UNIT-II

Requirements Engineering Analysis and Specifications

Q What is Requirement Engineering?

Ans Requirement Engineering is the process of eliciting stakeholder needs and desires and developing them into an agreed-upon set of detailed requirements that can serve as a basis for all subsequent development activities.

Q What are the types of Requirements?

Ans:- 1 User Requirements :- The User Requirements Specification describes the business needs for what users require from the system. User Requirement specifications are written early in the validation process. typically before the System is created They are written by the system owner and end-user with input from Quality Assurance.

Q ~~Who~~ is Stakeholder?

System Requirement :- A structured document setting out detailed description of the System's function, service and operational constraints. Defines what should be implemented so may be part of Contract b/w Client and Contractor.

Q ~~Who~~

Readers of different types of requirements

* User Requirement
  ↓
  Client managers
  System end-users
  Client engineers
  Contract managers
  System architectures

* System requirements
  ↓
  System end-users
  Client engineers
  System architects
  Software developers

Q Who are Stakeholders?

Ans It is important to identify the stakeholders in the project. Stakeholders are the people who will find benefit in the project and the software being developed.
They may include:-

- Customers
- End users
- Business operations managers
- Product managers
- Advertising and marketing staff

Q What are the steps involved in requirement Engineering?

Ans :- 1 Inception :-

In this step there are lot of things examined like the problem which the software should solve and gaining an understanding of both the problem's nature and the nature of the desired solutions.

The following questions may be asked :-
Who is going requesting the software?
Who will use the software?
What is the benefit that the software will bring?

2 Elicitation

In this Step the overall problem which software is attempting to solve is identified.
Team ~~the~~ come up with many solutions.
~~Also in this step~~ Then there is negotiation ~~bif~~ between the different approaches to solving the problem and finally specify a basic set of requirements.

This can be done by calling a meeting between all of the stakeholders. It is important to nominate someone act like a facilitator who will guide the meeting.

3 Negotiation

- This step involves negotiating b/w the various stakeholders in order to remove any conflict in the requirements.
- A useful technique for resolving these Conflicting requirements is to provide each of the stakeholder with a finite number of priority points. They may then allocate points b/w the conflicting requirements as they see it.

4 Specification

The specification step produces the final product of the requirements engineering process.

The specification need not to be a written document, but could also be a graphical model, software prototype or formal model or a collection of these.

5 Validation

This Step is Concerned with ensuring that the gathered requirements in the software specification ~~met~~ meet Certain ~~standards~~ standards of quality.

A useful action during validation is to ensure that each requirement has a source attributed to it. In this way, if more information is required, the requirements engineers know who to contact.

6 Management

Requirement changes our time. Requirements management is concerned with controlling and tracking change in the requirements.

Requirements management proceeds by associating requirements with various aspect of the software engineering process. As these aspects are changed the requirements are changed all aspects of development associated with the modified requirements can be examined, and in this way the changes can more easily be propogated through the project.

SRS

After the analyst has gathered all the required information regarding the software to be developed and has removed all incompleteness, inconsistencies and anomalies from the specification he start to systematically organise the requirements in the form of an SRS document. The SRS usually contains all the user requirements in a structured though an informal form.

Characteristics of a Good SRS document

1. **Correctness** :- User review is used to ensure the correctness of requirements stated in the SRS.

2. **Consistency** :- Requirements in SRS are said to be consistent if there are no conflicts between any set of requirements. Examples of conflict include differences in terminologies used at separate places, logical conflicts like time period of report generation etc.

3 Unambiguousness :- ASRS is said to be unambiguous if all the requirements stated have only 1 interpretation. Some of the ways to prevent unambiguousness include the use of modelling techniques like ER diagrams, proper reviews etc.

4 Modifiability :- SRS should be made as modifiable as possible and should be capable of easily accepting changes to the system to some extent. Modifications should be properly indexed and cross-referenced.

5 Traceability :- One should be able to trace a requirement to design component and then to code segment in the program. Similarly one should be able to trace a requirement to the corresponding test cases.

6 Design Independence :- There should be an option to choose from multiple design alternatives for the final system. More specifically the SRS should not include any implementation details.

7 Testability: A SRS should be written in such a way that it is easy to generate test cases and test plans from the document.

8 Understandable by the Customer:

An end user maybe an expert in his/her domain but might not be expert in computer science. Hence, the use of formal notations and symbols should be avoided to as much extent as possible. The language should be kept easy and clear.

Use Case

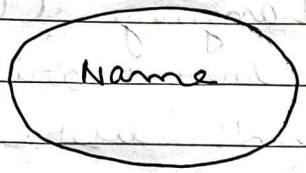
In the Unified Modeling Language (UML) a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:-

1. Scenarios in which your system or application interacts with people, organizations or external systems.

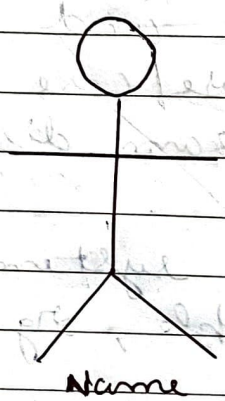
2. Goals that your system or application helps those entities (known as actors) achieve.

