3. System Design

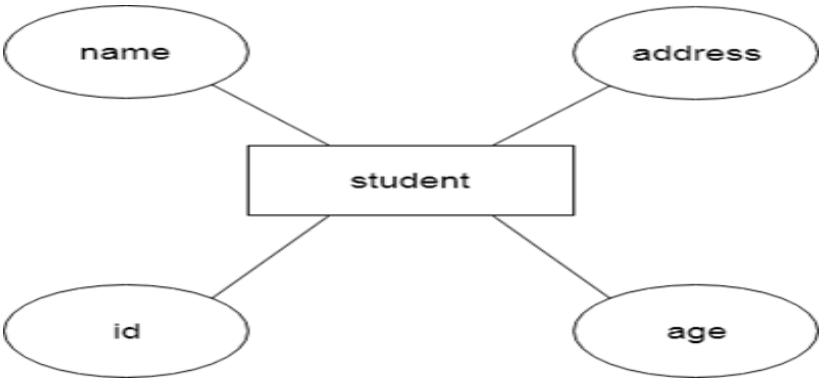
3.1 E-R Diagram

An Entity-Relationship (ER) model illustrates the structure of a database using a visual representation known as an Entity-Relationship Diagram (ER Diagram). This model serves as a blueprint for designing the database schema and capturing the relationships between different entities and attributes.

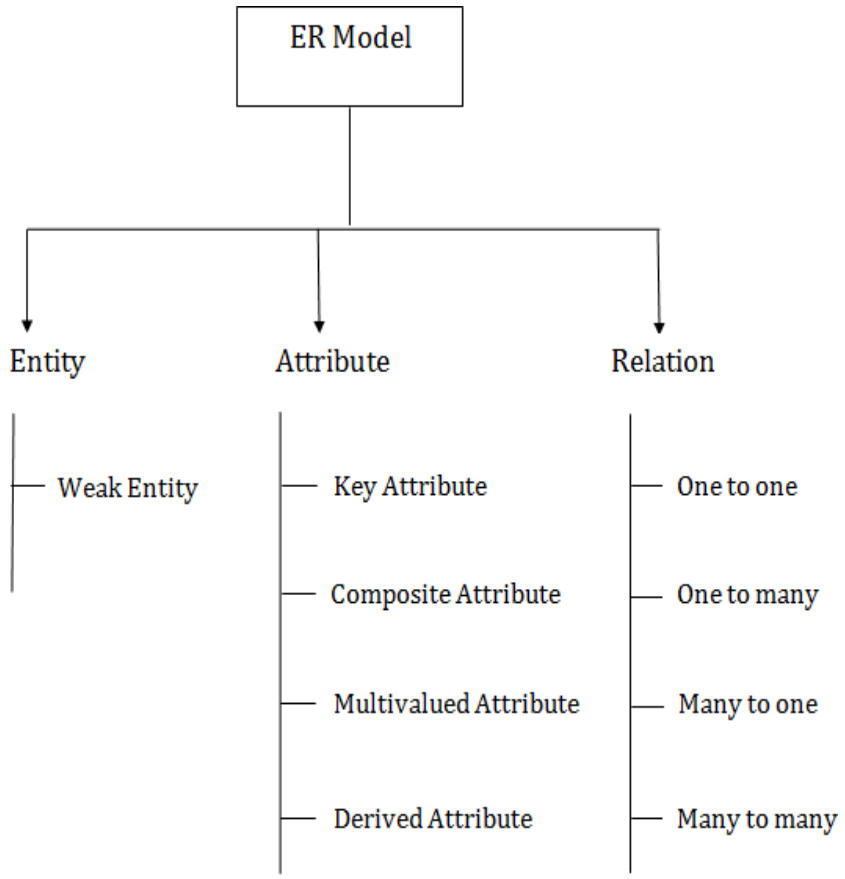
The ER model provides a systematic approach to organizing and conceptualizing the data within a database system. It represents entities as well as the relationships between them, helping to clarify how data elements are connected and organized.

3.1.1 ER model

- The Emergency Room model corresponds to an Entity-Relationship model, serving as a high-level representation of data structures. It is utilized to illustrate the data components and relationships within a defined system.
- It establishes a structured framework for the database. Moreover, it provides a straightforward and easily understandable perspective on the data.
- In Entity-Relationship modeling, the organizational database structure is depicted through a design known as an Entity-Relationship diagram.
- For instance, consider designing a school database. An educational record could be represented as an entity with attributes such as name, ID, age, etc. Similarly, the address could be another entity with attributes like city, street name, zip code, etc., and there would be a relationship between them.



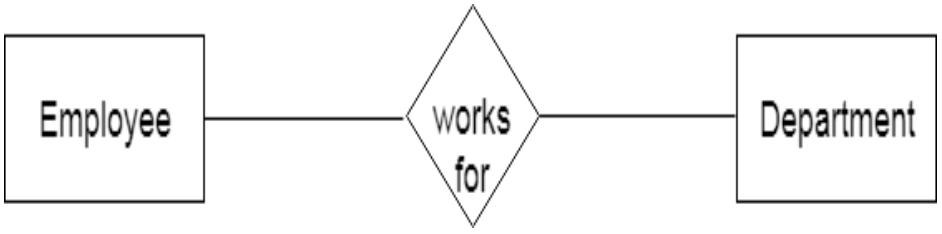
Component of ER Diagram



1. Entity

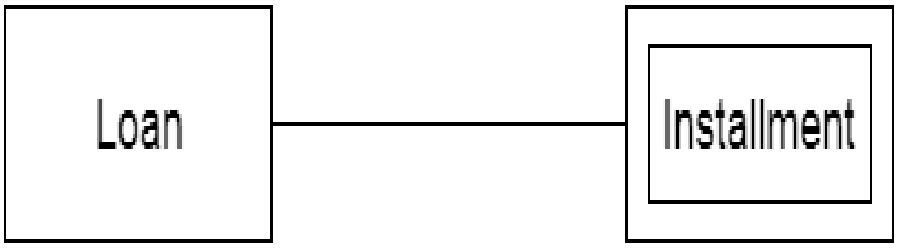
A substance may be anything, class, individual or spot. In the ER frame, a substance can be tended to as square shapes.

