EXPERIMENT NO: 05

Aim: Design USE case diagram, state diagram for given scenario.

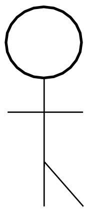
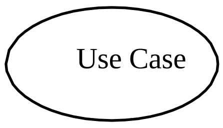
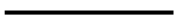
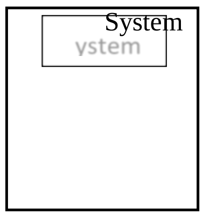
Theory:

- **What is a USE case diagram?**
  In the Unified Modelling 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:
  - Scenarios in which your system or application interacts with people, organizations, or external systems.
  - Goals that your system or application helps those entities (known as actors) achieve.
  - The scope of your system.

- **When to apply use case diagrams**

  Use case diagrams specify the events of a system and their flows. But use case diagram never describes how they are implemented. Use case diagram can be imagined as a black box where only the input, output, and the function of the black box is known.
  These diagrams are used at a very high level of design. This high-level design is refined again and again to get a complete and practical picture of the system. A well-structured use case also describes the pre-condition, post condition, and exceptions. These extra elements are used to make test cases when performing the testing.

  Use case diagrams can be used for –
  - Requirement analysis and high-level design.
  - Model the context of a system.
  - Reverse engineering.
  - Forward engineering

Notation Description	Visual Representation
Actor Someone interacts with use case (system function). Similar to the concept of user, but a user can play different roles Actor has a responsibility toward the system (inputs), and Actor has expectations from the system (outputs).	
Use Case System function (process - automated or manual) i.e. Do something Each Actor must be linked to a use case, while some use cases may not be linked to actors.	
Communication Link Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using messages.	
Boundary of system The system boundary is potentially the entire system as defined in the requirements document. For example, for an ERP system for an organization, each of the modules such as personnel, payroll, accounting, etc.	

Relationships in Use Case Diagrams

There are five types of relationships in a use case diagram. They are

- Association between an actor and a use case
- Generalization of an actor
- Extend relationship between two use cases
- Include relationship between two use cases
- Generalization of a use case

- **How to Create a Use Case Diagram**

  - **Identifying Actors**
    Actors are external entities that interact with your system. It can be a person, another system or an organization. In a banking system, the most obvious actor is the customer. Other actors can be bank employee or cashier depending on the role you're trying to show in the use case.

  - **Identifying Use Cases**
    Now it's time to identify the use cases. A good way to do this is to identify what the actors need from the system. In a banking system, a customer will need to open accounts, deposit and withdraw funds, request check books and similar functions. So, all of these can be considered as use cases.

    Top level use cases should always provide a complete function required by an actor. You can extend or include use cases depending on the complexity of the system.

  - **Look for Common Functionality to use Include**
    Look for common functionality that can be reused across the system. If you find two or more use cases that share common functionality you can extract the common functions and add it to a separate use case. Then you can connect it via the include relationship to show that it's always called when the original use case is executed.

  - **Is it Possible to Generalize Actors and Use Cases?**
    There may be instances where actors are associated with similar use cases while triggering a few use cases unique only to them. In such instances, you can generalize the actor to show the inheritance of functions.

    One of the best examples of this is "Make Payment" use case in a payment system. You can further generalize it to "Pay by Credit Card", "Pay by Cash", "Pay by Check" etc. All of them have the attributes and the functionality of payment with special scenarios unique to them.

  - **Optional Functions or Additional Functions**
    There are some functions that are triggered optionally. In such cases, you can use the extend relationship and attach an extension rule to it. In

the below banking system example “Calculate Bonus” is optional and only triggers when a certain condition is matched.