3. The scope of your system.

User Case Elements.

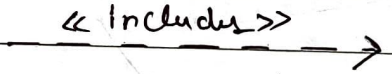


Use Case

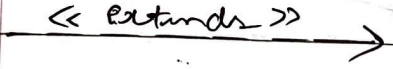


Actor

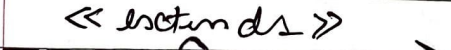
Association



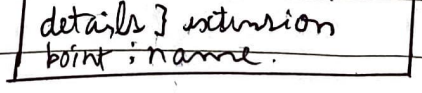
Includes relationships
[It is mandatory and it will definetly take play]



Extends relationships.



and with defined extension point.
[It is not mandatory depends on criteria]



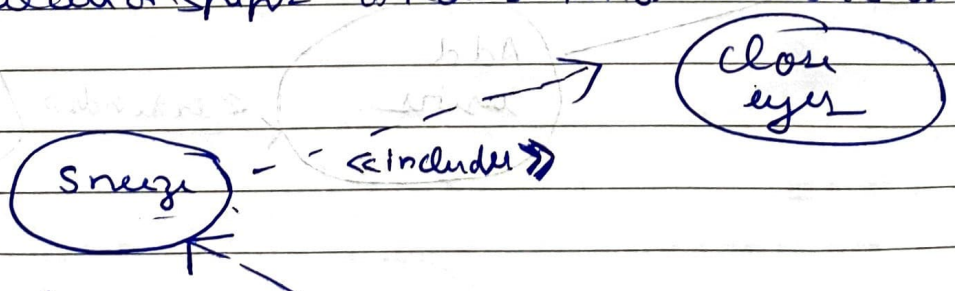
Use Case diagram Components

1 Actors :- The users interact with a system. An actor can be a person, an organisation, or an outside systems that interact with your application or system. They must be an external objects that produce or consume data.

2 System :- A specific sequence of actions and interactions between actors and the system. A system may also be reffered as a scenario.

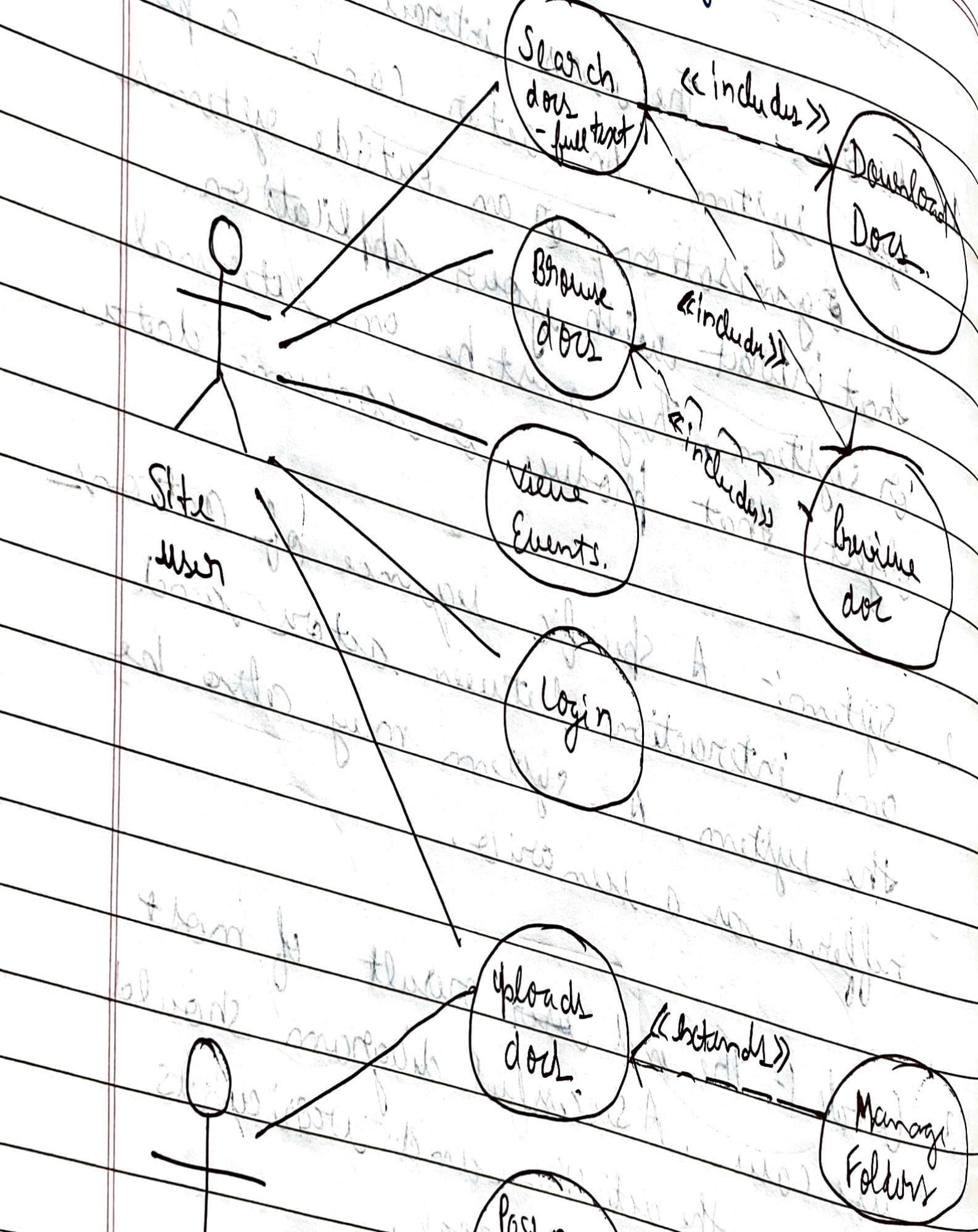
3 Goals :- The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