Think about a relationship as a delineation chief, thing, specialist, office, etc can be taken as a substance.



2. Powerless Entity

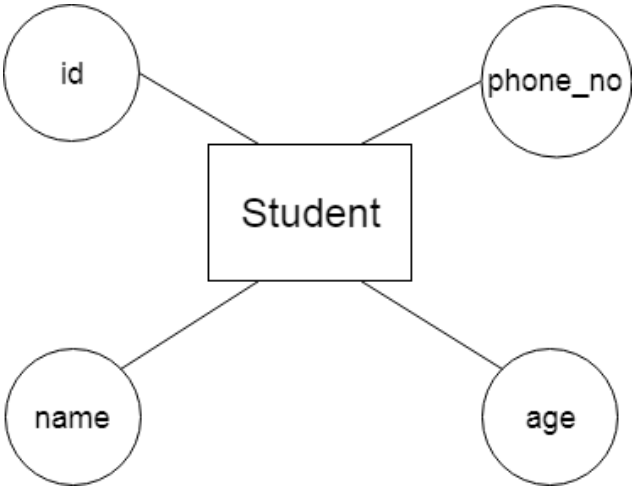
A substance that depends upon another component called a frail substance. The frail element contains no critical trait of its own. The feeble substance is addressed by a twofold square shape.



Characteristic

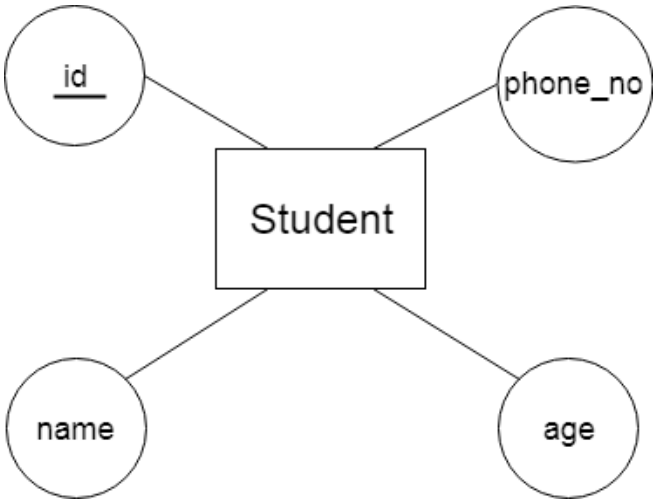
The quality is utilized to depict the property of a section. Obscure is utilized to address a quality.

For example, id, age, contact number, name, etc can be attributes of a student.



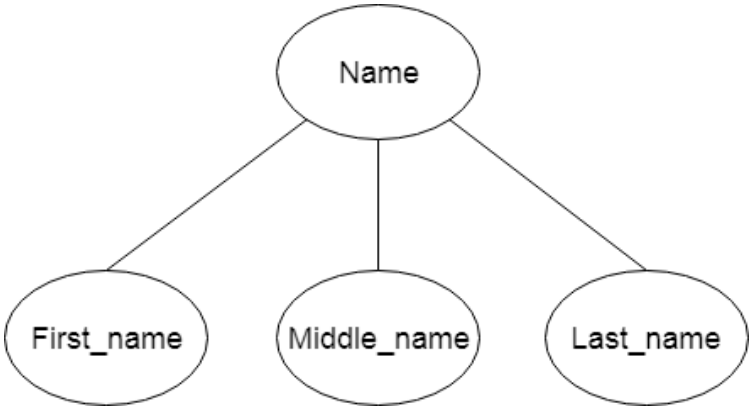
a. Key Attribute

The key quality is used to address the essential ascribes of a substance. It tends to a fundamental key. The key property is tended to by a circle with the text underlined**.**



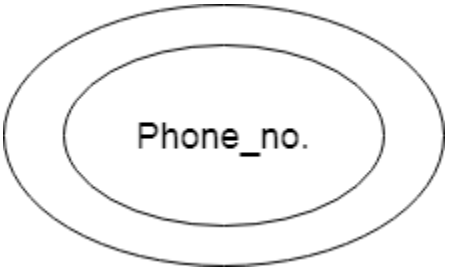
b. Composite Attribute

A property that made from various attributes is known as a composite quality. The composite trademark is tended to by an oval, and those circles are related with a circle.



c. Multivalued Attribute

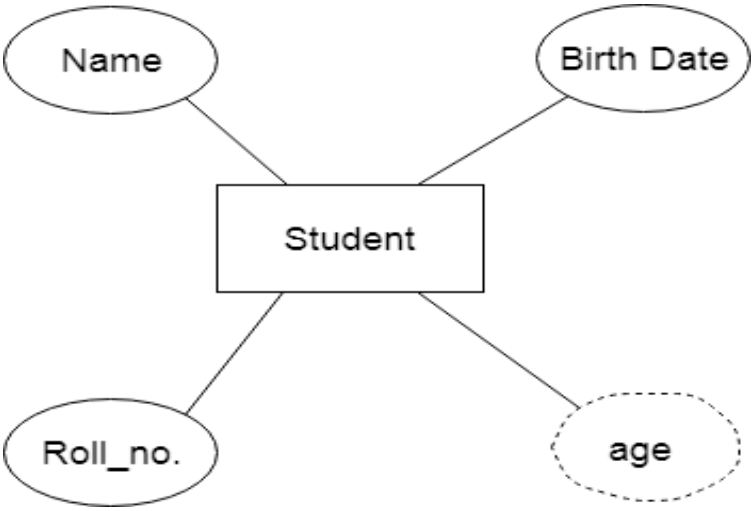
A quality can have more than one worth. These qualities are known as a multivalued property. The twofold oval is used to address multivalued property. For example, a student can have more than one phone number.



d. Determined Attribute

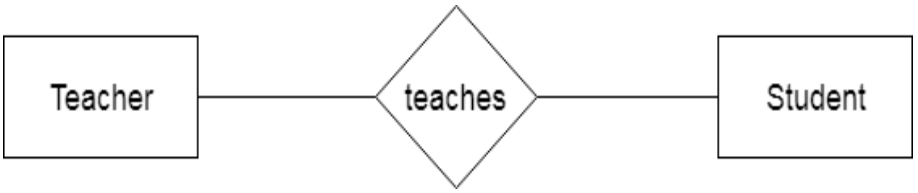
A property that can be gotten from another quality is known as a decided attribute. It will in general be tended to by a ran circle.