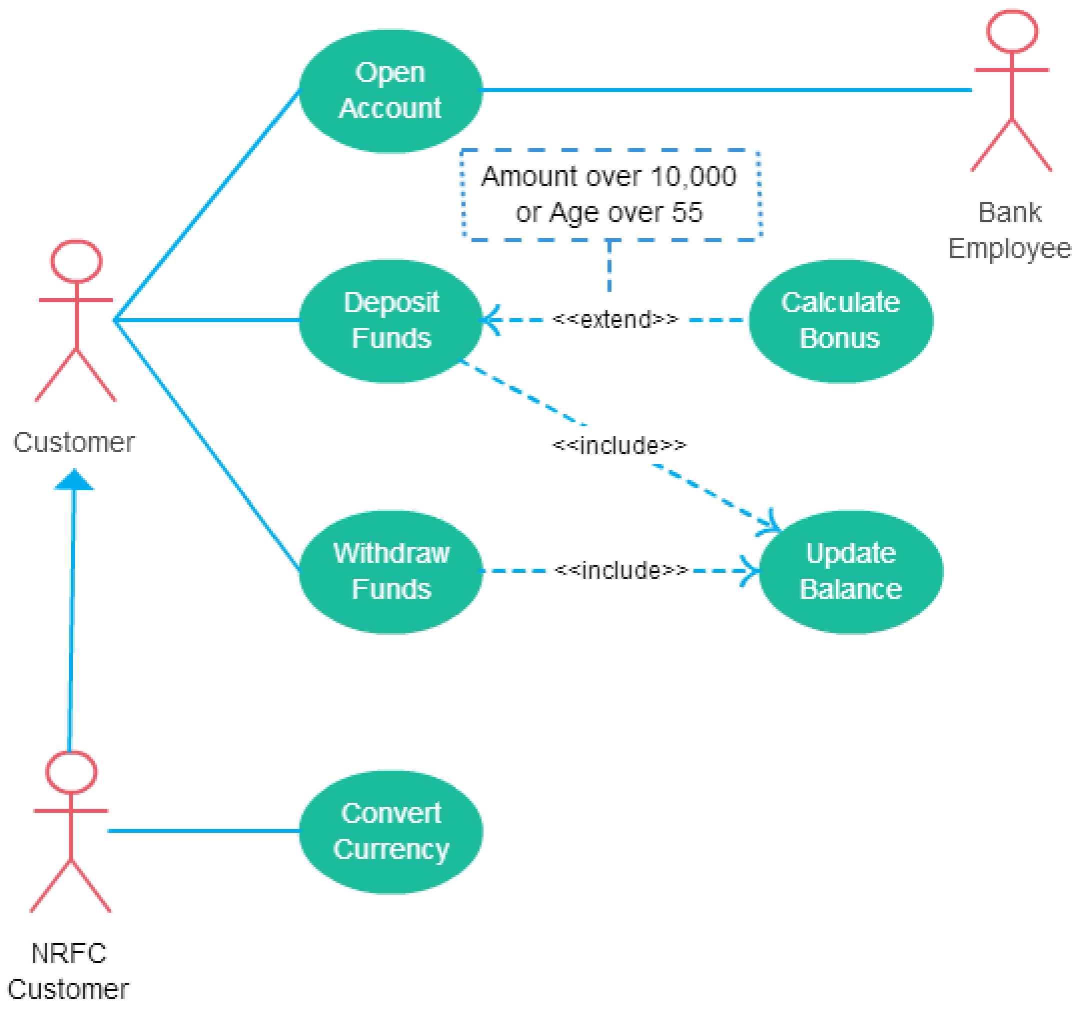


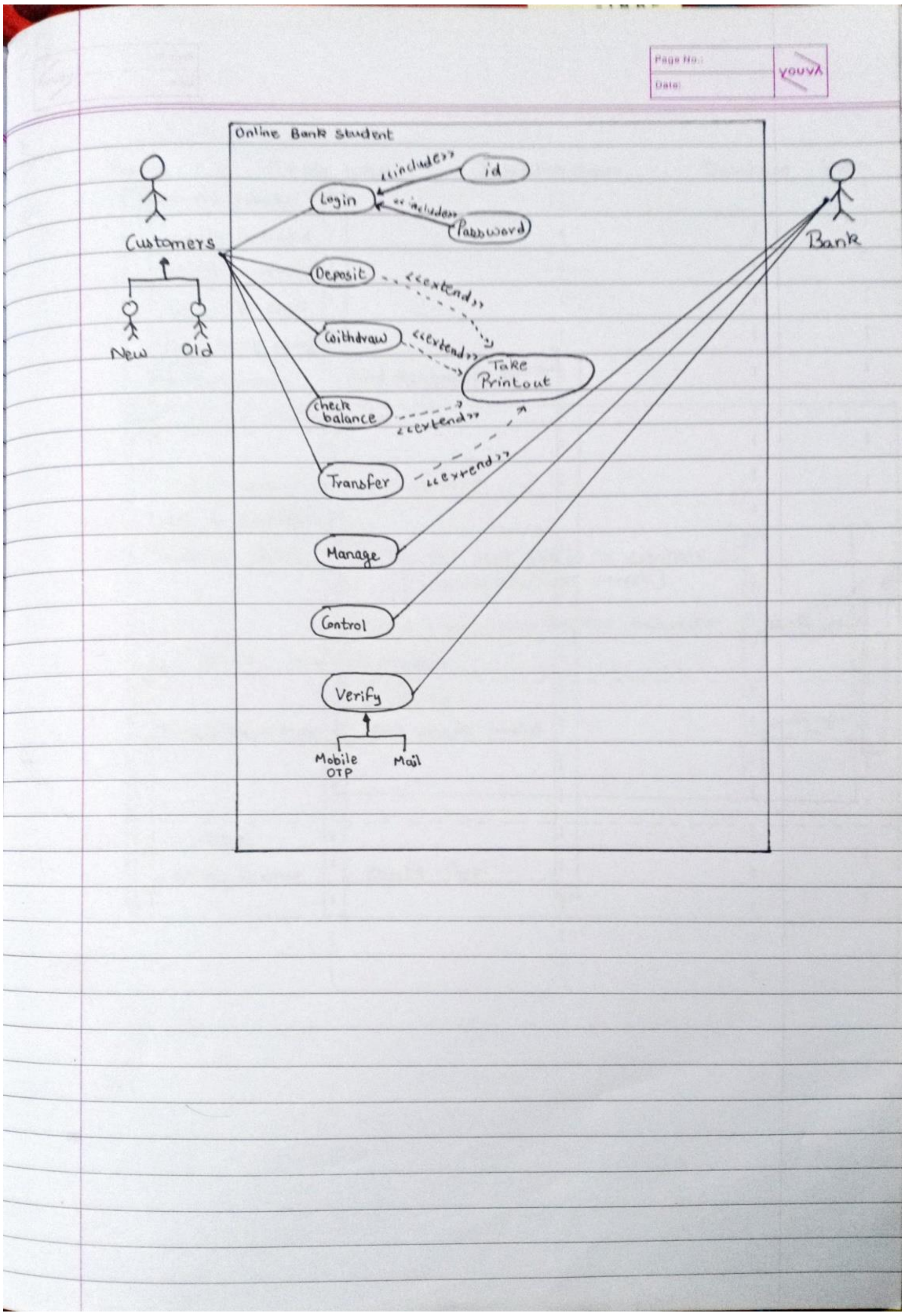
Fig. Example of use case diagrams

- **Scenario: Online Banking System**

- **Questions:**

1. **What is the Importance of Use Case Diagrams?**
    - The use case diagram can summarize the details of your system's users (also called as actors) and their interactions with the system.
    - Use case diagrams specify the events of a system and their flows, An effective use case diagram can help the team in discussing and representing factors such as:
        - Scenarios in which your system or application interacts with people, organisations, or external systems.
        - Goals that your system or application helps those entities (known as actors) achieve.
        - The scope of your system.
    - A well-structured use case also describes the pre-condition, post condition and exceptions of the software system.

2. **Draw a USE case diagram for given scenario?**



3. **Are use cases the same as functional requirements are different from use cases?**
    - No, use case diagrams are different from functional requirements.
    - Functional requirements are a set of requirements that define the system functionality being developed.
    - Functional requirements is mostly in text form.
    - Whereas, use case diagrams can be said to be a pictorial / diagrammatic representation of a software system.
    - The major difference is that use case diagrams are a graphical representation of the systems requirements, whereas functional requirements are in text form.
    - Use case diagrams can also have text in them, but the major focus is on the diagram itself, whereas in functional requirements the major focus is on the written text.

4. **Which part of a use case description can also be modelled by using an activity diagram? and why?**
   - It is possible to model the events and activities of a cuse case diagram in an activity diagram.
   - It is because in the use case diagram the data flow in the system shows with the help of flow activity part which is also used in activity diagram to show the flow of data in the system.
   - By using that part, both the diagrams show the flow of data or sequence.

- **Conclusion:**

  Hence, by performing this experiment on the Use Case Diagrams. I learnt about the use case diagram, the various notations, descriptions, visual representation and how to create it. I also designed Use Case Diagram for the given scenario.

(10)	**(20)**	**(10)**	**(10)**	**TOTAL**

EXPERIMENT NO: 06

Aim: Create Sequence diagram, state diagram for given scenario.

Theory:

- **What is a Sequence diagram?**

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are sometimes called event diagrams or event scenarios. Sequence diagrams are preferred by both developers and readers alike for their simplicity.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

- **High-Level Sequence Diagrams:**

High-level sequence diagrams give a good overview of the interactions between customers, partners, and the business system. They serve as the basis for the electronic data transfer between the business system and customers, business partners, and suppliers (see Modelling for System Integration).

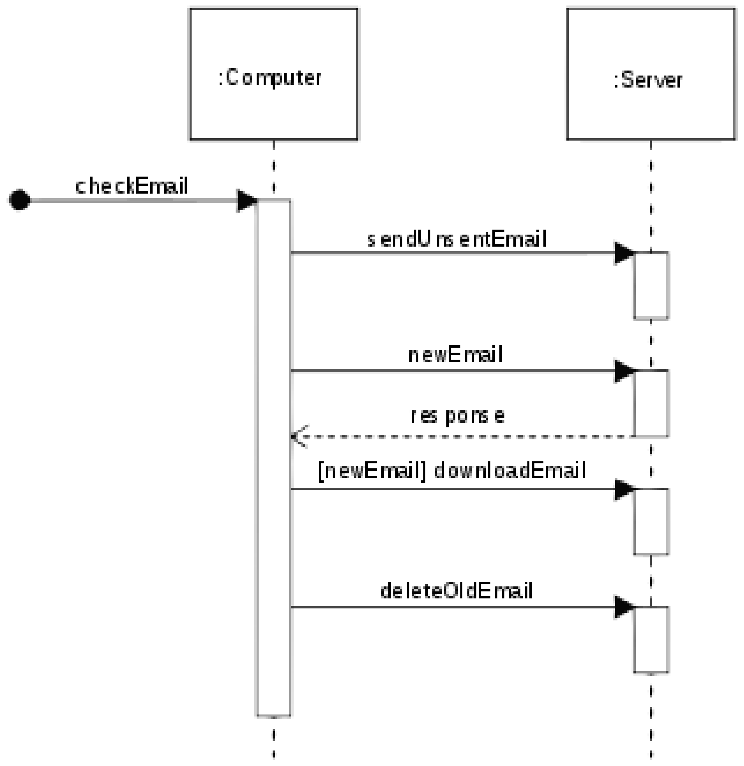


Fig: Example of sequence diagram

• What is State diagram?

