Module -1

1. What is software? Explain Types of software.

Ans. > In a computer system, the software is basically a set of instructions or commands that tell a computer what to do. In other words, the software is a computer program that provides a set of instructions to execute a user's commands and tell the computer what to do. For example like MS-Word, MS-Excel, PowerPoint, etc.

---> Types of Software

1.System Software

- Operating System
- Language Processor
- Device Driver

2.Application Software

- General Purpose Software
- Customize Software
- Utility Software

---> System Software

System software is software that directly operates the computer hardware and provides the basic functionality to the users as well as to the other software to operate smoothly. Or in other words, system software basically controls a computer's internal functioning and also controls hardware devices such as monitors, printers, and storage devices, etc. It is like an interface between hardware and user applications, it helps them to communicate with each other because hardware understands machine language(i.e. 1 or 0) whereas user applications are work in human-readable languages like English, Hindi, German, etc. so system software converts the human-readable language into machine language and vice versa.

--->Types of System Software

It has two subtypes which are:

- **Operating System:** It is the main program of a computer system. When the computer system ON it is the first software that loads into the computer's memory. Basically, it manages all the resources such as computer memory, CPU,

printer, hard disk, etc., and provides an interface to the user, which helps the user to interact with the computer system. It also provides various services to other computer software. Examples of operating systems are Linux, Apple macOS, Microsoft Windows, etc.

- **Language Processor:** As we know that system software converts the human-readable language into a machine language and vice versa. So, the conversion is done by the language processor. It converts programs written in high-level programming languages like Java, C, C++, Python, etc(known as source code), into sets of instructions that are easily readable by machines(known as object code or machine code).

- **Device Driver:** A device driver is a program or software that controls a device and helps that device to perform its functions. Every device like a printer, mouse, modem, etc. needs a driver to connect with the computer system eternally. So, when you connect a new device with your computer system, first you need to install the driver of that device so that your operating system knows how to control or manage that device.

---> Features of System Software

Let us discuss some of the features of System Software :

- System Software is closer to the computer system.
- System Software is written in a low-level language in general.
- System software is difficult to design and understand.
- System software is fast in speed(working speed).
- System software is less interactive for the users in comparison to application software.

---> Application Software

Software that performs special functions or provides functions that are much more than the basic operation of the computer is known as application software. Or in other words, application software is designed to perform a specific task for end-users. It is a product or a program that is designed only to fulfill end-users' requirements. It includes word processors, spreadsheets, database management, inventory, payroll programs, etc.

---> Types of Application Software

There are different types of application software and those are:

- **General Purpose Software:** This type of application software is used for a variety of tasks and it is not limited to performing a specific task only. For example, MS-Word, MS-Excel, PowerPoint, etc.
- **Customized Software:** This type of application software is used or designed to perform specific tasks or functions or designed for specific organizations. For example, railway reservation system, airline reservation system, invoice management system, etc.
- **Utility Software:** This type of application software is used to support the computer infrastructure. It is designed to analyze, configure, optimize and maintains the system, and take care of its requirements as well. For example, antivirus, disk fragmenter, memory tester, disk repair, disk cleaners, registry cleaners, disk space analyzer, etc.

---> Features of Application Software

Let us discuss some of the features of Application Software:

- An important feature of application software is it performs more specialized tasks like word processing, spreadsheets, email, etc.
- Mostly, the size of the software is big, so it requires more storage space.
- Application software is more interactive for the users, so it is easy to use and design.
- The application software is easy to design and understand.
- Application software is written in a high-level language in general.

2. Explain the SDLC Each phase process.

Ans. > The Software Development Life Cycle (SDLC) is a process used by software development organizations to plan, design, develop, test, deploy, and maintain software applications.

SDLC (Software Development Life Cycle) is used in Every Software Development Company because it is the root of the Development Cycle, if that model would not exist in the world, firstly no software can build secondly if any how it would be made, it's not going to succeed it has no use, because of no maintenance, but Luckily SDLC model exist in Tech world But why we need it Actually!

---> There are several reasons why organizations use the Software Development Life Cycle (SDLC) when developing software applications:

To provide a structured and organized approach to software development: The SDLC provides a framework for managing the software development process, which helps to ensure that all necessary steps are taken and that the final product meets the requirements.