For example, a singular's age changes long term and can be gotten from one more quality like Date of birth.



3. Relationship

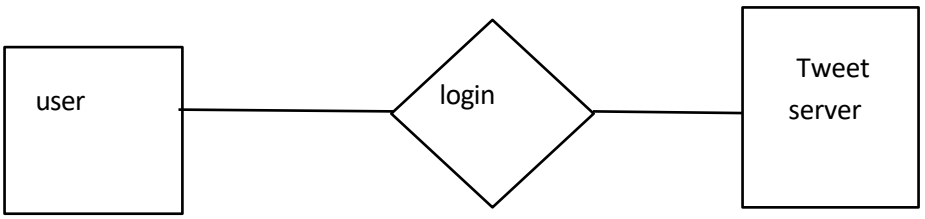
A relationship is used to depict the connection between substances. Important stone or rhombus is utilized to address the relationship.



Sorts of relationship are as per the following

a. One-to-One relationship

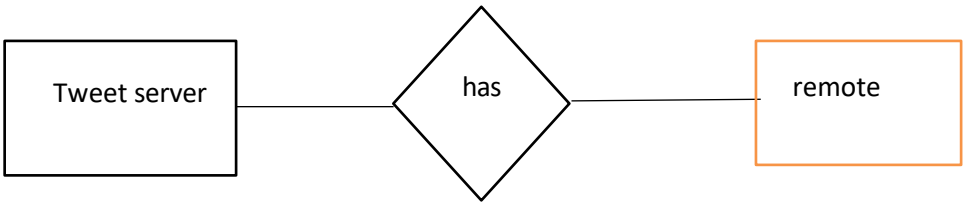
At the point when just a single instance of a component is connected with the relationship, then it is known as facilitated relationship. For instance, A female can wed to one male, and a male can wed to one female.



b. One-to-many relationship

Exactly when simply a solitary illustration of the substance on the left, and more than one event of a component on the right associates with the relationship then this is known as a one-to-various connections.

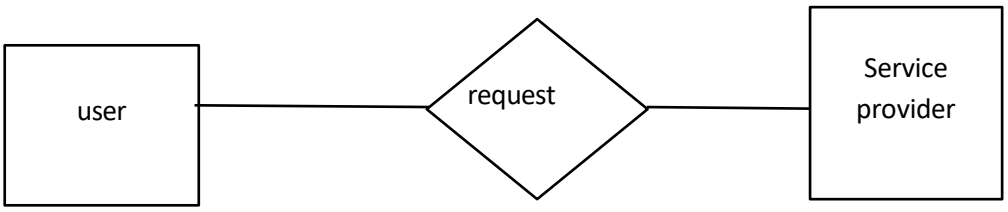
For example, Scientist can envision various manifestations, but the improvement is done by the really express analyst.



c. Many-to-one relationship

Exactly when more than one event of the component on the left, and simply a solitary event of a substance on the right associates with the relationship then it is known as a many-to-one relationship.

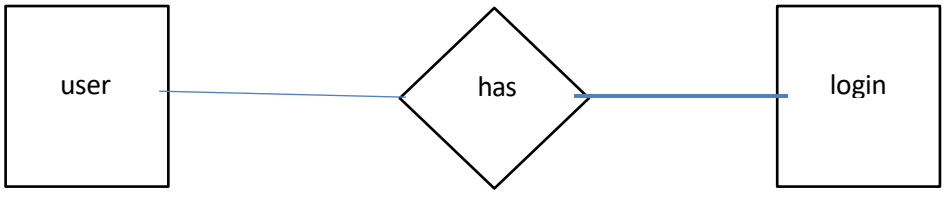
For example, Student enols for only a solitary course, but a course can have various students.



d. Many-to-many relationship

At the point when more than one event of the substance on the left, and more than one event of a component on the right associates with the relationship then it is known as a many-to-various connections.

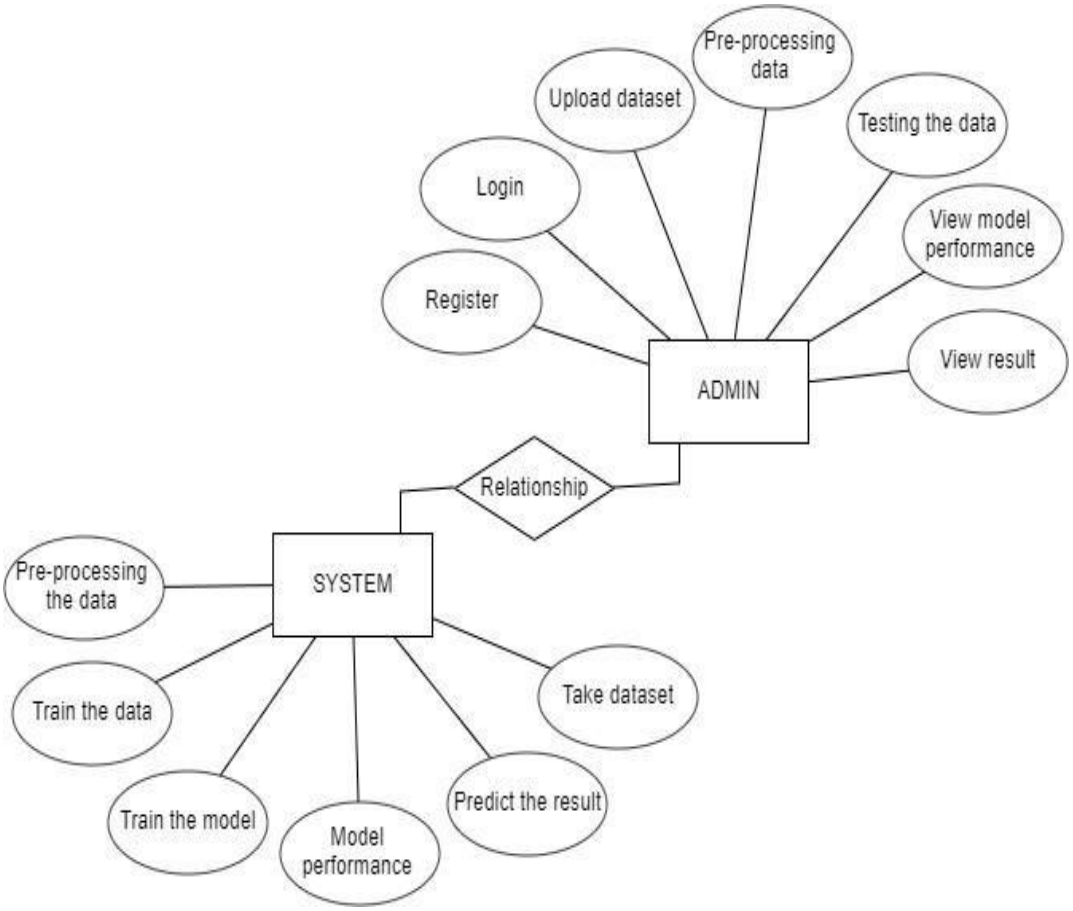
For example, Employee can allot by numerous exercises and project can have various specialists.