A state diagram is a type of diagram used in computer science and related fields to describe the behaviour of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

State diagrams are used to give an abstract description of the behaviour of a system. This behaviour is analysed and represented as a series of events that can occur in one or more possible states. Hereby "each diagram usually represents objects of a single class and track the different states of its objects through the system".

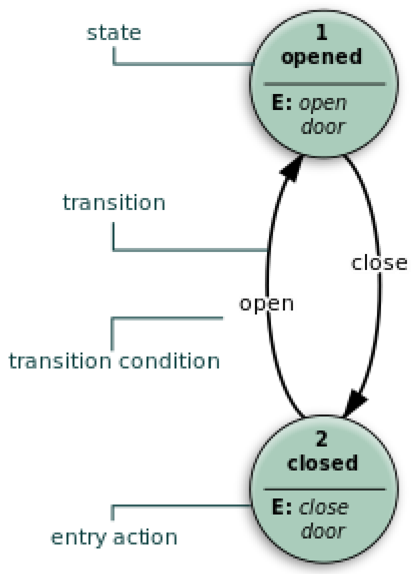


Fig: Example of State diagram

- **Purpose of state diagram:**

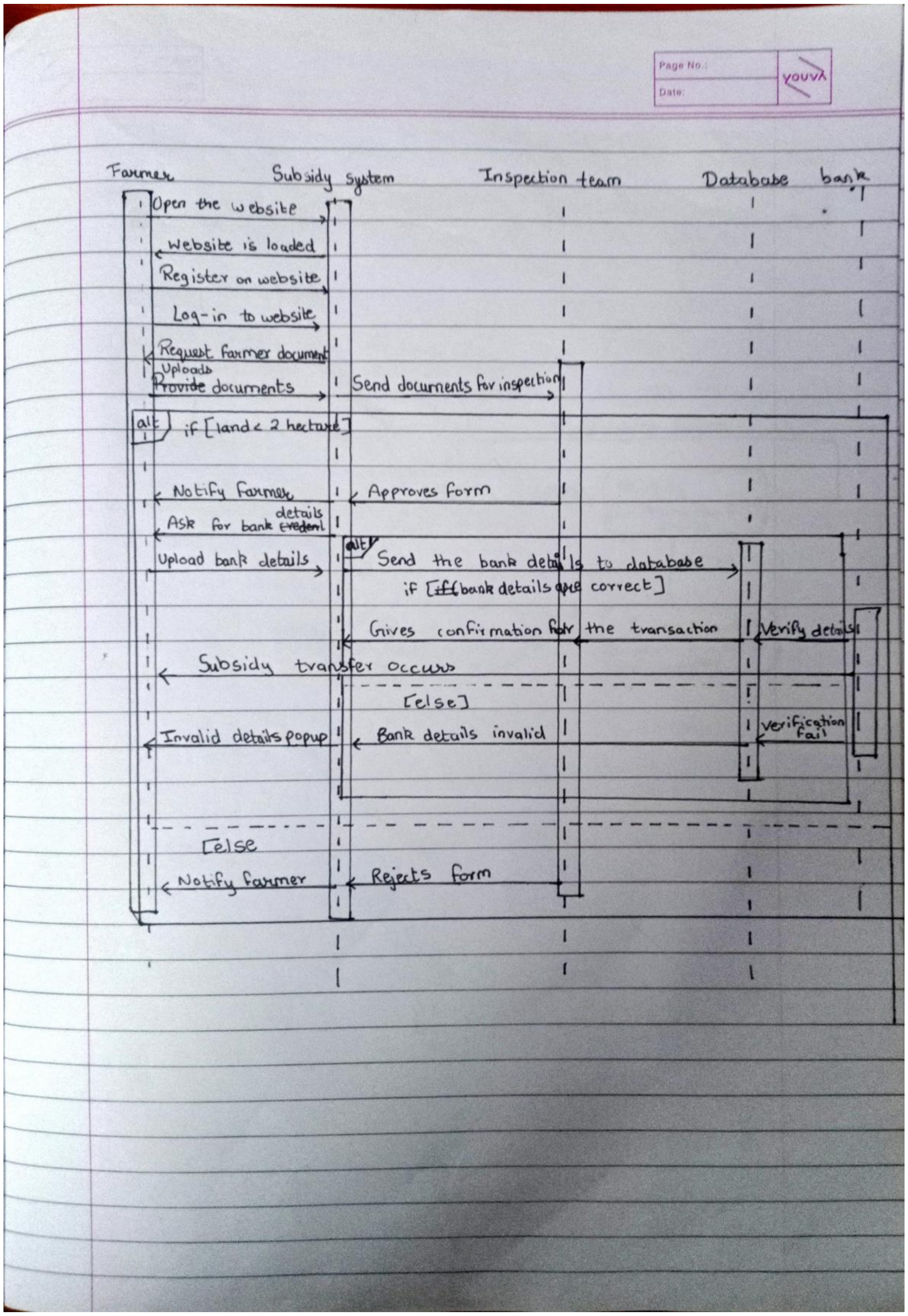
Its specific purpose is to define the state changes triggered by events. Events are internal or external factors influencing the system. State chart diagrams are used to model the states and also the events operating on the system.