1. To ensure that the software is of high quality: The SDLC includes testing and quality assurance phases, which help to ensure that the software is free of bugs and that it meets the requirements.
2. To manage risks and costs: The SDLC helps organizations to identify and manage risks early in the development process, which can help to reduce costs and minimize the impact of any issues that do arise.
3. To improve communication and collaboration: The SDLC helps to ensure that all stakeholders, including customers, end-users, and developers, are involved in the development process and that their needs are taken into account.
4. To improve efficiency and productivity: The SDLC helps organizations to optimize the use of resources and to streamline the development process, which can improve efficiency and productivity.
5. To increase the likelihood of a successful project outcome: Following a well-defined SDLC process can greatly increase the chances of success of the project, as the process guides the team towards the goal in a systematic and efficient way.

Overall, the SDLC is a valuable tool for organizations to use when developing software applications, as it helps to ensure that the final product is of high quality, meets the requirements, and is delivered on time and within budget.

---> The SDLC typically includes the following phases:

Requirement collection

- Phase of collecting requirements from client
- Will be done by business analyst of company
- He will create questioner, in which put answers from client

Analysis & SRS (Software Requirement Specification)

- Collected requirements will be analyzed for time limit, budget and market trade
- Will be filtered and SRS will be created as result
- Will be discussed with client
- Based on requirements users, their activities and flow of data, modules will be defined
- Will be done by system analyst
- Based on diagrams all will be done
- Main diagrams are use-case, DFD and flow charts

Designing

- The Design document should reference what you are going to build to meet the requirements, and not how it can include pseudo code but shouldn't contain actual code functionality.
- Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams , pseudo code, and a complete entity-relationship diagram with a full data dictionary.
- These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input. At this phase the test plans are developed.

Implementation (Coding)

- To launch the coding phase, develop a shell program that is then put under some form of version control.
- This phase includes the set up of a development environment, and use of an enhanced editor for syntax checking.

Testing

- Each developer insures that their code runs without warnings or errors and produces the expected results.
- The code is tested at various levels in software testing. Unit, system and user acceptance tasting's are often performed. This is a grey area as many different opinions exist as to what the stages of testing are and how much if any iteration occurs.
- **Types of testing:** Defect testing, Path testing, Data set testing, Unit testing, System testing , Integration testing, Black box testing, White box testing, Regression testing, Automation testing , User acceptance testing, Performance testing, etc.

Maintenance

- User's guides and training are developed to reflect any new functionality and changes which

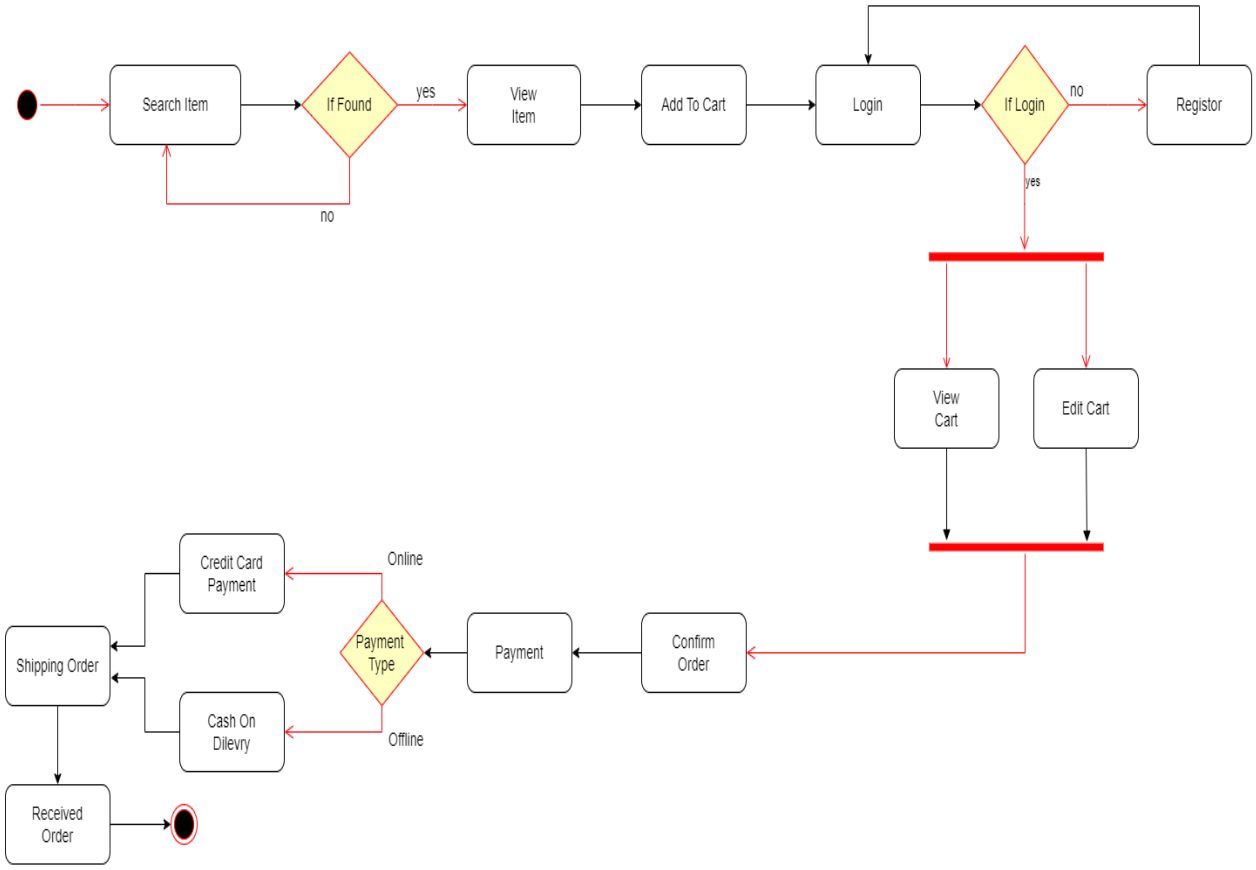
need to be identified to the production staff.