E-R Diagram

The ER diagram consists of entities such as User, Blockchain, Transaction, Digital Signature, Private Key, and Public Key. User generates Digital Signature using Private Key. Digital Signature is used to sign Transaction. Transaction is verified on Blockchain using Public Key. Blockchain stores Transaction and Digital Signature.

This ER diagram illustrates the key entities and relationships involved in using a new digital signature primitive in a blockchain application.



An Entity-Relationship (ER) Diagram is a graphical representation of a database structure. It visually depicts how entities (objects or concepts**)** are related to each other in a database system. ER diagrams are widely used in database design to model data relationships before actual implementation.

Components of an ER Diagram Entities

Represent objects or things in the system Depicted as rectangles in the ER diagram.

Attributes

Describe properties of an entity (e.g., "User ID", "Age", "Activity Type"). Represented as ovals connected to their entities.

Relationships

Show how entities are connected (e.g., "User performs Activity"). Represented as diamonds between entities.

Primary Key (PK)

A unique identifier for each record in an entity (e.g., "User ID" in the Elderly User table). Underlined in the ER diagram.

Cardinality

Defines the number of relationships between entities (e.g., "One Elderly User can have Multiple Activities"). Represented using symbols like (1:1, 1:M, M:N).

3.2. Data Dictionary

A Data Dictionary compiles names, definitions, and attributes concerning data elements utilized or stored within a database, information system, or part of a research project. It delineates the meanings and functions of data elements within the context of a project and offers guidance on understanding, recognizing meanings, and description. Additionally, a Data Dictionary offers metadata about data elements, aiding in defining the scope and attributes of data elements, as well as the guidelines for their usage and application.

Name	**Type**	**Collation**	**Null**	**Default**
id	Int(11)		No	None
User name	Varchar (30)	Latin_swedish_ci	No	None
email	Varchar (30)	Latin_swedish_ci	No	
pswd	Varchar (30)	Latin_swedish_ci	No	
phone no	Varchar (30)	Latin_swedish_ci	No	
country	Varchar (30)	Latin_swedish_ci	No	
state	Varchar (30)	Latin_swedish_ci	No	

Table no.3.2.1 Data Dictionary

Id	User name	email	pswd	phno	country	state	city
1	vijay	vkappati@gmail.com	********	6281407028	India	AP	kurnool
2	vinod	vinod@gmail.com	********	8790557392	India	AP	kurnool
3	Ajay	ajay@gmail.com	********	7780490892	India	AP	mdk
4	vishnu	ramu@gmil.com	********	9849466191	India	AP	rzp

Table no 3.2.2 Data Table

Id	uname	ureview	sanalysis	tname	suggestion
1	Kumar	The traffic for ambulance bad	False positive	Traffic	Better to avoid traffic
2	kumar	Exam System is good fees students	False Negative	Exam	To know about exam
3	Vinod	We can find more details in too and can view all	Malicious Bots	Sexual_assults	To know about sexual assaults
4	Manjunath	This is excellent	False Negative	Carona_vacine	Consume all people
5	Chinni	Invention if vaccine comes to	False Negative	IPL	Provide more information

Table no 3.2.3 Received User Data Analysis

3.3 Normalization

Normalization is the primary method for optimizing data in a database to fulfill two essential criteria.

Data dependencies are logical, ensuring that all related data items are stored together. Normalization is crucial for various reasons, primarily because it enables databases to occupy minimal disk space, resulting in enhanced performance.

Normalization is also referred to as data standardization.

The three primary types of normalization are outlined below. Note: "NF" stands for "normal form."

First typical structure (1NF)

Tables in 1NF should comply with certain standards:

1. Every cell should contain just a solitary (nuclear) esteem.
2. Each part in the table ought to be astoundingly named.
3. All characteristics in a part ought to connect with a comparative region.

User ID	User name	Password
015	John	********
016	Princess	********
027	Tom	********
028	Claire	********
029	Robert	********

Table no 3.3.1 1NF

Second typical structure (2NF)

Tables in 2NF ought to be in 1NF and not have any most of the way dependence (e.g., each non-prime quality ought to be dependent upon the table's fundamental key).

User Id	Received Data through IOT	pswd	Login
1	11	********	Sign_up
2	12	********	Sign_up
3	13	********	Sign_up
4	14	********	Sign_up
5	15	********	Sign_up

Table no3.3.2 2NF

Third ordinary structure (3NF)

Tables in 3NF ought to be in 2NF and have no transitive reasonable circumstances on the fundamental key. The going with two NFs furthermore exists anyway are only here and there used.

ID	NAME	EMAIL	STATE	CITY	COUNTRY
11	Vijay	vijay@gmail.com	AP	RZP	INDIA
12	Vinod	vinod@gmail.com	AP	RZP	INDIA
13	Ramu	Ramu@gmail.com	AP	RZP	INDIA
14	Vishnu	vishnu@gmail.com	AP	RZP	INDIA

Table no 3.3.3 User Detail

User Details

USER ID	PASSSWORD	LOGIN
server	********	Sign_up
Vijay	********	Sign_up

Table no3.3.4 User details

Boyce-Codd Normal Form (BCNF)

Normalization is a critical process in database management aimed at organizing tables to minimize anomalies and ensure data integrity. It follows a series of stages known as normal forms.