- **<u>Scenario:</u>**

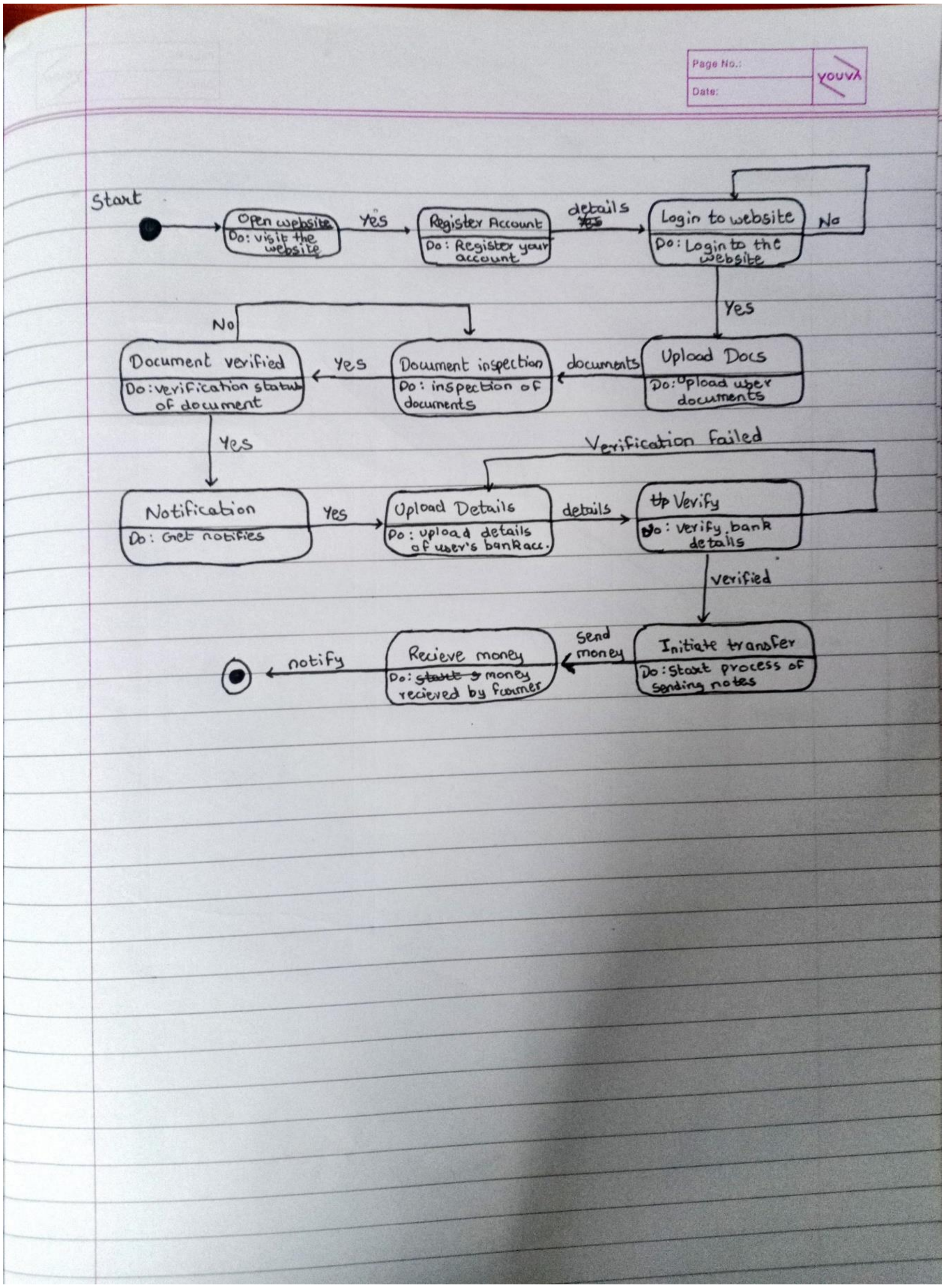
1. **Government wish to transfer subsidy to those farmer's bank account having land less than 2 hectares.**
    a. **Farmer needs to register themselves on a provided website.**
    b. **Farmer needs to upload required documents.**
    c. **Inspection team inspects farmer document and land.**
    d. **Government successfully transfers the subsidy in farmer's bank account.**

- **Questions:**

1. **Draw Sequence diagram for given scenario.**



2. Draw State diagram for given scenario.



3. Comparison between sequence diagram and state diagram.

- A sequence diagram generally shows the execution of a particular use case for the application of and the objects that are involved in carrying out that use case.
- A state diagram shows the various states that are valid for an object. That could be a particular functionality or the system as a whole.

- A sequence diagram is aimed at one specific function, example, fetching a document from database and printing it. A state diagram.
- A state diagram on the other hand can show a model for the whole system.

- **Conclusion:**

Hence, by performing this experiment I learnt about the state diagram and the sequence diagram, and their application. I also created sequence diagram and state diagram for the scenario – ‘Government wish to transfer subsidy to farmer’s bank with less than 2 hectares of land.’

(10)	(20)	(10)	(10)	TOTAL

EXPERIMENT NO:07

Aim: Draw E-R diagram, DFD and create data dictionary for above system.

Theory:

- **What is Entity?**

An entity is any object in the system that we want to model and store information about. Entities are usually recognizable concepts, either concrete or abstract, such as person, places, things, or events which have relevance to the database.

Some specific examples of entities are Employee, Student, Lecturer.

- **What is E-R diagram?**

An entity relationship model, also called an entity - relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems.

For example: In the following ER diagram we have - two entities Student and College and these two entities have many to one relationship as many student's study in a single college.

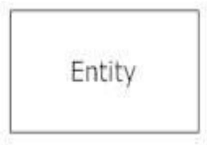
- **ER Diagram Uses:**

 When documenting a system or process, looking at the system in multiple ways increases the understanding of that system.

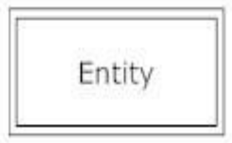
ERD diagrams are commonly used in conjunction with a data flow diagram to display the contents of a data store.

- **Common Entity Relationship Diagram Symbols:**

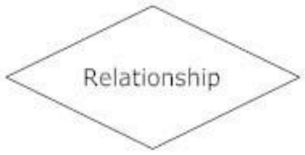
 Entities, which are represented by rectangles. An entity is an object or concept about which you want to store information.



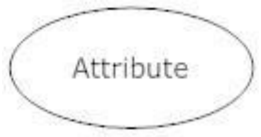
- **weak entity** is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.



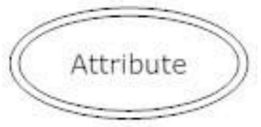
- **Relationship**, which are represented by diamond shapes, show how two entities share information in the database.



- **Attributes**, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity.



- **multivalued attribute** can have more than one value. For example, an employee entity can have multiple skill values.

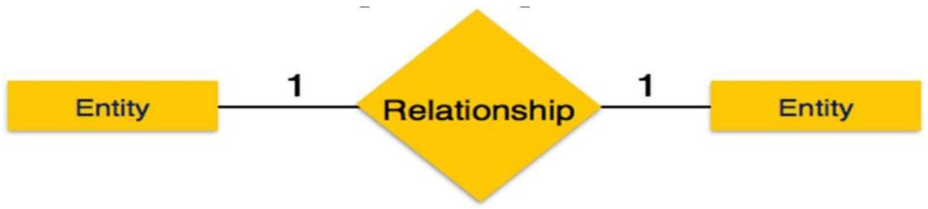


- **Connecting lines**, solid lines that connect attributes to show the relationships of entities in the diagram.

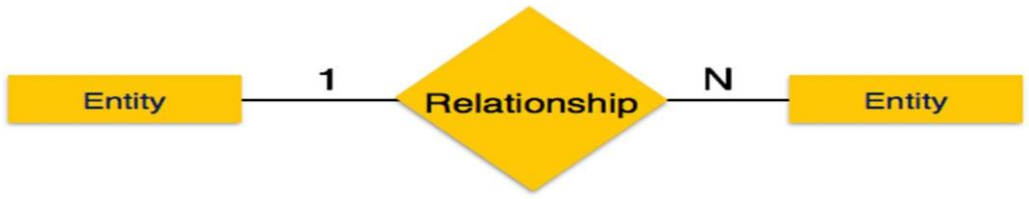
- **Relationship**

  Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

  - <u>**One-to-one**</u> − When only one instance of an entity is associated with the relationship, it is marked as '1:1'. The following image reflects that only one instance of each entity should be associated with the relationship. It depicts one-to-one relationship.