- Any changes needed to operations and/or maintenance need to be addressed.
- Every run in production needs to be verified. Any problems with production need to be addressed immediately.
- A Change Request system may be set up to allow for feedback for enhancements.

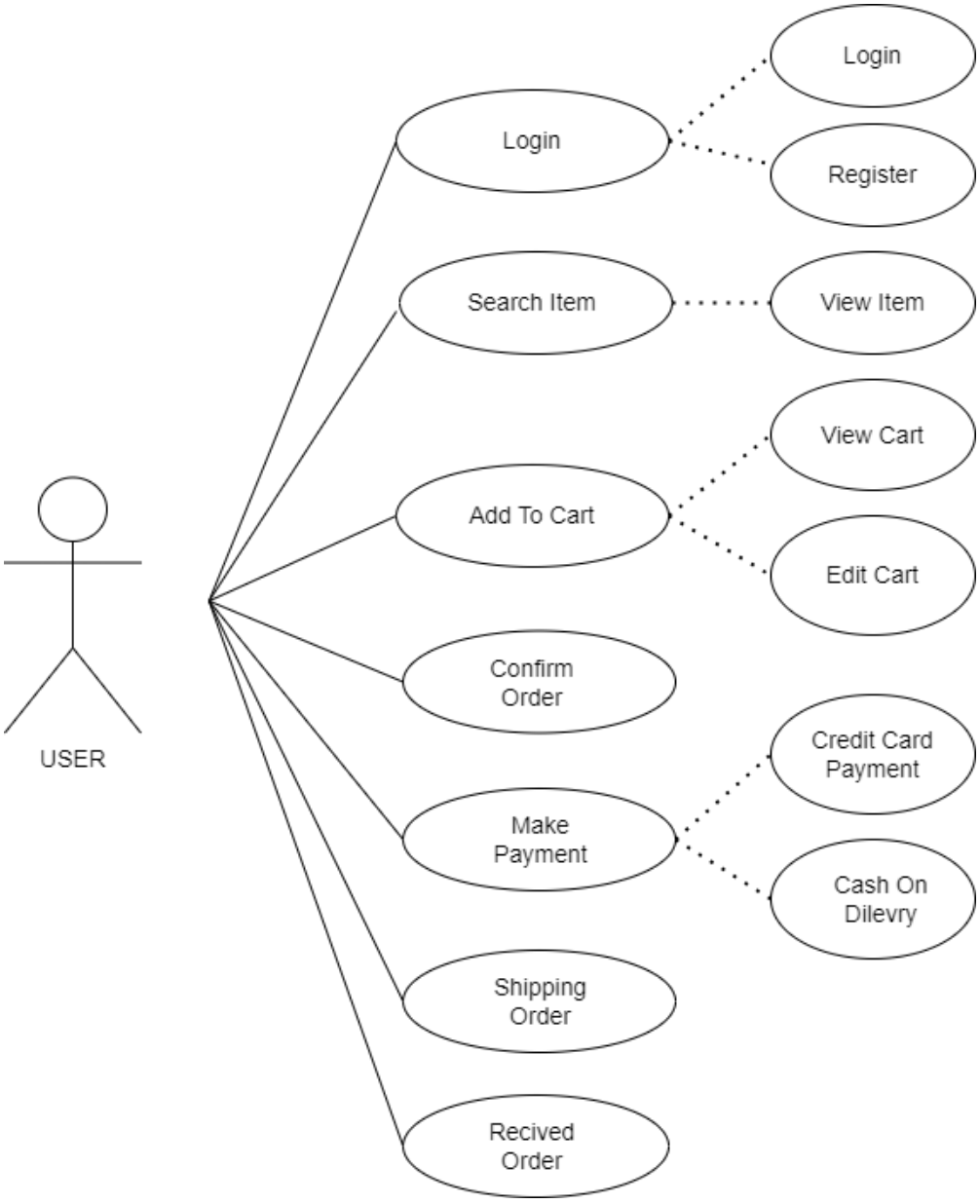
3. Create the DFD and use case of Flipkart .