Unnormalized Form (UNF): The initial state of a table where data is not organized according to any specific rules.

First Normal Form (1NF): In 1NF, each column contains atomic values, and there are no repeating groups or arrays within a row.

Second Normal Form (2NF): 2NF requires that every non-key attribute be fully functionally dependent on the primary key.

Third Normal Form (3NF): In 3NF, no transitive dependencies should exist, meaning that non-key attributes should not depend on other non-key attributes.

Elementary Key Normal Form (EKNF): EKNF is a further refinement of 3NF, emphasizing the use of elementary keys.

Boyce-Codd Normal Form (BCNF): BCNF addresses anomalies that may arise when multiple candidate keys exist. Fourth Normal Form (4NF): To achieve 4NF, a table must be in BCNF and should not have multi-valued dependencies.

Normal Form	Description
1NF	An alliance is in 1NF enduring it contains an atomic worth.
2NF	An association will be in 2NF expecting it is in 1NF and all non-key credits are totally down to earth ward on the fundamental key.
3NF	An alliance will be in 3NF enduring it is in 2NF and no change dependence exists.
BCNF	A more grounded importance of 3NF is known as Boyce Codd's common design.
4NF	An association will be in 4NF expecting it is in Boyce Codd's commonplace construction and has no multi-regarded dependence.
5NF	An association is in 5NF. In case it is in 4NF and contains no join dependence, joining should be lossless.

Benefits of Normalization

Reduction of data redundancy: Normalization helps eliminate redundant data by organizing it efficiently across tables. Improved overall database organization: By structuring data according to normalization rules, databases become more organized and easier to manage. Data consistency within the database.

Normalization ensures that data remains consistent across tables, reducing the risk of inconsistencies. More flexible database design

Normalization allows for more flexibility in database design, making it easier to accommodate changes and updates. Upholds the principle of data integrity: Normalization promotes data integrity by minimizing anomalies and ensuring accurate representation of data relationships.

Disadvantages of Normalization

Careless decomposition: If normalization is done without a clear understanding of user requirements, it can lead to excessive decomposition and unnecessary complexity in the database design.

Decreased performance: As tables are normalized to higher normal forms such as 4NF or 5NF, it may lead to decreased performance due to increased join operations and complexity in querying the database.

3.4 UML Diagrams

Introduction to uml

You may better understand, modify, manage, or record information about a system by using a UML diagram, which is a diagram based on the UML (Unified Modelling Language). A UML diagram shows the system's principal players, roles, actions, artifacts, or classes. Structured, behavioral, and interaction overview diagrams are the three main types of UML diagrams.

An analysis model may be expressed by a software engineer using the modeling notation that is regulated by a set of syntactic, semantic, and pragmatic principles in the unified modelling language. There are five separate viewpoints that may be used to depict a UML system, each offering a unique viewpoint on the system.

The Unified Modeling Language (UML) is built up from two distinct domains. UML Analysis modeling, which is concerned with the system's user model, and structural model modeling.

Modeling using the Unified Modeling Language (UML), with an emphasis on the perspectives of behavior, implementation, and the environment.

Uml Diagrams

The software business is always seeking new ways to automate software development, boost quality, decrease cost, and shorten time-to-market, all because software is becoming more crucial for many firms. Methods like as component technology, visual programming, patterns, and frameworks are examples of such approaches. As systems grow in breadth and size, businesses also look for ways to manage their complexity. The need of addressing common architectural issues such physical distribution, concurrency, replication, security, load balancing, and fault tolerance is particularly highlighted. The evolution of the World Wide Web has also contributed to the worsening of these architectural issues, even if it has simplified certain aspects.

A response to these demands was the Unified Modelling Language (UML). Systems design, in its simplest form, is the act of using UML diagrams to define a system's architecture, components, modules, interfaces, and data in order to meet certain criteria.

What is UML (Unified Modeling Language)?

UML (Unified Modeling Language) is a standardized modeling language used to visualize, design, and document software systems**.** It provides a set of diagrams to represent different aspects of a system, making it easier to understand and communicate system design

Why Use UML?

- Helps in software planning & design before actual coding Improves communication between developers, designers, and stakeholders. Enhances software maintainability by providing clear documentation. Supports object- oriented programming concepts.

Types of UML Diagram

UML diagrams are mainly classified into

In the project eight basic UML diagrams have been explained

- Use Case Diagram
- Class Diagram
- Sequence Diagram
- Collaboration Diagram
- Activity Diagram
- State Chart Diagram
- Component Diagram
- Deployment Diagram

Object Diagram: Represents system objects and their relationships.

Use Case Diagram: Represents user interactions with the system.

Class Diagram: Shows system classes, attributes, and relationships.

Sequence Diagram: Shows the flow of messages between system components.

Activity Diagram: Illustrates workflow or business processes.

State Diagram: Represents the use case diagram.

Component Diagram: Visualizes system components and dependencies.

Deployment Diagram: Shows hardware and software deployment.

Behavioral Diagrams (System Functionality & Workflow) These diagrams show how.

The UML diagrams are arranged into fundamental charts, social frameworks, and besides correspondence frame graphs. The diagrams are logically organized in the going with figure:

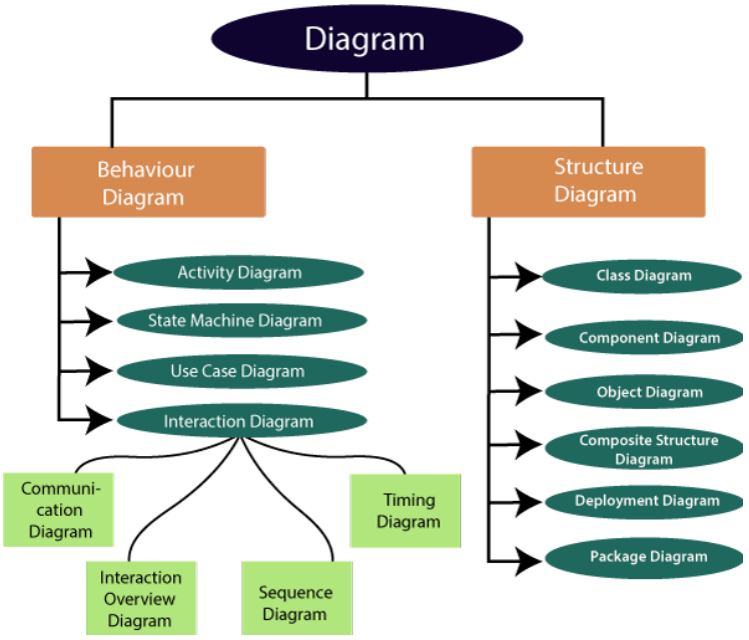


Fig.no 3.4 UML Diagram

Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram described and expressed using a Use-case analysis. Its objective is to present a graphical representation of the value provided by a system in terms of actors, their primary purpose of a use case diagram is to illustrate which system functions are performed for which actor.