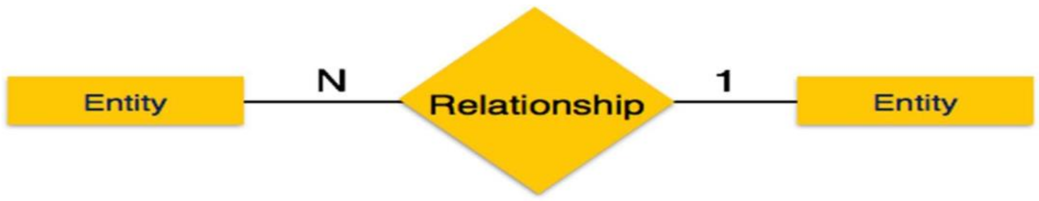


- **One-to-many** − When more than one instance of an entity is associated with a relationship, it is marked as '1: N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many

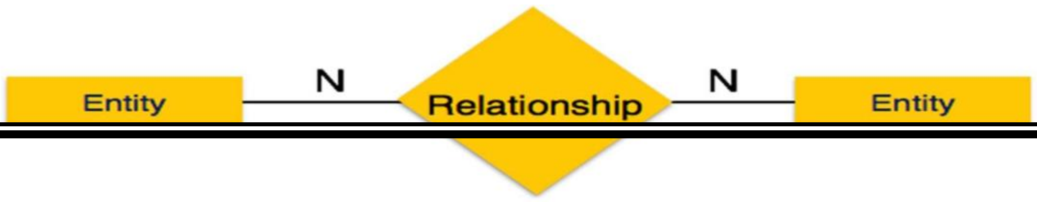


relationship.

- **Many-to-one** − When more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship



- **Many-to-many** − The following image reflects that more than one instance of an entity on the left and more than one

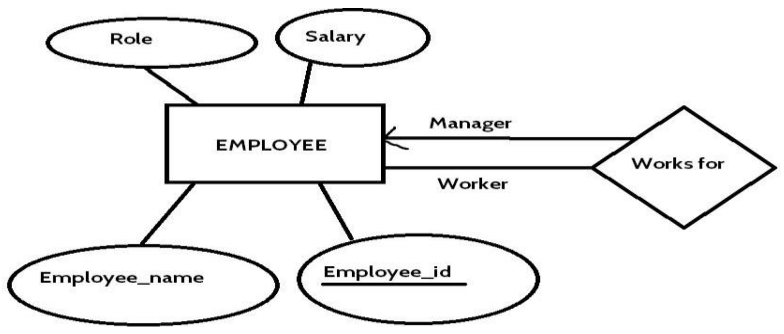


instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.

- **How to Draw ER Diagrams?**

Below points show how to go about creating an ER diagram.

1. Identify all the entities in the system. An entity should appear only once in a particular diagram. Create rectangles for all entities and name them properly.
2. Identify relationships between entities. Connect them using a line and add a diamond in the middle describing the relationship.
3. Add attributes for entities. Give meaningful attribute names so they can be understood easily



Advantages of ER Diagram:

- Conceptually it is very simple: ER model is very simple because if we know relationship between entities and attributes, then we can easily draw an ER diagram.
- Better visual representation: ER model is a diagrammatic representation of any logical structure of database. By seeing ER diagram, we can easily understand relationship among entities and relationship.

- Effective communication tool: It is an effective communication tool for database designer.
- Highly integrated with relational model: ER model can be easily converted into relational model by simply converting ER model into tables.

- Easy conversion to any data model: ER model can be easily converted into another data model like hierarchical data model, network data model and so on.

Disadvantages of ER Diagram:

- Limited constraints and specification.
- Loss of information content: Some information be lost or hidden in ER model.
- Limited relationship representation: ER model represents limited relationship as compared to another data models like relational model etc.
- No representation of data manipulation: It is difficult to show data manipulation in ER model.
- Popular for high level design: ER model is very popular for designing high level design

- **What is Data Flow Diagram?**

  Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

  Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

- **DFD Symbols:**

There are four basic symbols that are used to represent a data-flow diagram.

- **<u>Process</u>**

  A process receives input data and produces output with a different content or form. Processes can be as simple as collecting input data and saving in the database, or it can be complex as producing a report containing monthly sales of all retail stores in the northwest region.

  Every process has a name that identifies the function it performs. The name consists of a verb, followed by a singular noun.

  <u>Example:</u>

  - Apply Payment
  - Calculate Commission
  - Verify Order

- **<u>Data Flow</u>**

  A data-flow is a path for data to move from one part of the information system to another. A data-flow may represent a single data element such the Customer ID or it can represent a set of data element (or a data structure).

  Example:

  - Customer info (Last Name, FirstName, SS#, Tel #, etc.)
  - Order info (Ordered, Item#, Order Date, Customer, etc.).

- **<u>Data Store</u>**

  A data store or data repository is used in a data-flow diagram to represent a situation when the system must retain data because one or more processes need to use the stored data in a later time.

- **<u>External Entity</u>**

  It Is also known as actors, sources or sinks, and terminators, external entities produce and consume data that flows between the entity and the system being diagrammed. These data flows are the inputs and outputs of the DFD.

- **How to draw a data flow diagram?**

  Lucid chart makes it easy to create a customized data flow diagram starting with a simple template. Choose the symbols you need from our library—processes, data stores, data flow, and external entities—and drag-and-drop them into place. Since Lucid chart is an online tool, it facilitates collaboration and bypasses the hassles of desktop DFD software.

- **Levels in Data Flow Diagrams (DFD)**

  In Software engineering DFD (data flow diagram) can be drawn to represent the system of different levels of abstraction. Higher level DFDs are partitioned into low levels-hacking more information and functional elements. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see mainly 3 levels in data flow diagram, which are: 0- level DFD, 1-level DFD, and 2-level DFD.

0- level DFD:

It is also known as context diagram. It’s designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as single bubble with input and output data indicated by incoming/outgoing arrows.

1- level DFD:

In 1-level DFD, context diagram is decomposed into multiple bubbles/processes.in this level we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses.

2- level DFD:

2- level DFD goes one step deeper into parts of 1-level DFD.It can be used to plan or record the specific/necessary detail about the system's functioning.

▪ **Advantages of data flow diagram:**

- A simple graphical technique which is easy to understand.
- It helps in defining the boundaries of the system.
- It is useful for communicating current system knowledge to the users.
- It is used as the part of system documentation file.
- It explains the logic behind the data flow within the system.

▪ **Disadvantages of data flow diagram:**

- Data flow diagram undergoes lot of alteration before going to users, so makes the process little slow.
- Physical consideration is left out. It makes the programmers little confusing towards the system.

- **What is Data Dictionary?**

  A data dictionary contains metadata i.e. data about the database. The data dictionary is very important as it contains information such as what is in the database, who is allowed to access it, where is the database physically stored etc. The users of the database normally don't interact with the data dictionary, it is only handled by the database administrators.

  The different types of data dictionary are:

- **<u>Active Data Dictionary</u>**

  If the structure of the database or its specifications change at any point of time, it should be reflected in the data dictionary. This is the responsibility of the database management system in which the data dictionary resides.

- **<u>Passive Data Dictionary</u>**

  This is not as useful or easy to handle as an active data dictionary. A passive data dictionary is maintained separately to the database whose contents are stored in the dictionary. That means that if the database is modified the database dictionary is not automatically updated as in the case of Active Data Dictionary.

- **Creating the Data Dictionary**

  When you use the Database Configuration Assistant to create a database, Oracle automatically creates the data dictionary. Thereafter, whenever the database is in operation, Oracle updates the data dictionary in response to every DDL statement.

  The data dictionary base tables are the first objects created in any Oracle database. They are created in the system tablespace and must remain there. The data dictionary base tables store information about all user-defined objects in the database.