Ans. >

---> Data Flow Diagram



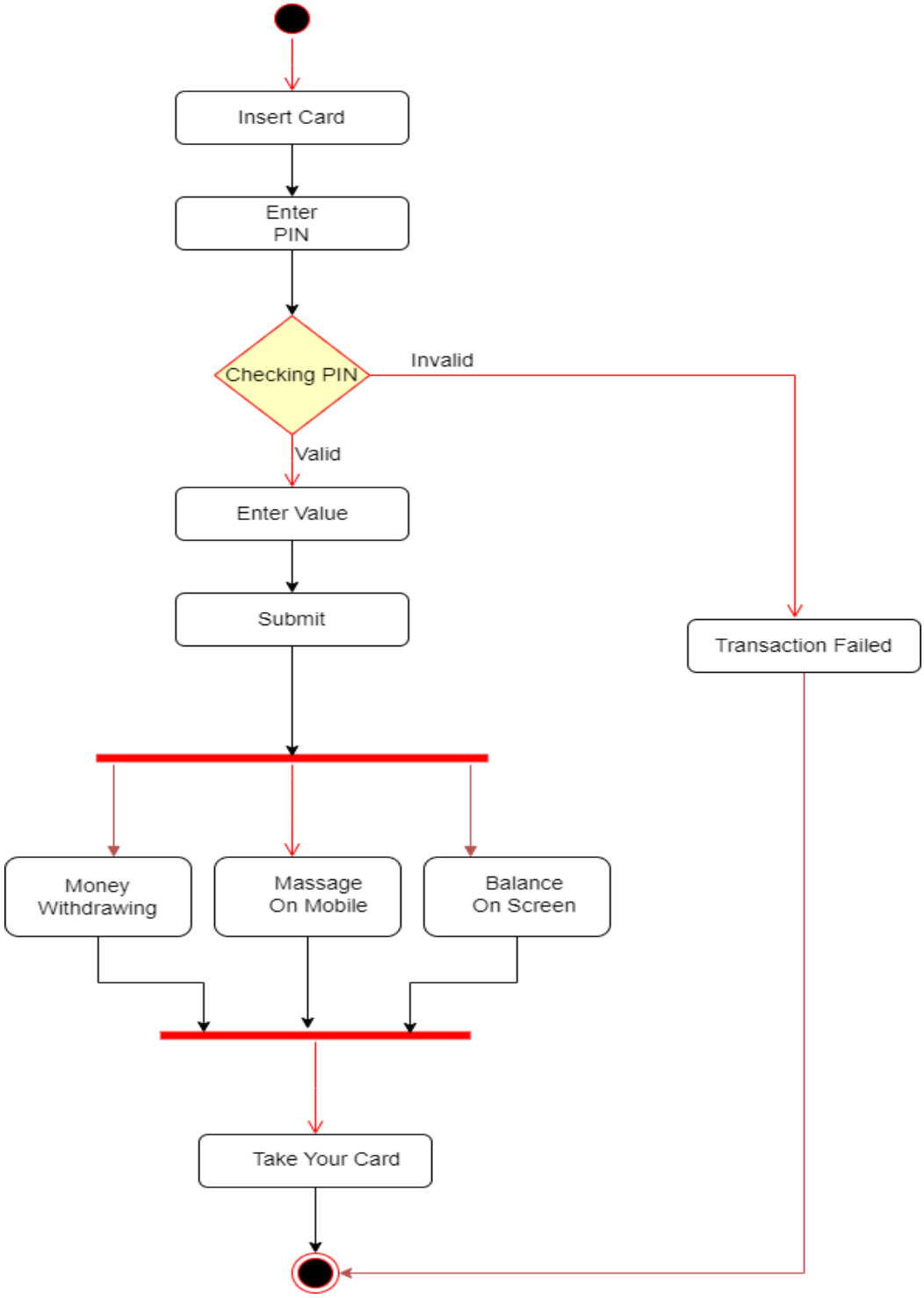
---> Use Case Diagram



4. Create the DFD and use case of ATM system

Ans. >

---> Data Flow Diagram



---> Use Case Diagram