Description: Use case diagram for admin and system

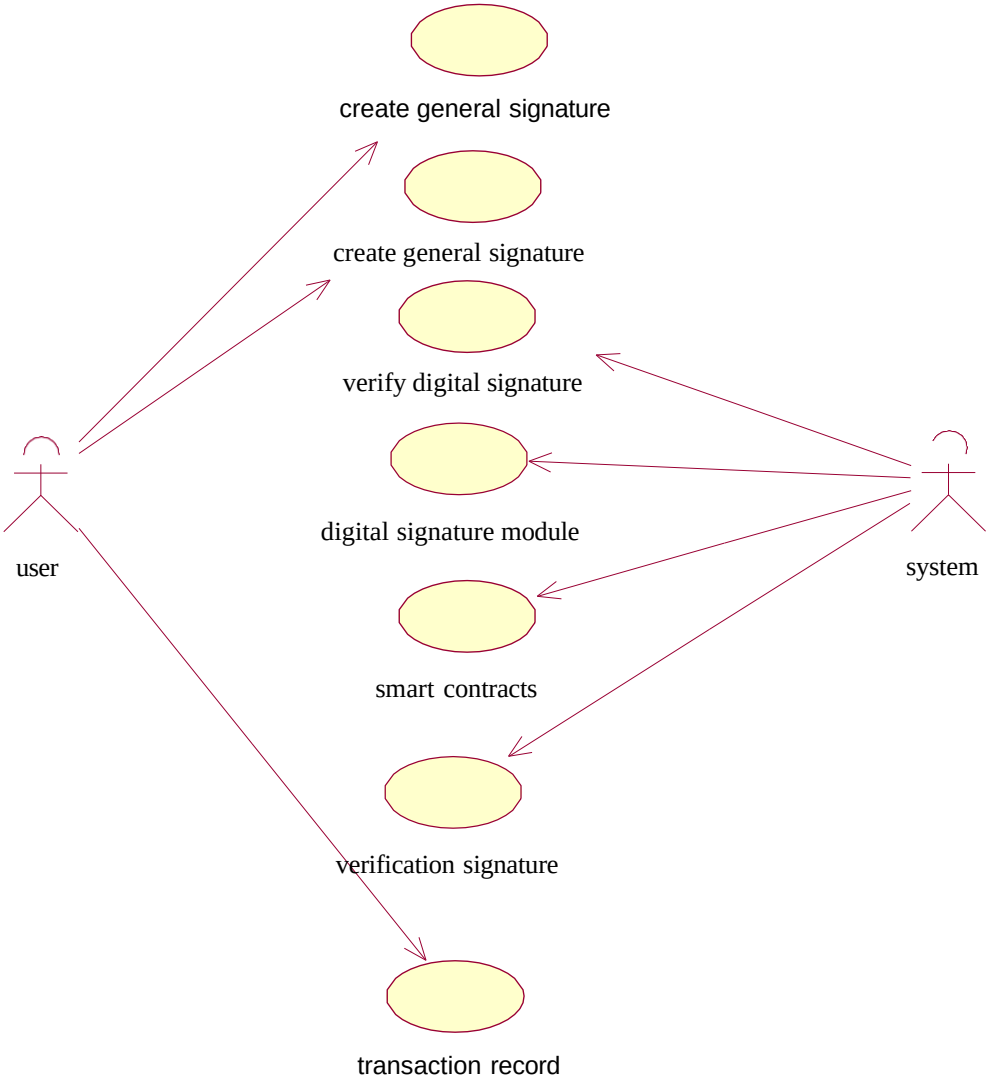


Fig.no 3.4.1 Use Case Diagram

Class Diagram

In computer programming, a class diagram in the Unified Modeling Language (UML) is a type of static structural diagram that illustrates the architecture of a system by displaying the structure's classes, their properties, operations (or methods), and the relationships among the classes. It delineates which class holds data.

Description: Class Diagram for User , Sensor data collector , DRl Model ,Path Planner and Navigation pn controler