Q What is the difference b/w include relationships and extend relationships ?



अगर हमें छींक आये तो हम अपनी आँखे बंद कर लेते हैं। वो हम रोक नहीं सकते। तो वो है Include relationship.

अगर छींकने के बाद Excuse me कहना हमारे ऊपर Depend करता है। ये कोई Mandatory event नहीं है इसलिये ये Include relationship है।


Search docs - full text
«includes»
Download Docs.
Browse docs
«includes»
«includes»
View Events.
Review doc
Site user
Login
Uploads docs.
«extends»
Manage Folders

Use of Use Case Diagram

Use cases diagrams have two main uses :-

1. modelling Contexts
2. modelling requirements.

Although these two types of use should be related they are distinctly different.

Q How to create a use case diagram?

Ans:- 1 Identify the Actors (role of users) of the system.

2 For each category of users, identify all roles played by the users relevant to the system.

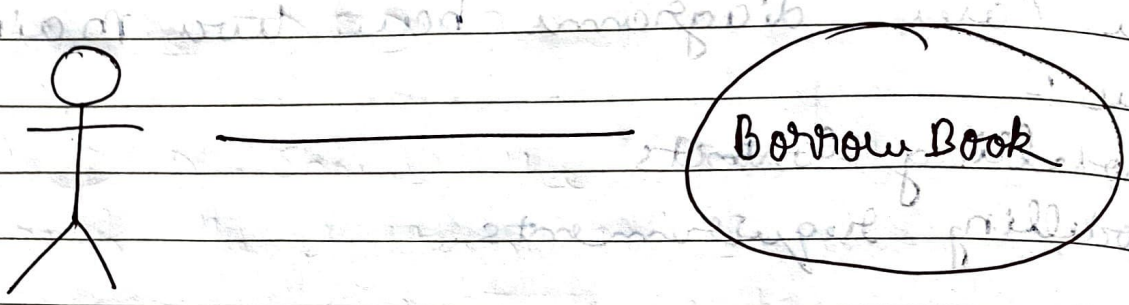
3 Identify what are the users required the system to be performed to achieve these goals.

4 Create use case for every goal.

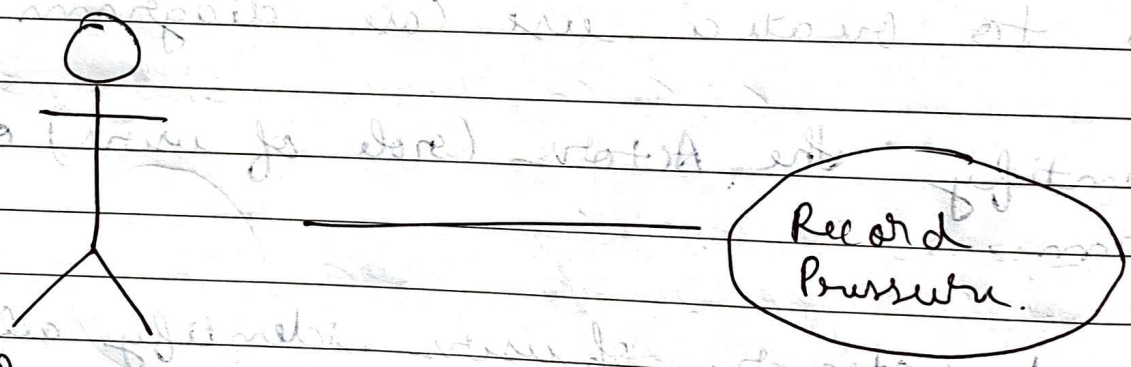
5 Structure the use cases.

6 Prioritize, review, estimate and validate the uses.

Two Simple Use Cases.

