGN
4. IMPLEMENTATION
5. TESTING
6. DEPLOYMENT
7. MAINTENANCE

SDLC involves a series of well-defined phases, each with specific objectives and deliverables. The main goal of SDLC is to ensure the successful development of high-quality software that meets the desired requirements and is delivered on time and within budget.

Requirements Gathering: In this phase, project stakeholders, including users, business analysts, and developers, identify and document the software requirements. The goal is to understand the needs, expectations, and constraints of the software system to be developed.

Analysis: In this phase, the gathered requirements are analyzed in detail. The system's functional and non-functional requirements are examined, and any gaps or inconsistencies are addressed. Use cases, data models, and process flows may be created to clarify system behaviour.

Design: The design phase involves creating a blueprint for the software system. The architectural design specifies the system's structure, components, and interfaces. Detailed designs for individual modules or components are developed, including database design, user interface design, and algorithm design.

Implementation: In this phase, the software is developed based on the designs created in the previous phase. The programming code is written and unit tested to ensure it meets the design specifications. The implementation phase involves coding, debugging, and integration of various components.

Testing: The testing phase focuses on validating the software system to ensure it meets the specified requirements. It involves various types of testing, including unit testing, integration testing, system testing, and acceptance testing. Defects are identified, reported, and fixed to improve the quality of the software.

Deployment: Once the software has been thoroughly tested and approved, it is deployed to the production environment. This involves activities such as installation, configuration, data migration, and user training. The software is made available to end-users for their use.

Maintenance: The maintenance phase involves ongoing support and enhancement of the software system. It includes bug fixes, performance optimization, and incorporating new features or functionality based on user feedback and changing requirements.

It's important to note that the SDLC is not a linear process, and iterations or incremental development approaches may be used. Additionally, various methodologies, such as Waterfall, Agile, or DevOps, can be employed to adapt the SDLC to different project needs and organizational preferences.

Assignment – 04

Q1. What is DFD? Create a DFD diagram on flip kart

DFD stands for Data Flow Diagram. It is a graphical representation of the flow of data within a system or process. DFDs are used to visualize how data moves from one component to another in a system, representing the inputs, outputs, processes, and data stores involved.

DFDs consist of several symbols and connectors, including circles, arrows, and rectangles, which represent different components and their relationships. Here are the key elements used in DFDs:

1. Process
2. Data Flow
3. External Entity
4. Data Store

Process: Represented by a circle or oval, a process signifies a transformation or manipulation of data. It represents a specific task or action performed on the data within the system.