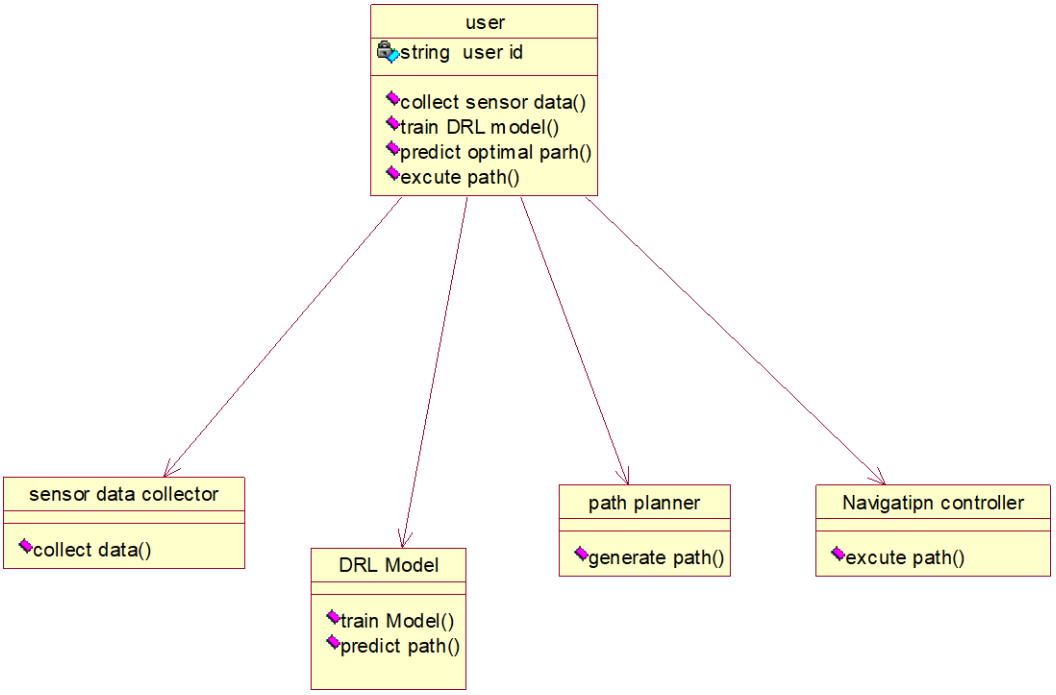


Fig.no 3.4.2 Class Diagram

Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a type of communication diagram that illustrates how processes interact with each other and the order in which they occur. It is a variation of a Message Sequence Chart. Sequence diagrams are sometimes referred to as event diagrams, event scenarios, or timing diagrams.

Description: Sequence diagram for user, key generator ,signature algorithm , sv and integration

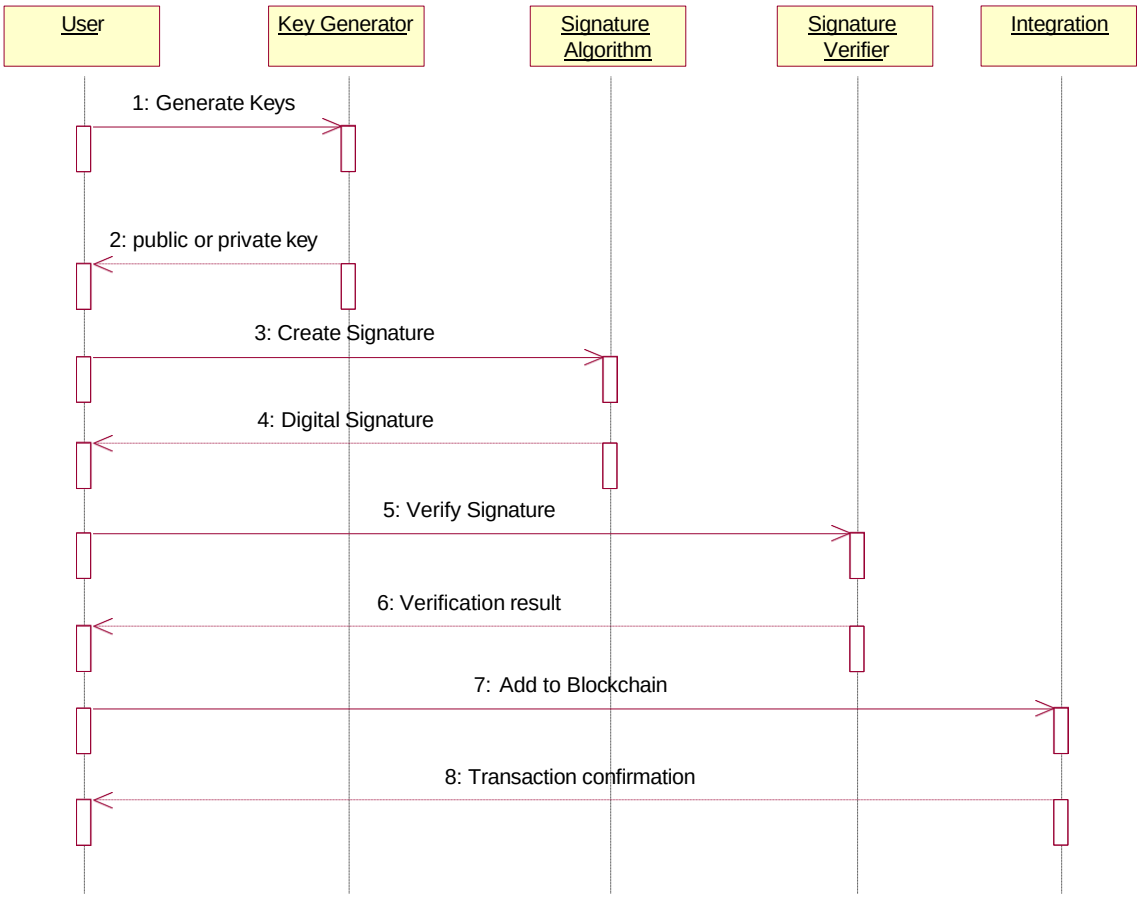


Fig.no 3.4.3 Sequence Diagram

Collaboration Diagram

In a collaboration diagram, the sequence of method calls is depicted using a numbering technique to illustrate how methods are called sequentially. This numbering indicates the order in which the methods are invoked. Let's consider the order management system to describe the collaboration diagram. The method calls in a collaboration diagram resemble those in a sequence diagram. However, the key difference lies in the fact that while the sequence diagram focuses solely on method invocation, the collaboration diagram provides additional information by illustrating the organization of objects involved in the interactions.`

Description: Collaboration diagram for key, signature algorithm, signature verifier, key generator

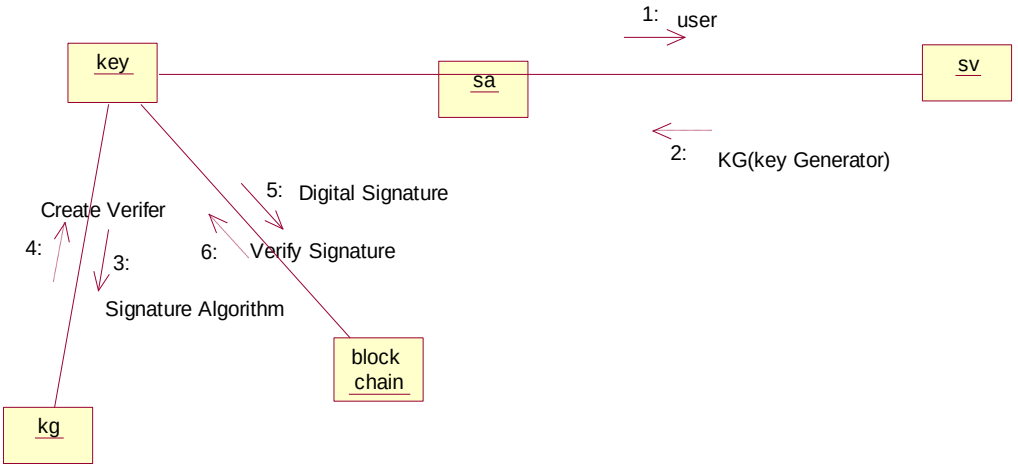


Fig.no 3.4.4 Collaboration Diagram

Activity Diagram

In the Unified Modeling Language (UML), an Activity Diagram is a type of behavioral diagram that illustrates the flow of actions or activities within a system. Commonly used to model workflows, business processes, or the detailed logic of a use case, it depicts the sequence and conditions for coordinating lower-level behaviors, thereby representing the dynamic aspects of the system.