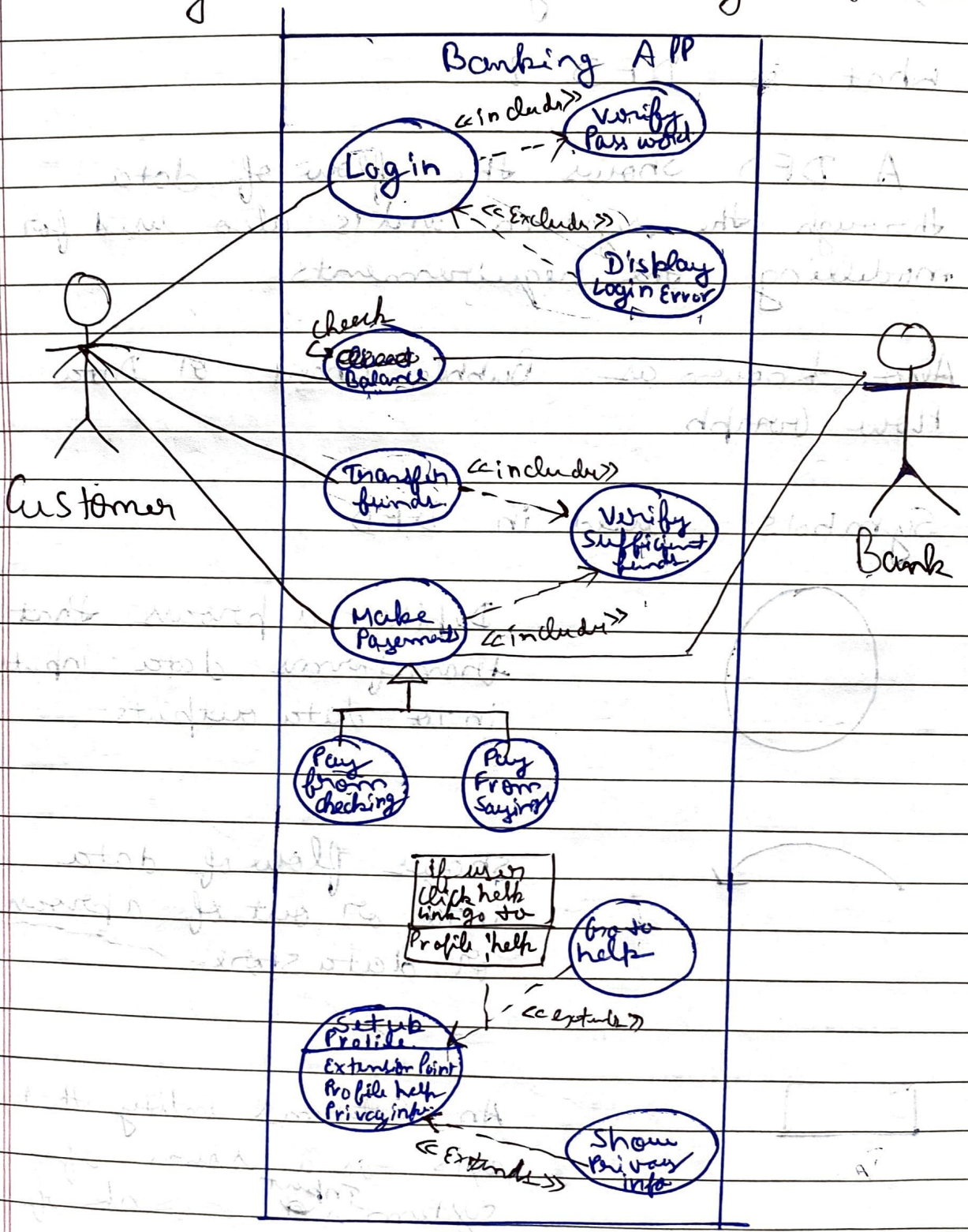


Book Borrower.



Pressure Sensor

Banking APP use Case Diagram



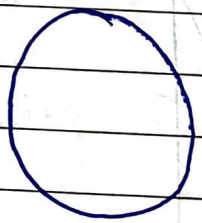
Data Flow Diagram (DFD)

Q What is DFD ?

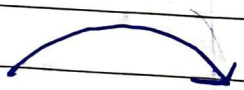
Ans A DFD shows the flow of data through the system and is also used for modelling the requirements.

Also known as Bubble Chart or Data Flow Graph.

Symbols used in DFD



Depicts a process that transforms data inputs into data outputs.