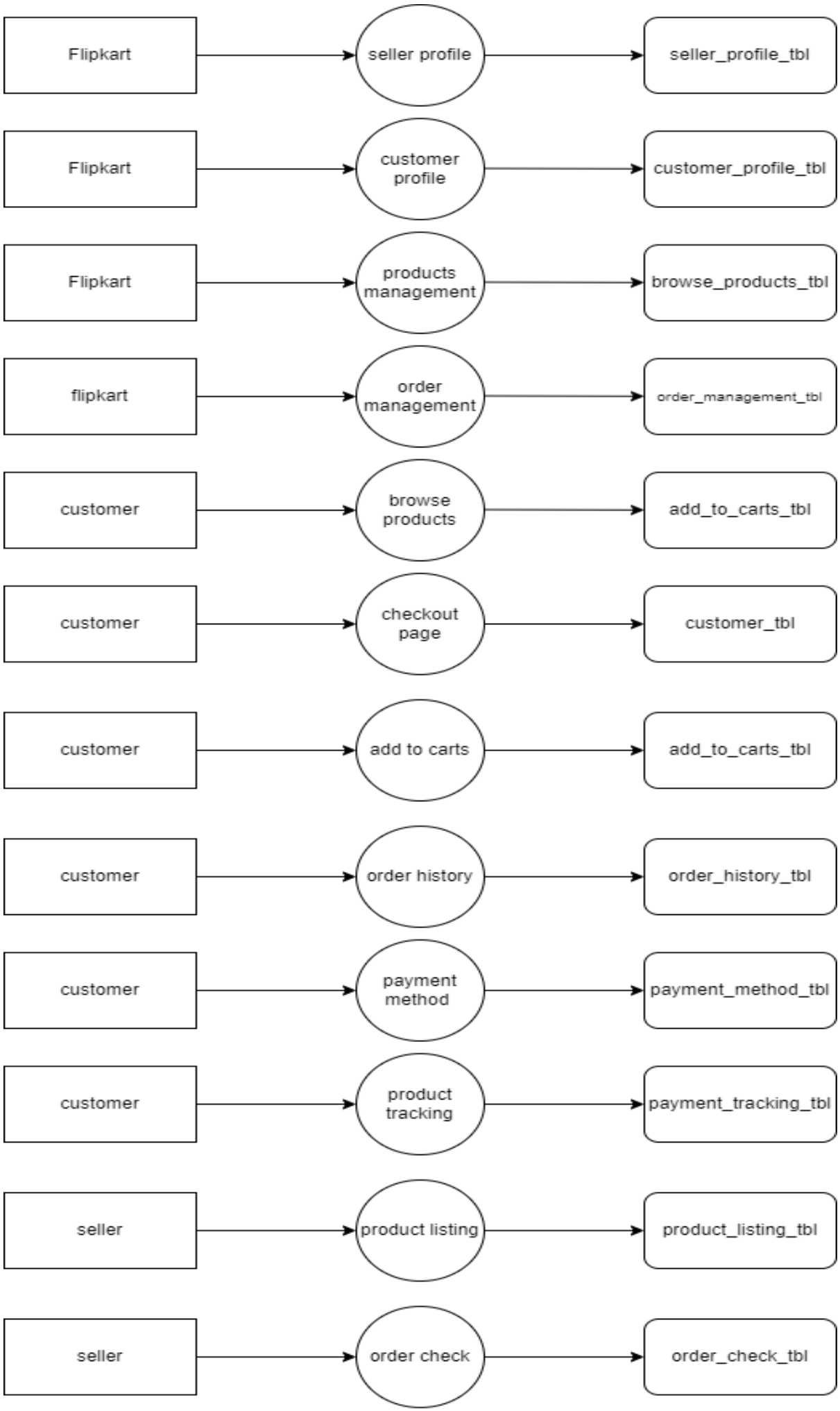
Data Flow: Represented by arrows, data flows show the movement of data between different components of the system. They illustrate the path of data as it is input, processed, and output by the system.

External Entity: Represented by rectangles, external entities represent the sources or destinations of data outside the system being analysed. They can be users, other systems, or organizations that interact with the system.

Data Store: Represented by a rectangle with two parallel lines at the sides, a data store represents a repository where data is stored and retrieved. It can be a database, a file, or any other form of persistent data storage.

DFDs are useful for understanding the data flow and interactions between components in a system. They help in visualizing the system's overall structure, identifying data dependencies, and highlighting areas for improvement or optimization. DFDs are commonly used in software development, system analysis, and business process modelling to represent the flow of data within a system or process.

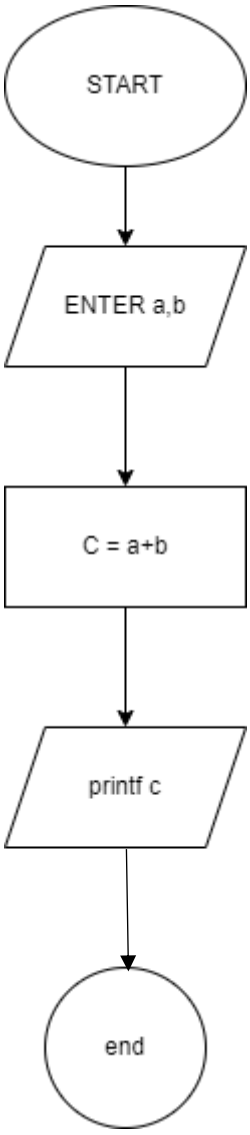
BELOW IS THE DGD DIAGRAM FOR FLIPKART: -


Flipkart
seller profile
seller_profile_tbl
Flipkart
customer profile
customer_profile_tbl
Flipkart
products management
browse_products_tbl
flipkart
order management
order_management_tbl
customer
browse products
add_to_carts_tbl
customer
checkout page
customer_tbl
customer
add to carts
add_to_carts_tbl
customer
order history
order_history_tbl
customer
payment method
payment_method_tbl
customer
product tracking
payment_tracking_tbl
seller
product listing
product_listing_tbl
seller
order check
order_check_tbl

Assignment – 05

Q1 What is Flow chart? Create a flowchart to make addition of two numbers

A flowchart is a graphical representation of a process, algorithm, or system, using various shapes and arrows to depict the flow of control and decision points. It helps visualize the steps involved in a process and provides a clear understanding of the logical flow.

