1. What is software? What is software engineering?

Programs, data and instructions make up what we call software guiding computers on how to carry out tasks. It covers everything from the system that handles hardware resources to applications such, as word processors, web browsers and video games.

Contrastingly software engineering is an approach to creating, running and upkeeping software. It involves applying engineering principles throughout the software development journey. From gathering requirements and designing to testing, launching and maintaining. Software engineers use techniques to ensure that software is dependable, adaptable easy to maintain and meets user and stakeholder needs. This encompasses aspects, like designing software writing code testing it out thoroughly documenting processes accurately managing projects effectively and ensuring quality standards are met.

2. Explain types of software

Software can be categorized into various types, as many classifications are used according to the function, use, license, and method of developing software. undefined

System Software: This category of software comprises operating systems, device drivers, utilities, and other hardware that can interact with computers. They contain examples like Windows, macOS, Linux, and for printer and graphics card drivers.

Programming Software: They are the means for software programming along with debugging and maintenance tasks. There are for example IDEs, like Visual Studio, Eclipse and JetBrains IntelliJ IED as well as text editors and compilers.

Embedded Software: A software that is located at the source within devices that manage their functions. It is commonly employed in consumer electronic, automotive systems, medical devices and industrial tools. Cases here will be of firmware like the one in the routers, automotive engine control systems, and smart home devices.

Enterprise Software: Application of a software concept intended for running organizations and other companies' operations and administration processes. Among them, CRM, ERP, and HRMS are widespread.

Open Source Software: The code is made available through software licenses that permit users to see, edit, and distribute a given code's source at their own discretion. Examples are Linux operation systems, Apache web server, and Mozilla Firefox browsers.

3. What is SDLC? Explain each phase of SDLC

SDLC is the abbr. for Software Development Life Cycle. It is a scrutinized methodology applied by software developers to streamline and systemize the design, development, test, and deployment of software programs. The SDLC is comprised of a number of cycles, each of which has its own tasks and deliverables. The phases might be switched with each other depending on the particular methodology of the procedure, e.g. Waterfall, Agile, or DevOps. undefined

Requirement Gathering and Analysis:

Part of this phase involves the team member in contact with stakeholders to collect and assess the requirements for the software.
Requirements can work in various ways - either functional (what the software should accomplish) or non-functional (like, performance, scalability, and security).
The end product of this phase is Software Requirements Specification (SRS) document that quantifies the software's features, functionalities and constraints.

System Design:

The system architecture and design plans are set up, subsequently developing.
The phase in which high level and low level design documents are made, guided by the system architecture, data model, and interfaces.
Some of the architectural choices can include choosing programming languages, databases, frameworks, and technology stacks.
The realization stage produces a design document in a form of the blueprint which is followed by the developers.

Implementation (Coding):

This phase would be the real place where coding and developing of the software will actually occur.
Developers code according to the design specifications, stick to code standards, and follow prescribed practices by experts.
The implementation phase furnish modules, classes, functions, and other software adaptations.
Version control systems provide means of coping with the changes and coordination among the developers.

Testing:

When the code is ready it enters the testing phase where all possible bugs are identified and the code is modified, as needed.
Testing can embrace a variety of testing techniques, for instance, with unit testing (testing individual components), integration testing (testing the interaction between components), system testing (testing the whole system), and acceptance testing (confirming against user needs).

Test cases are created on the basis of specs and requirements, and automated tools of the trade may prove to be quite helpful in this case.
Testing scope is to make sure that the product is up to the minimum quality level and operates as it should.

Deployment:

Finally, the developed software is subjected to testing and validation before being shipped to the production environment.
These tasks are comprised of installing software on servers, configuring the settings, and validating its compatibility with other systems.
Migration of the data, user training and documentation updates, the process is also possible during the deployment.
Through CI/CD practices, the continuous integration and continuous deployment (CI/CD) process is continuously automated and made smooth.