- **Advantages of data Dictionary:**

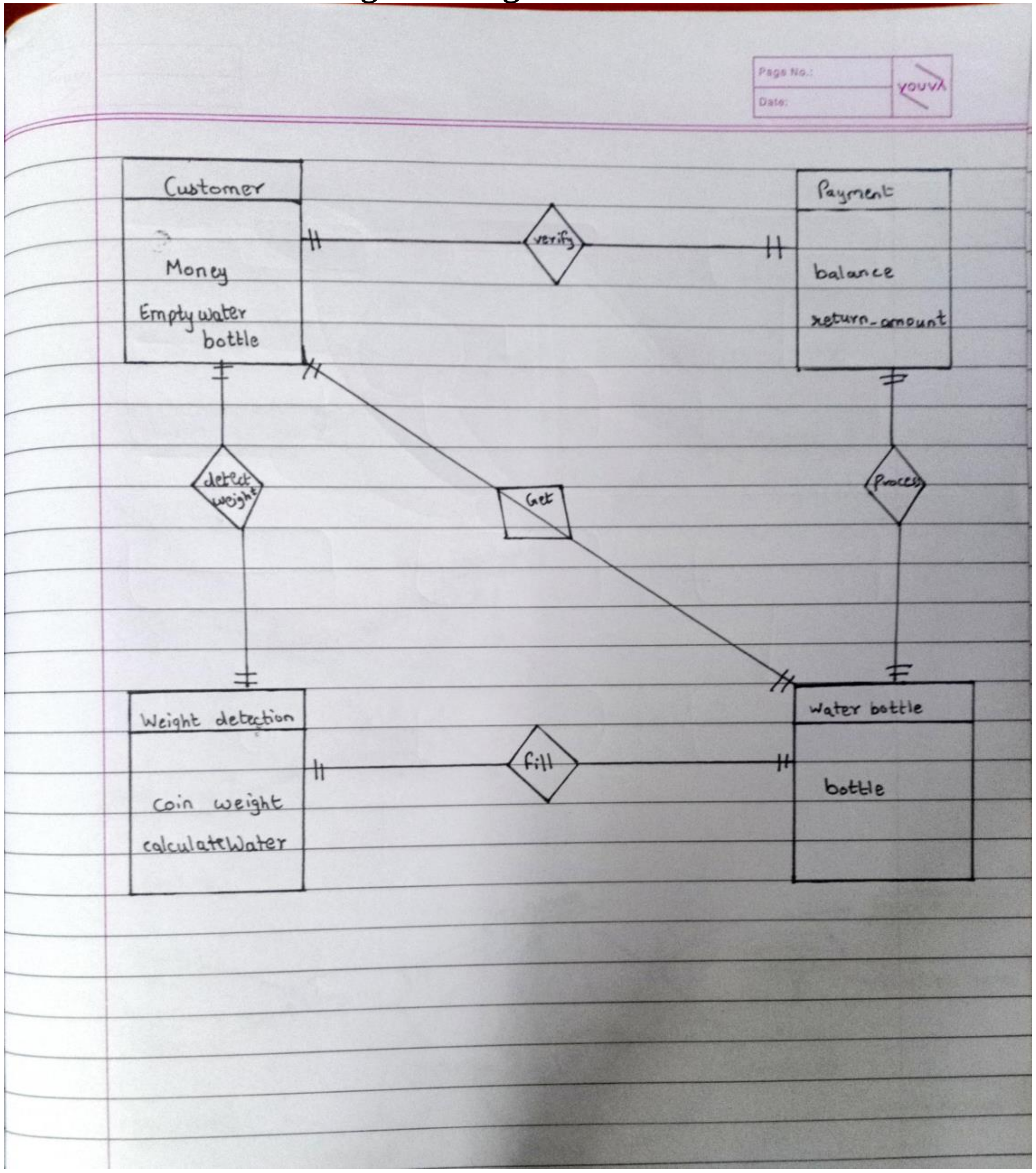
  There are a number of advantages of using Data Dictionary in computer system analysis and design. The advantages are: consistency, clarity; reusability; completeness; increase in sharing and integration; and ease of use for the developer.