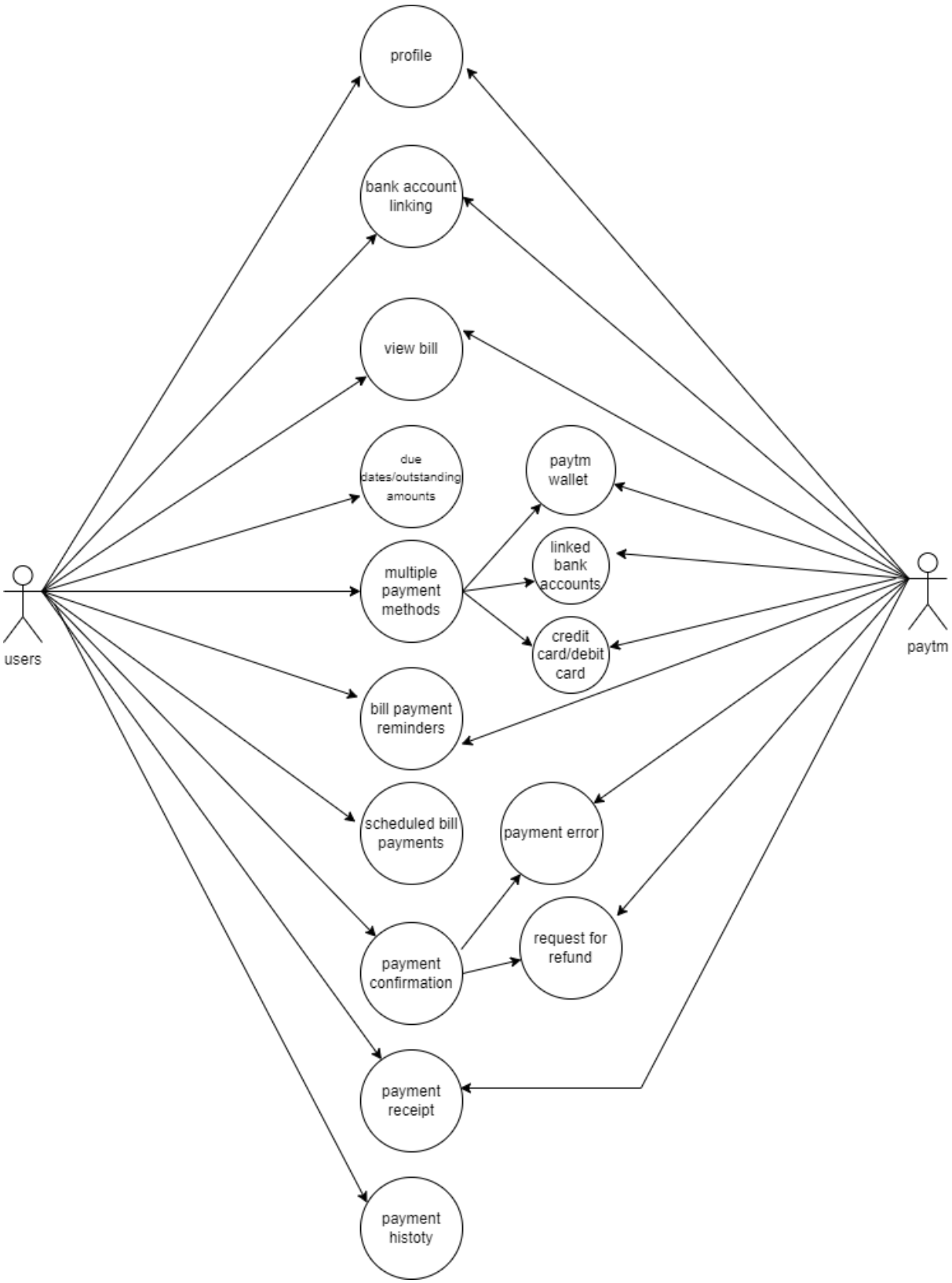


Assignment – 06

Q1. What is Use case Diagram? Create a use-case on bill payment on paytm.

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

A use case diagram is a visual representation of the functional requirements of a system from the perspective of its actors (users or external entities). It helps illustrate the interactions between the actors and the system, showcasing the different use cases or functionalities that the system provides.


profile
bank account linking
view bill
due dates/outstanding amounts
multiple payment methods
bill payment reminders
scheduled bill payments
payment confirmation
payment receipt
payment histoty
paytm wallet
linked bank accounts
credit card/debit card
payment error
request for refund
users
paytm