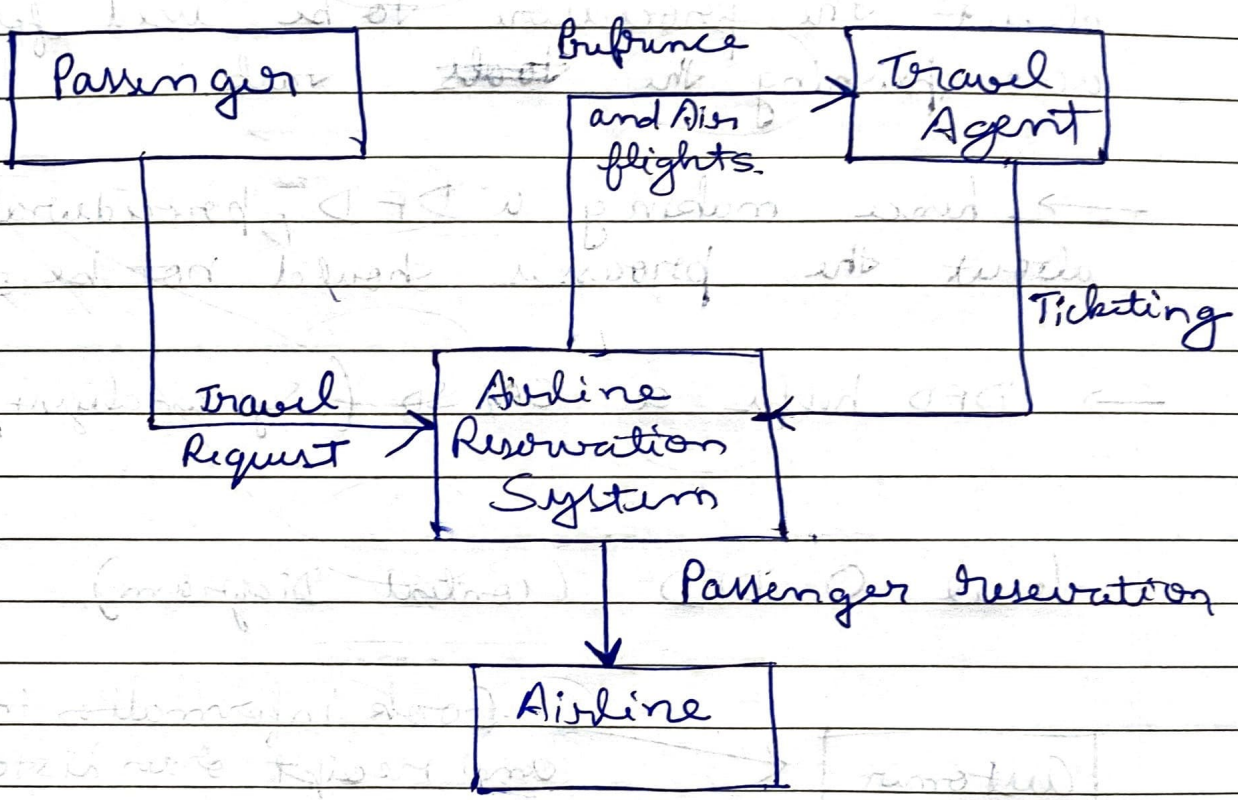


Shows flow of data into or out of a process or data store.

An External entity that acts as a source of system Input or sink of systems output.

Construction of DFD

Eg. The Context level flow diagram for an airline reservation system!



IEEE defins DFD as :-

a diagram that depicts data sources, data sinks, data storage and process performed on data as nodes and logical flow or links b/w the nodes.

1. it represents the system data in a hierarchal manner and with requirement levels of detail.

2. it depicts process acc to defined user requirements and s/w scope.