Description: Activity Diagram for Create Signature, verify signature, integrate with block chain and confirmation transaction.

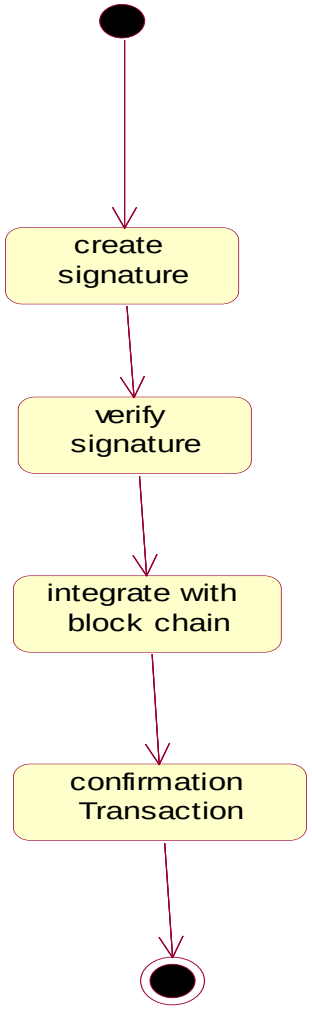


Fig.no 3.4.5 Activity Diagram

Component Diagram

A component diagram is used to break down a large object-oriented system into smaller components, making them more manageable. It provides a physical view of a system, such as executables, files, libraries, etc., that reside within the node.

It visualizes not only the relationships but also the organization between the components present in the system. It assists in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it requires an interface to interact and execute a function. It operates like a black box whose behavior is understood by the inputs and required interfaces.

Description: Component Diagram for User Key generator, signature algorithm, signature verify and block chain integration

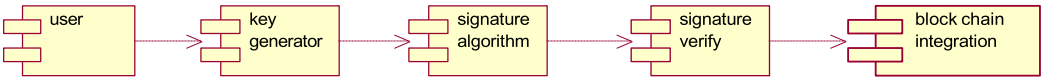


Fig.no 3.4.6 Component Diagram

Deployment Diagram

The deployment diagram illustrates the physical hardware on which the software will be deployed. It represents the static deployment view of a system, depicting the nodes and their relationships. It details how software is distributed across the hardware. The deployment diagram maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. As it involves multiple nodes, the relationships are depicted using communication pathways.

Description: Deployment Diagram for user device, cloud server, data base server and model server

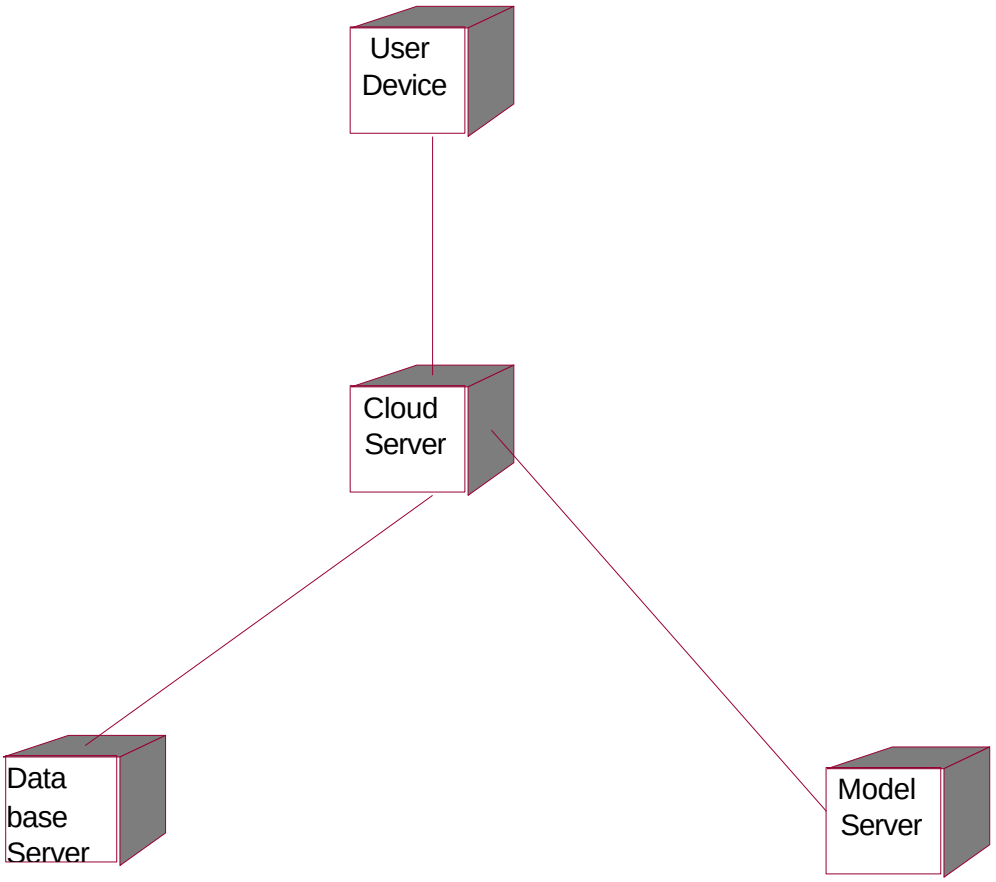


Fig.no 3.4.7 Deployment Diagram