Maintenance:

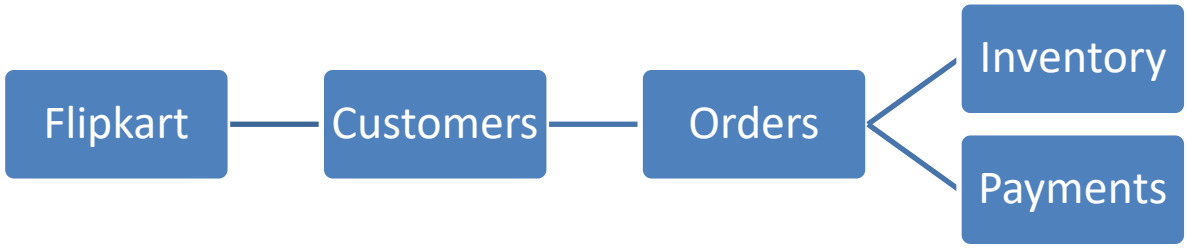
Software entry into operation marks the beginning of the maintenance when it is being monitored, updated and enhanced.
The maintenance can be synonymous with fixing bugs, building new features, code performance optimization, and user feedback.
Updates and patches are published by software providers on a regular basis, in order to patch security vulnerabilities or improve application performance.
It promotes the lasting operation and integrity of the software.
These phases are normally repetitive since their products may be revisited and edited to include refinements, reaction, and feedback. Furthermore, there are a number of SDLC models like Waterfall, Agile or Spiral, depending on which one these phases are given priority and structured differently.

4. What is DFD? Create a DFD diagram on Flipkart

DFD consists of data flow diagram. It is a graphical portrayal of data movement within a system which depicts how data are captured, processed and what results are attained. DFDs are the widely applied in the software engineering to model both the functional and data structure of a system. It is composed by external parties, methods, data structures, and entity data flows.

Development of a DFD for Flipkart, e-commerce system, consists in identification of the essential components and their relations. undefined

In this DFD diagram:



1. **External Entities:**
- **Customers**: Symbolizes website visitors who are searching for the product details from the Flipkart website or placing any order, etc.

2. **Processes:**
- **Orders**: Symbolizes the point of customer input in which they place their orders through Flipkart platform.
- **Inventory**: The function is dedicated for managing the product stock, which allows for adding up the level of products, monitoring the products, etc.
- **Payments**: Functions as the industry of dealing with transactions of payment for the order made by clients.

3. **Data Stores:**
- The diagram is only general and not specific in data stores, but in a detailed DFD a data base could be displayed where the information such as product, orders, list of customers is located.

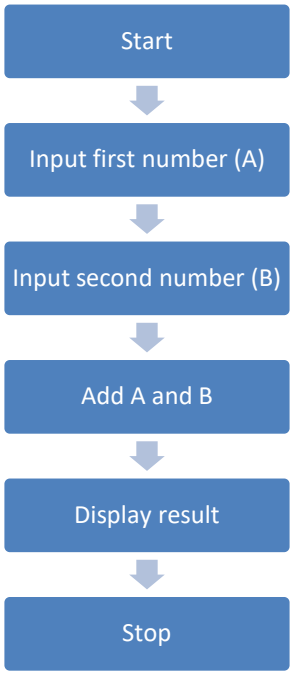
4. **Data Flows:**
- The data flows illustrate the data exchange that occurs between various parts of the system. This graph shows the way data is flowing in the diagram by customers who are interacting with the process of Orders, Inventory, and Payments.

This DFD presents the general working outline of how data passes within the Flipkart system, covering the communication processes between external and internal entities (Customers on one side and

Orders, Inventory and Payments on the other). With level of required detail imposed, additional processes, data stores, and data flows may be introduced to make the representation of a system more comprehensive.