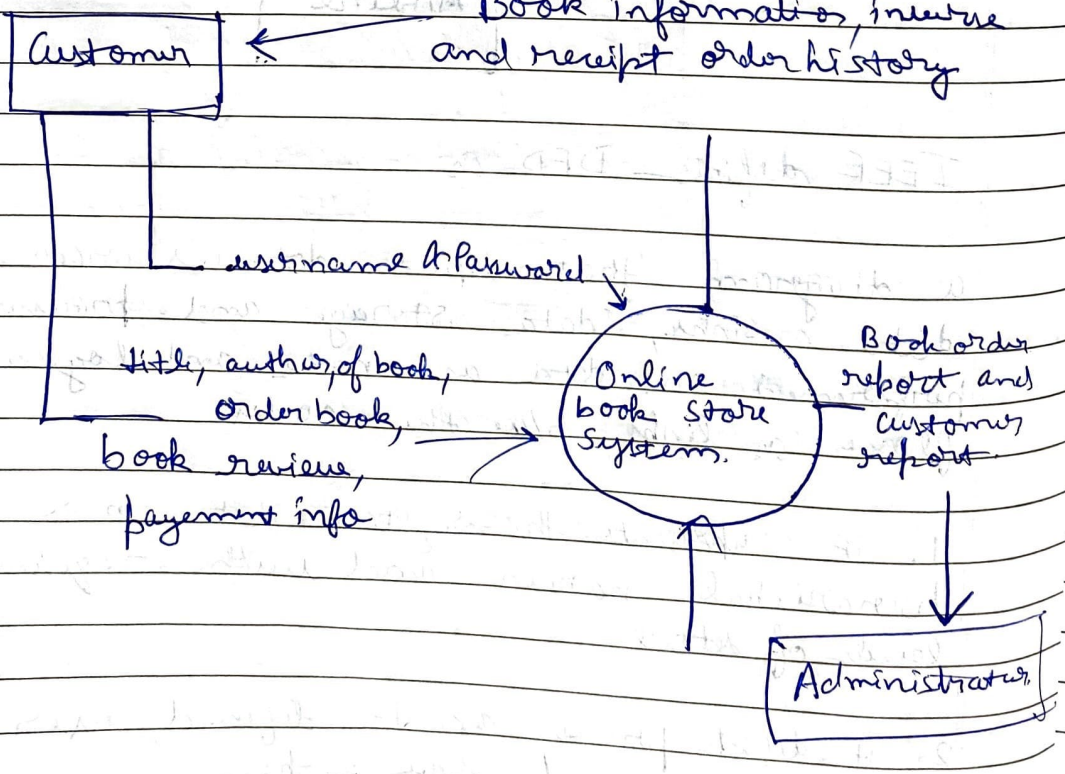
DFD Vs Flow ~~diagram~~ chart :-

DFD represents the flow of data whereas flow chart represent the flow of control. Also a DFD doesn't depict the information about the procedure to be used for accomplishing the ~~task~~ task.

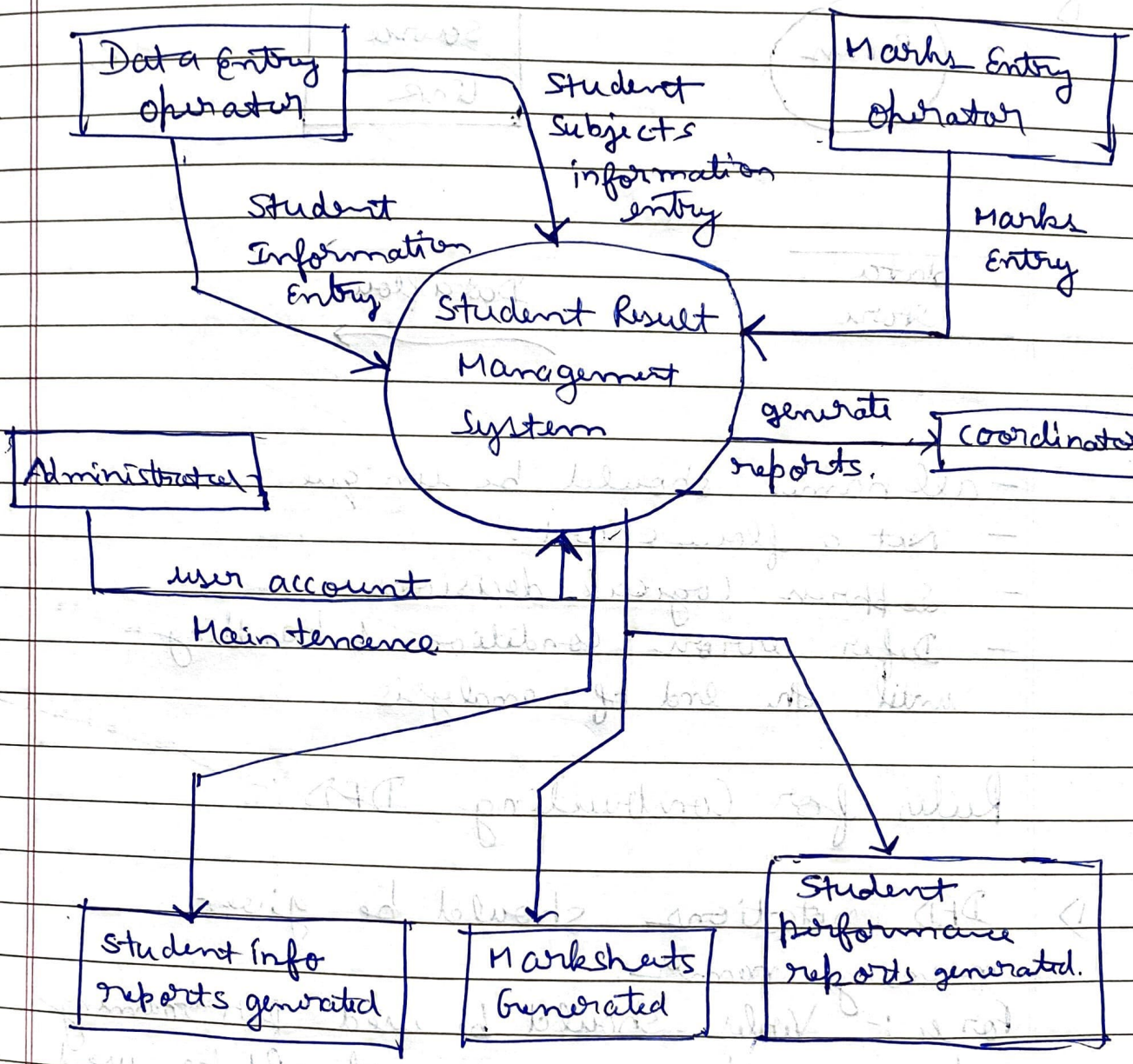
→ Hence making a DFD, procedural details about the process should not be shown.