- **Scenario:**

    1. **Library Management System**
    2. **Water Vending Machine**

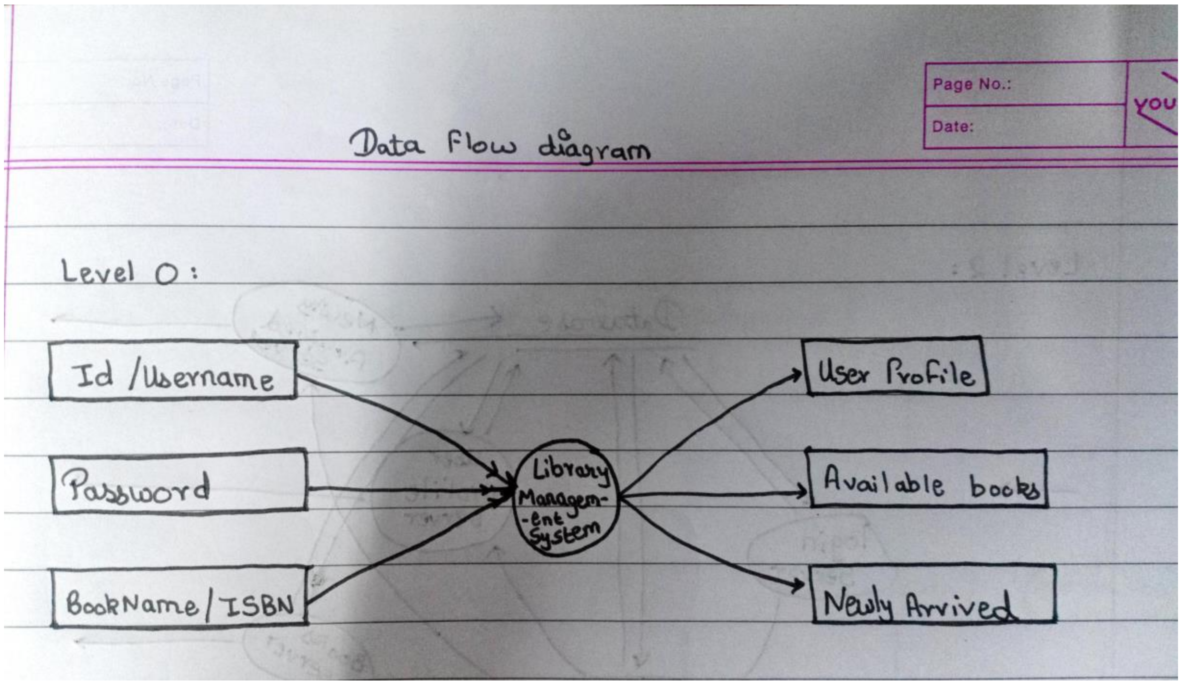
- **Questions:**

    1. **Draw the ER diagram for given scenario 2?**

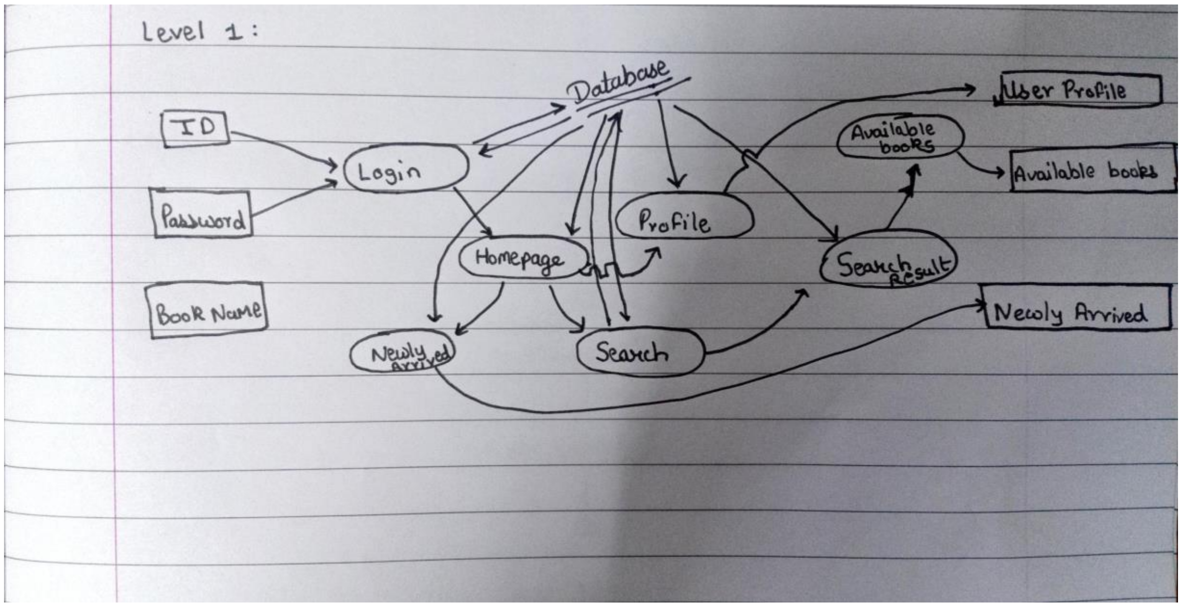


2. Draw DFD diagram for given scenario 1?

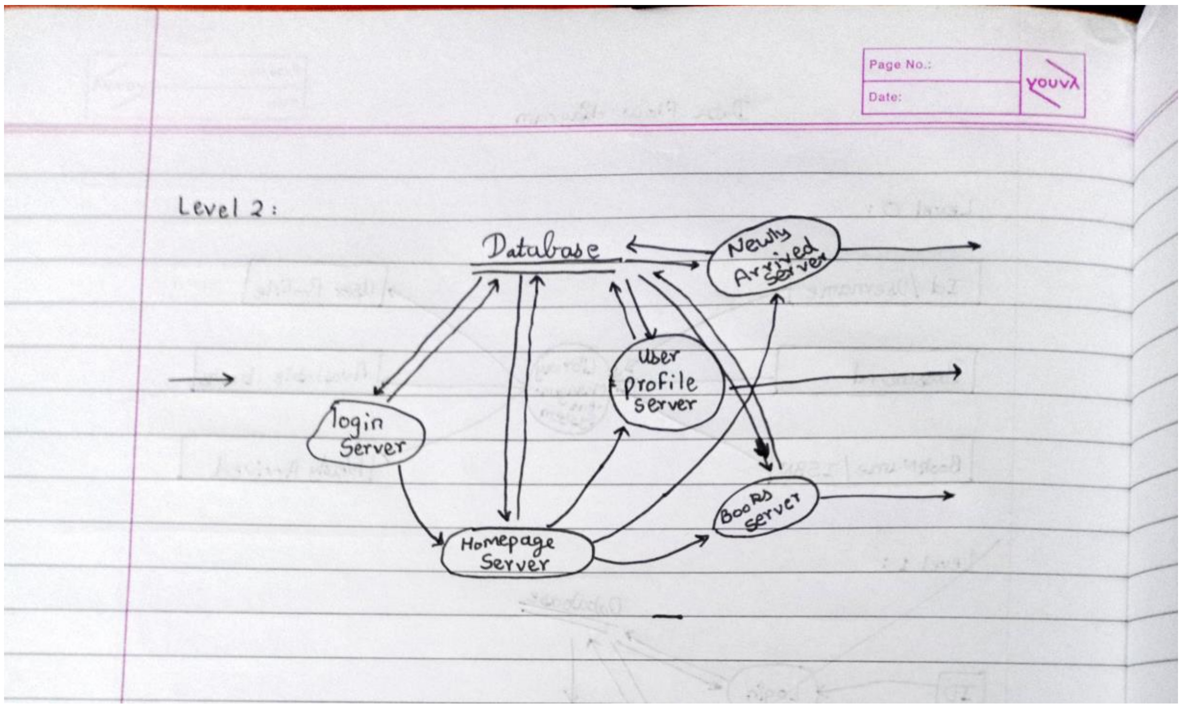
Level 0:



Level 1:



Level 1:


Page No.:
Date:
Level 2:
Database
Newly Arrived Server
User profile server
login Server
Homepage Server
Books server

3. Create data dictionary for given scenario 2?

Data Dictionary

Name	Data type	Max Fields	Required
Money	Integer	99999	Yes
Water	Float	100.0	Yes
Coin weight	Float	1000.0	Yes
Calculate_water()	Float	10litres	Yes
Balance	Float	-	No
Water_bottle	Integer	-	Yes

4. **What is the objective of maintaining data dictionary?**

Data dictionaries are used to provide detailed information about the contents of a dataset or database, such as the names of measured variables, their data types or formats, and text descriptions.

This will provide a document that is consistently formatted and contains what is needed for others to understand your data

Conclusion:

Hence, by performing this practical, I learnt about the concepts of data dictionary, E-R diagram, and DFD, their uses and application. I also drew ER diagram, data flow diagram, and data dictionary for Library management system and water vending machine

(10)	(20)	(10)	(10)	TOTAL

EXPERIMENT NO. 8

Aim: Draw activity diagram for above system.

Theory:

- **Activity diagram:**
  - Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.
  - Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.
  - The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc
  - The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity Diagram