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

A flowchart is an illustration of a process or algorithm via the depiction of the steps involved and the sequence in which they follow. Flow Charts give a representation of the control flow with the help of symbols and arrows. Undefined



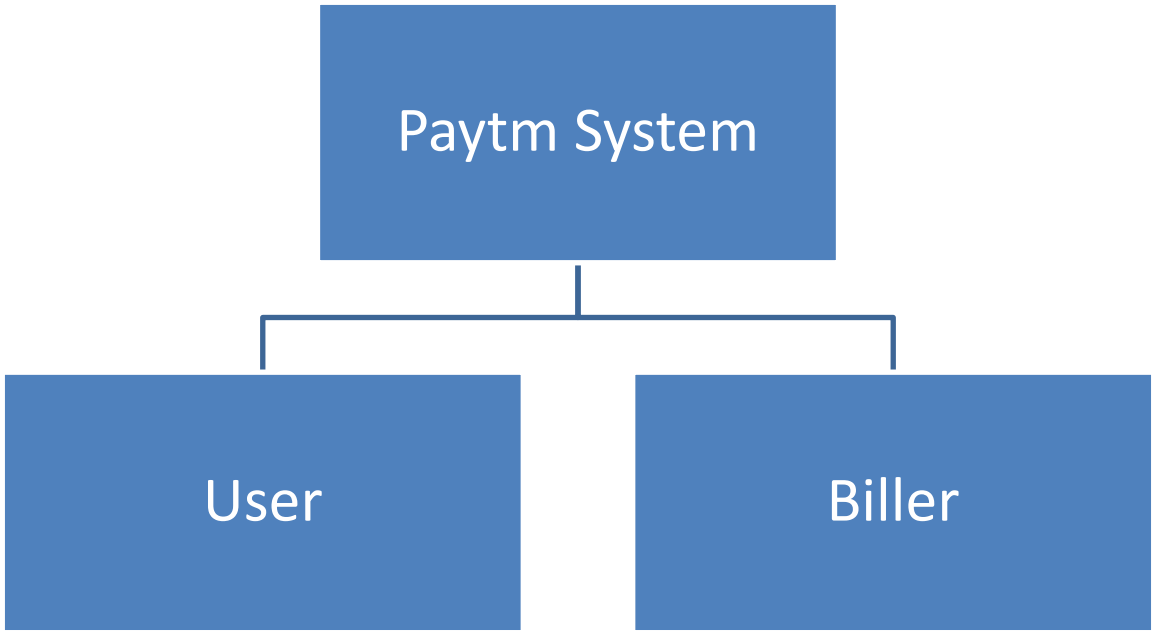
In this flowchart:

- **Start/Stop**: Indicates the start and the finishing points of the process. #
- **Input**: Computes the first number (A), followed by the second number (B).
- **Add**: Symbolizes the moment the two numbers get added up to each other.
- **Display result**: The symbol indicates the sum is written.
- **Arrows**: The directions point to the loops of control showing how the processes are carried out one by one.

The following flow chart is a simple algorithm for adding two numbers. The process commences with the user entering the two numbers (A and B) as inputs. And after that, the numbers are added together and the result is then shown.Finally, the process ends.

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

A use case diagram is a graphical representation of the step-scenes between actors (users or external systems) and a system considered. It covers the different systems users employ to achieve certain goals or a task. undefined



1. **Actors:**
- **User**: Is the end-user who contacts the Paytm system to conduct utility bill payments.
- **Biller**: It is the name of the individual or business to which payment from a user will be transferred (e.g., mobile service provider, utility company).

2. **Use Cases:**
- **Make Bill Payment**: It is the most prominent use case where it all starts by the user approaching our Paytm platform in order to pay his bills.
- **View Bill Details**: Alongside the primary use case, provides an additional function where the user can read a bill before they make the payment.
- **Confirm Payment**: Expresses the actual implementation of payment confirmation and fund withdrawal.
- **Receive Payment Notification**: Represents a scenario where a biller receives a notification that the user has paid for the amount due via the platform payment service.

- **Association**: Connectors joining actors with use-cases convey a relation between actors and their role within the system.

With the help of this process diagram we get a high level representation of the system flows between the user and the Paytm system when we make a bill payment. The user interface is accomplished by permitting the user to initiate bill payment process, viewing the bills, confirming the payment and getting the bill payment confirmation.