→ DFD helps a SA/SD (Sys analysist / Syst designer)

level 0 DFD (Context Diagram)

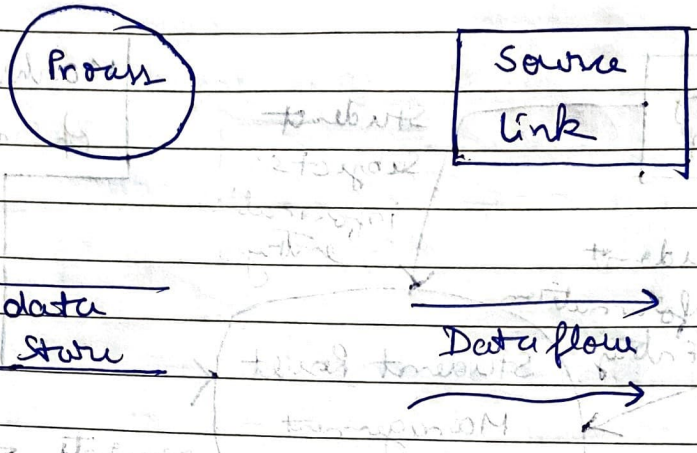


* Level 0 DFD for Student Result Management System :-



Symbols

1)



- all names should be unique.
- Not a flow chart.
- Suppress logical decisions
- Defer error conditions & handling until the end of analysis.

Rules for Constructing DFD :-

1) DFD notations should be given meaning names.
For ex :- Verbs should be used for naming a process, whereas nouns should be used for naming external entity (Data source/sinks) as well as data stores and data flows.

2) Abbreviation should be avoided in DFD notations.

3) Each process should be numbered uniquely but the numbering should be Consistent.

4) A DFD should be ~~Cor~~ created in organized manner so that it easily understood.

5) Unnecessary notation should be avoided in DFD so as to avoid complexity.

6) A DFD should be logically consistent.

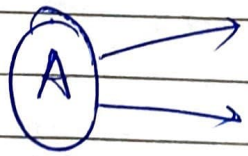
7) There should be no loops in DFD whereas loops in flow chart.

8) A DFD should be refined until each process performs a simple function.

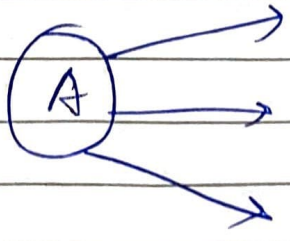
9) A DFD should be organised in series of level so that each level provide more details than the previous level.

10) The name of the process should be carried to the next level of DFD.

11) No process can have only outputs

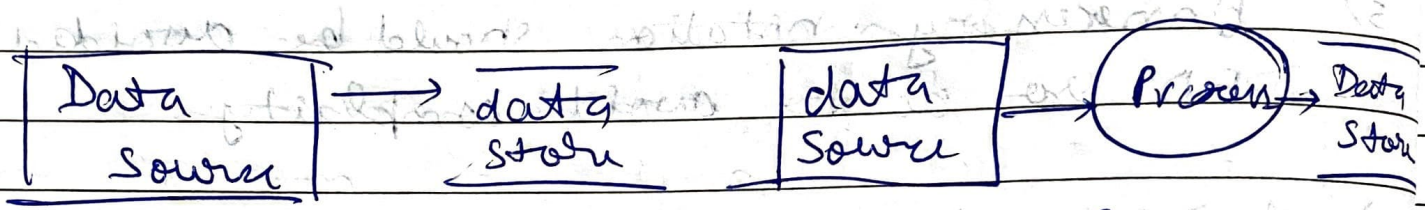


Incorrect because without any input.



true

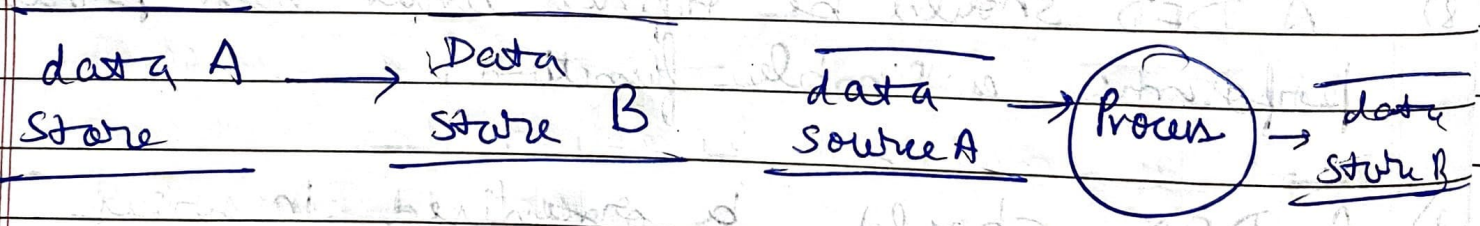
Data can't move directly from an outside source to a data store.