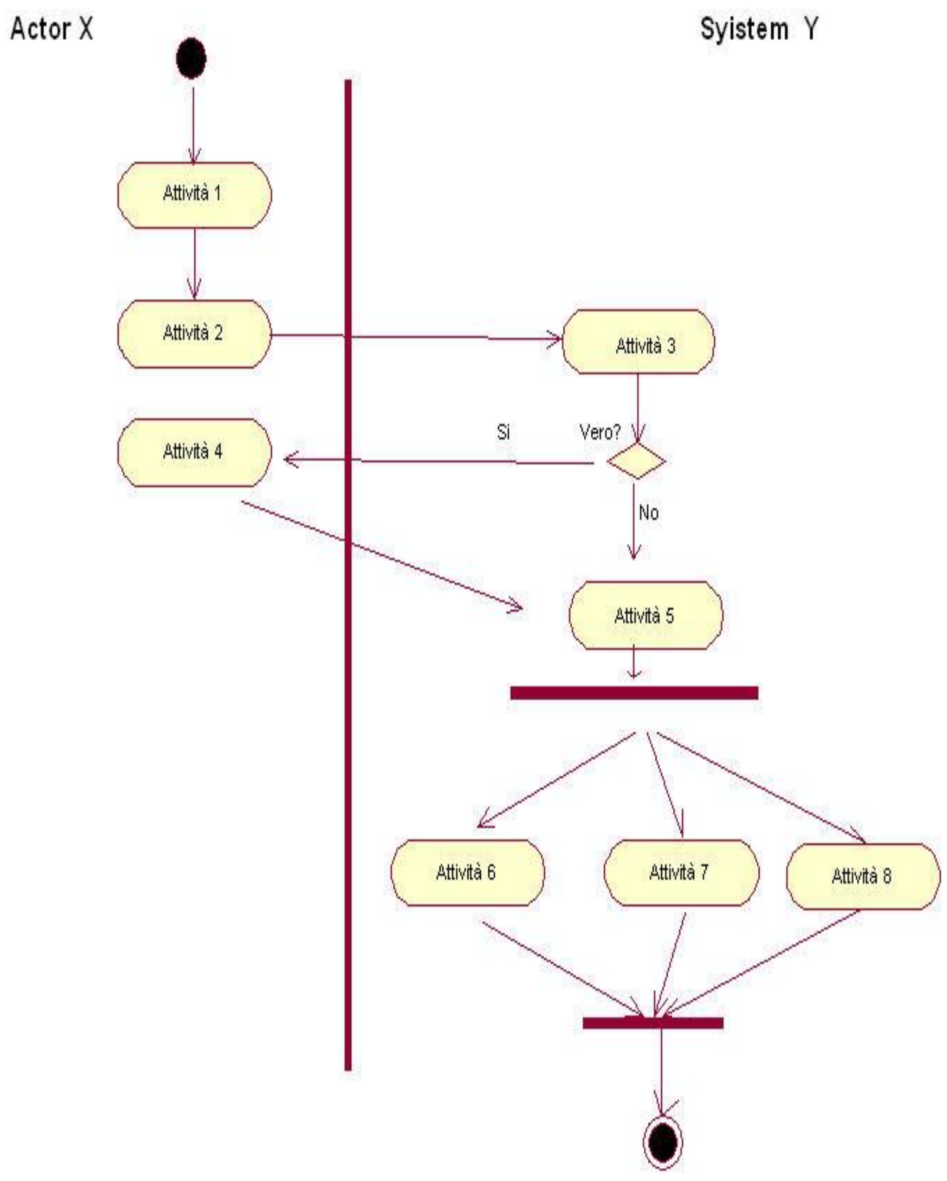


Fig: Activity Diagram

The purpose of an activity diagram can be described as −

- The activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

Activity diagram can be used for −

- Modelling work flow by using activities.
- Modelling business requirements.
- High level understanding of the system's functionalities.
- Investigating business requirements at a later stage.

- **Scenario: New online bank account opening and management system**

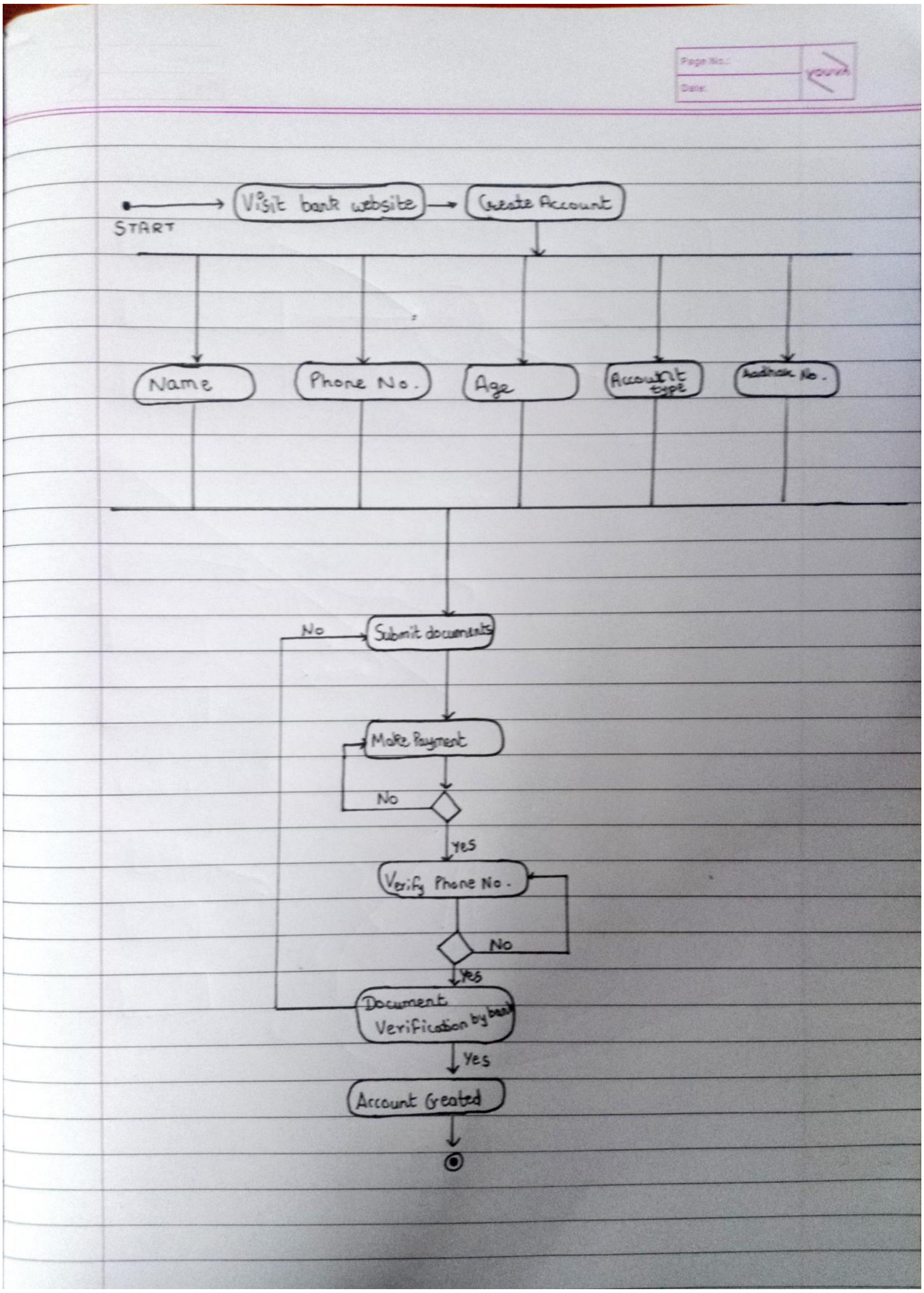
- **Questions:**

1) When to Use Activity Diagram?

Activity diagram can be used in the following scenarios:

- Modelling work flow by using activities.
- Modelling business requirements.
- High level understanding of the system’s functionalities.
- Investigating business requirements at a later stage.

2) Draw activity diagram for given scenario.



3) Comparison between activity diagram and sequence diagram.

Sequence Diagram	Activity Diagram
The sequence diagram represents the UML, which is used to visualize the sequence of calls in a system that is used to perform a specific functionality. The sequence diagram shows the message flow from one object to another object.	The activity diagram represents the UML, which is used to model the workflow of a system. The Activity diagram shows the message flow from one activity to another.
Main focus is the interaction between different objects over a specific period of time.	Main focus is the flow of activities.
Helps to visualize the sequence of calls in a system to perform a specific functionality.	Helps to model the workflow a system.
Sequence diagram issued for the purpose of dynamic modelling. Sequence diagram is mainly used to represent the time order of a process.	Activity diagram is used for the purpose of functional modelling. Activity diagram is used to represent the execution of the process.

- **Conclusion:**

Hence, by performing this practical I got to know about the activity diagrams, its various components, and its use. I also drew an activity diagram for the scenario – 'New online bank account opening and management system.'

10	20	10	10	Total