Incorrect

Correct

Data can't move directly from one data store to another data store.

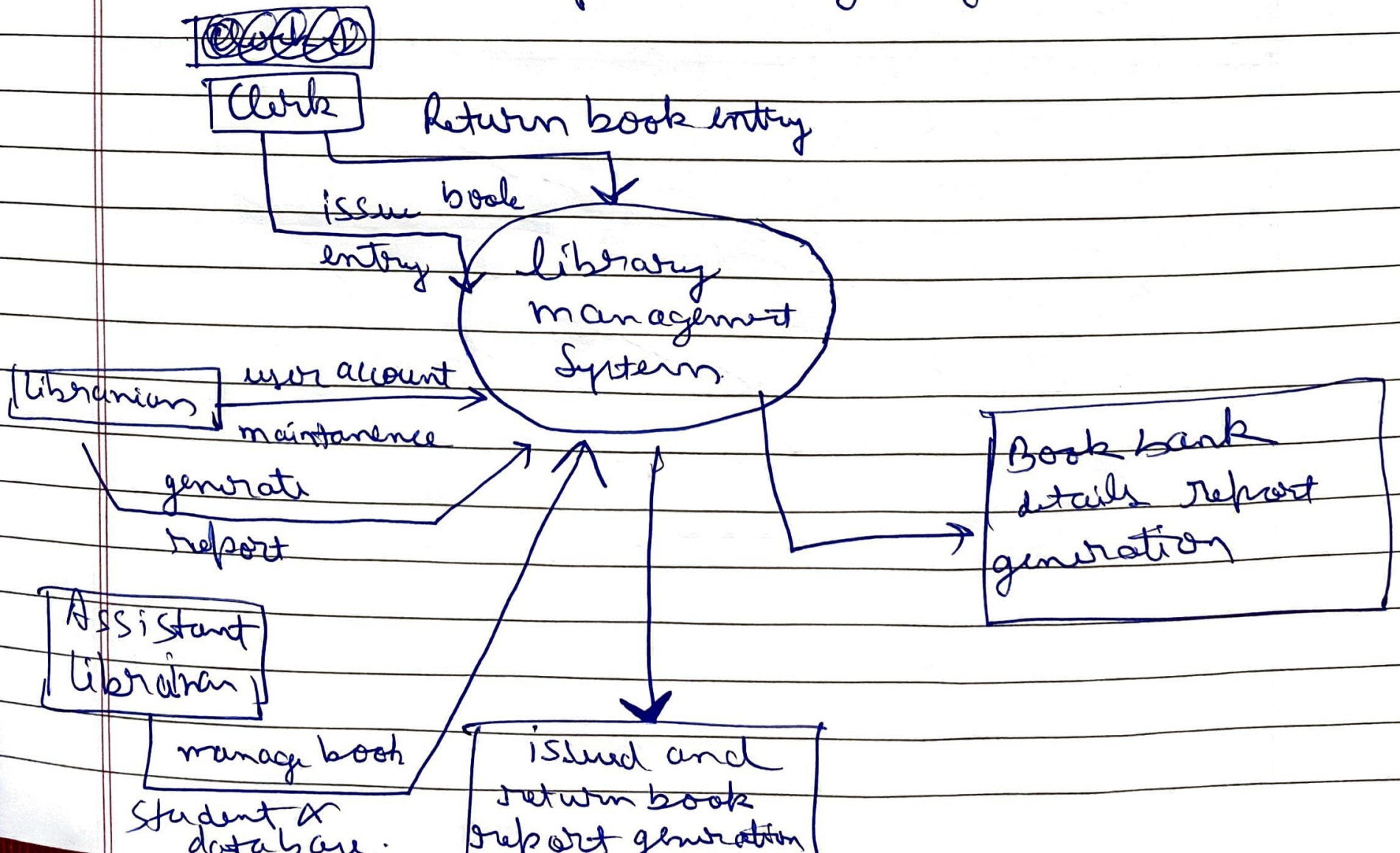


Incorrect.

Correct

and external entities.
It is top level DFD. It contain only one process node, that generalise.

* level 0 DFD for library system



Level-1 DFD :-

Shows how the system is divided into sub-systems (processes) each of which deals with one or more of the data flows to or from an external agent and which together provide all of the functionality of the system as a whole.

Level 2 DFD :-

It offers a more detailed look at the processes that make up an info system that level-1 does.